AD.H

Arenadata Hyperwave

Arenadata Hyperwave (ADH) is a universal hybrid platform based on open-source components and proprietary developments, designed for storing, processing, and analyzing data of any structure and volume.

Version CURRENT Language: EN

Contents

1. Get started
  - Online installation
    - Install ADCM
    - Prepare hosts
    - Install ADH cluster
      - Create a cluster
      - Add services
      - Add hosts to a cluster
      - Add components
      - Configure services
      - Configure a cluster
      - Install a cluster
    - Install monitoring
      - Way 1. Monitoring service
      - Way 2. Monitoring cluster
        - Create a cluster
        - Add services
        - Add hosts to a cluster
        - Add components
        - Configure services
        - Install a cluster
        - Integrate with ADH cluster
  - Offline installation
    - Install ADCM
    - Prepare hosts
    - Install Enterprise Tools cluster
      - Create a cluster
      - Add services
      - Add hosts to a cluster
      - Add components
      - Install a cluster
    - Install ADH cluster
      - Create a cluster
      - Add services
      - Add hosts to a cluster
      - Add components
      - Configure services
      - Configure a cluster
      - Import ET settings
      - Install a cluster
    - Install monitoring
      - Way 1. Monitoring service
      - Way 2. Monitoring cluster
        - Configure Enterprise Tools cluster
        - Integrate with ADH cluster
2. How to
  - Airflow
    - Connect to Airflow
      - CLI
      - REST API
    - UI overview
    - Work with DAGs
      - Create a simple DAG
      - Work with TaskFlow
      - Use sensors
      - Customize DAG scheduling
      - Add custom operators and hooks
      - Generate DAGs dynamically
    - Logging
    - Service management via ADCM
  - Core configuration
    - Get client configurations
    - Service management via ADCM
  - Flink
    - UI overview
    - Connect to Flink
      - CLI
      - PyFlink
    - Flink SQL Gateway
    - Flink on YARN
    - Logging
    - Performance tuning
    - Build a streaming ETL with Flink
    - Service management via ADCM
      - Flink
      - Flink2
  - HBase
    - Connect to HBase
      - Use native API
      - Use external API
    - UI overview
    - Data querying
      - Quick start
      - Use filters
      - Use coprocessors
      - Scan over snapshot
      - Use medium-sized objects
    - Bulk loading
      - Built-in MapReduce jobs
    - Use snapshots
    - Replication
    - Backup and restore
    - Performance tuning
    - Logging
    - Service management via ADCM
  - HDFS
    - Connect to HDFS
    - UI overview
    - Basic file operations
      - Protect files
      - Copy data to HDFS
      - Add HDFS data directories
      - DataNode hot swapping
      - Remove files & directories
      - Enable WebHDFS
      - Check files via fsck
      - Local read
      - Use distcp
      - Use HttpFS
    - Storage management
      - Storage policies
      - Use quotas
      - Cache management
    - Replication
      - Replication factor
      - Rack awareness
      - Erasure coding
      - NameNode safemode
    - Backup and restore
      - Snapshots
      - NameNode recovery
    - Configure
    - Checkpointing
    - Integration with S3
    - Logging
    - Use Observer NameNode
    - Service management via ADCM
  - Hive
    - Connect to Hive
      - Beeline shell
      - JDBC
    - UI overview
    - Data querying
      - Work with tables
      - Computation example
      - Multi-table inserts
      - Analyze queries
      - Query types
        - Subqueries
        - Aggregate functions
        - Windowing functions
        - Use materialized views
    - Integrations
      - Hive on Spark
      - Hive with HBase
      - TEZ
        - View a Hive job
    - Performance tuning
    - Backup and restore
    - High availability
    - Logging
    - Use ACID
    - Iceberg tables
    - Service management via ADCM
  - HUE
    - UI overview
    - Add and use an interpreter
    - Iceberg tables
    - SPNEGO authentication
    - Service management via ADCM
  - Impala
    - Connect to Impala
      - impala-shell
      - JDBC
    - Web UIs overview
    - Performance tuning
    - Integrations
      - Impala with Hive
      - Impala with HBase
    - LDAP authentication
    - Iceberg tables
    - Service management via ADCM
  - Kyuubi
    - UI overview
    - LDAP authentication
    - Service management via ADCM
  - MapReduce
    - Quick start
    - Connect to MapReduce
      - CLI
      - REST API
    - Logging
    - Pluggable shuffle and sort
    - Encrypted shuffle
  - Monitoring
    - Service management via ADCM
  - Ozone
    - Connect to Ozone
      - CLI
    - UI overview
    - Ozone integration with cluster services
    - Rack awareness
    - Service management via ADCM
  - Phoenix
    - Connect to Phoenix
    - Data querying
      - Use SQLLine
      - JOIN
      - Subqueries
      - Sequences
      - Use salted tables
    - Map with HBase tables
    - Bulk loading
    - Performance tuning
    - Logging
  - Solr
    - Connect to Solr
      - CLI
    - Indexing
      - Overview
      - Index nested documents
      - Partial document updates
      - Reindex data
      - Deduplication
    - UI overview
    - Solr collections
    - Logging in Solr
    - Performance tuning
    - Service management via ADCM
  - Spark
    - Connect to Spark
      - spark3-submit & spark3-shell
    - Examples of launching jobs
      - In-memory batch processing
        - Dataset
        - RDD
        - DataFrame
    - Integrations
      - Spark and Hive
      - ADB Spark 3 Connector
        - Overview
        - Prerequisites
        - Usage examples
        - Options
      - ADQM Spark 3 Connector
        - Overview
        - Prerequisites
        - Usage example
        - Options
      - PySpark
      - Spark Connect
      - Work with Livy
    - Scheduler jobs
    - Logging
    - Performance tuning
    - Iceberg tables
    - Overwrite non-partitioned Hive tables
    - Service management via ADCM
  - SSM
    - UI overview
    - Define rules
    - Define actions
    - Rule usage examples
    - Service management via ADCM
  - Trino
    - Connect to Trino
      - CLI
      - JDBC
    - UI overview
    - Catalog management
      - Use DBCatalogManager
    - Manage fault-tolerant execution
    - Credential propagation
    - Trino Coordinator high availability
    - Integrations
      - Overview of Trino connectors
      - Iceberg catalog
      - Hive catalog
      - Trino ADB connector
        - Overview
        - Prerequisites
        - Usage example
        - Configuration
      - Trino OpenAPI connector
        - Overview
        - Usage example
        - Configuration
    - Security
      - LDAP authentication
      - Kerberos authentication
      - Ranger authorization with Hive policies
    - Service management via ADCM
  - YARN
    - UI overview
    - Connect to YARN
      - CLI
      - REST API
    - Schedulers
      - FairScheduler
      - CapacityScheduler
    - High availability
      - Overview
      - Examples
    - Logging
    - GPU on YARN
    - Service management via ADCM
  - Zeppelin
    - UI overview
    - Work with interpreters
      - Add a custom interpreter to a group
      - Install a third-party interpreter
      - User-managed interpreters
      - Write a new interpreter
    - Examples of notes
      - Phoenix
      - Solr
      - Spark
      - Python
    - Logging
    - Zeppelin performance tuning
    - Service management via ADCM
  - ZooKeeper
    - Connect to ZooKeeper
      - CLI
      - API
    - Work with znodes
    - Logging
    - Configure
    - Service management via ADCM
3. Tutorials
  - Cluster management via ADCM
    - Cluster actions
      - Check
      - Manage SSL
      - Manage Credential Encryption
      - Manage Kerberos
      - Prepare hosts
      - Reinstall status-checker
      - Start
      - Stop
      - Upgrade
    - Hosts actions
      - Add a host to a cluster
      - View host status
      - Remove a host from a cluster
    - Service actions
  - Security
    - ADPS
      - Overview
      - Ranger plugins
        - HBase plugin
        - HDFS plugin
        - Hive plugin
        - Impala plugin
        - Ozone plugin
        - Solr plugin
        - Spark3 plugin
        - Trino plugin
        - YARN plugin
    - Kerberos
      - Overview
      - MIT Kerberos
        - Kerberos server settings
        - Configure via ADCM
      - MS Active Directory
        - Install AD Certification Service
        - Create certificate template
        - Issue certificate template
        - Request a new certificate
        - Export CA certificate
        - Configure via ADCM
        - Enable Kerberos with prepared Active Directory server and no admin permissions
        - AD Kerberos common problems
      - FreeIPA
        - Overview
        - Configure via ADCM
        - Enable Kerberos with prepared FreeIPA server and no admin permissions
      - Samba
        - Configure via ADCM
      - Connect to Hive from DBeaver with Kerberos
      - SPNEGO
    - SSL
      - Requirements for SSL certificates
      - Set TLS version
    - Impersonation settings in ADH
4. References
  - Hadoop Command-line
    - Hadoop CLI
      - User commands
        - archive
        - checknative
        - CLASSNAME
        - classpath
        - conftest
        - credential
        - distch
        - distcp
        - dtutil
        - envvars
        - fs
        - gridmix
        - jar
        - jnipath
        - kerbname
        - kdiag
        - key
        - kms
        - trace
        - version
      - Administration commands
        - daemonlog
    - HDFS CLI
      - User commands
        - classpath
        - dfs
        - envvars
        - fetchdt
        - fsck
        - getconf
        - groups
        - httpfs
        - lsSnapshottableDir
        - jmxget
        - oev
        - oiv
        - oiv_legacy
        - snapshotDiff
        - version
      - Administration commands
        - balancer
        - cacheadmin
        - crypto
        - datanode
        - dfsadmin
        - dfsrouter
        - dfsrouteradmin
        - diskbalancer
        - ec
        - haadmin
        - journalnode
        - mover
        - namenode
        - nfs3
        - portmap
        - secondarynamenode
        - storagepolicies
        - zkfc
      - Debug commands
        - computeMeta
        - recoverLease
        - verifyMeta
    - FileSystem shell
      - appendToFile
      - cat
      - checksum
      - chgrp
      - chmod
      - chown
      - copyFromLocal
      - copyToLocal
      - count
      - cp
      - createSnapshot
      - deleteSnapshot
      - df
      - du
      - dus
      - expunge
      - find
      - get
      - getfacl
      - getfattr
      - getmerge
      - head
      - help
      - ls
      - lsr
      - mkdir
      - moveFromLocal
      - moveToLocal
      - mv
      - put
      - renameSnapshot
      - rm
      - rmdir
      - rmr
      - setfacl
      - setfattr
      - setrep
      - stat
      - tail
      - test
      - text
      - touch
      - touchz
      - truncate
      - usage
    - HDFS cheatsheet
  - HBase
    - HBase shell commands
      - General
        - processlist
        - status
        - table_help
        - version
        - whoami
      - DDL
        - alter
        - alter_async
        - alter_status
        - create
        - describe
        - disable
        - disable_all
        - drop
        - drop_all
        - enable
        - enable_all
        - exists
        - get_table
        - is_enabled
        - is_disabled
        - list
        - list_regions
        - locate_region
        - show_filters
      - Namespace
        - alter_namespace
        - create_namespace
        - describe_namespace
        - drop_namespace
        - list_namespace
        - list_namespace_tables
      - DML
        - append
        - count
        - delete
        - deleteall
        - get
        - get_counter
        - get_splits
        - incr
        - put
        - scan
        - truncate
        - truncate_preserve
      - Tools
        - assign
        - balancer
        - balancer_enabled
        - balance_switch
        - catalogjanitor_enabled
        - catalogjanitor_run
        - catalogjanitor_switch
        - cleaner_chore_enabled
        - cleaner_chore_run
        - cleaner_chore_switch
        - clear_block_cache
        - clear_compaction_queues
        - clear_deadservers
        - close_region
        - compact
        - compaction_state
        - compact_rs
        - flush
        - is_in_maintenance_mode
        - list_deadservers
        - major_compact
        - merge_region
        - move
        - normalize
        - normalizer_enabled
        - normalizer_switch
        - split
        - splitormerge_enabled
        - splitormerge_switch
        - trace
        - unassign
        - wal_roll
        - zk_dump
      - Replication
        - add_peer
        - append_peer_namespaces
        - append_peer_tableCFs
        - disable_peer
        - disable_table_replication
        - enable_peer
        - enable_table_replication
        - get_peer_config
        - list_peers
        - list_peer_configs
        - list_replicated_tables
        - remove_peer
        - remove_peer_namespaces
        - remove_peer_tableCFs
        - set_peer_bandwidth
        - set_peer_exclude_namespaces
        - set_peer_exclude_tableCFs
        - set_peer_namespaces
        - set_peer_replicate_all
        - set_peer_tableCFs
        - show_peer_tableCFs
        - update_peer_config
      - Snapshots
        - clone_snapshot
        - delete_all_snapshot
        - delete_snapshot
        - delete_table_snapshots
        - list_snapshots
        - list_table_snapshots
        - restore_snapshot
        - snapshot
      - Configuration
        - update_all_config
        - update_config
      - Quotas
        - list_quotas
        - list_snapshot_sizes
        - list_quota_snapshots
        - list_quota_table_sizes
        - set_quota
      - Security
        - grant
        - list_security_capabilities
        - revoke
        - user_permission
      - Procedures
        - abort_procedure
        - list_locks
        - list_procedures
      - Visibility labels
        - add_labels
        - clear_auths
        - get_auths
        - list_labels
        - set_auths
        - set_visibility
      - Rsgroup
        - add_rsgroup
        - balance_rsgroup
        - get_rsgroup
        - get_server_rsgroup
        - get_table_rsgroup
        - list_rsgroups
        - move_namespaces_rsgroup
        - move_servers_rsgroup
        - move_servers_namespaces_rsgroup
        - move_servers_tables_rsgroup
        - move_tables_rsgroup
        - remove_rsgroup
        - remove_servers_rsgroup
  - Hive
    - Hive on Spark parameters
    - Beeline command line
  - Impala
    - Impala shell commands
  - MapReduce
    - MapReduce CLI
      - User commands
        - archive
        - archives-logs
        - classpath
        - distcp
        - envvars
        - job
        - pipes
        - queue
        - version
      - Administration commands
        - frameworkuploader
        - historyserver
        - hsadmin
  - Ozone
    - Ozone CLI
      - bucket
        - addacl
        - clrquota
        - create
        - delete
        - getacl
        - info
        - link
        - list
        - removeacl
        - set-replication-config
        - setacl
        - setquota
        - update
      - key
        - addacl
        - cat
        - checksum
        - cp
        - delete
        - get
        - getacl
        - info
        - list
        - put
        - removeacl
        - rename
        - setacl
      - prefix
        - addacl
        - getacl
        - removeacl
        - setacl
      - snapshot
        - create
        - delete
        - diff
        - info
        - list
        - listDiff
      - token
        - cancel
        - get
        - print
        - renew
      - user
        - assign-admin
        - assign
        - get-secret
        - info
        - list
        - revoke-admin
        - revoke
        - set-secret
      - volume
        - addacl
        - clrquota
        - create
        - delete
        - getacl
        - info
        - list
        - removeacl
        - setacl
        - setquota
        - update
  - Spark
    - spark3-shell
    - spark3-submit
  - SSM
    - Actions
    - Rules
  - YARN
    - YARN CLI
      - User commands
        - application
        - applicationattempt
        - classpath
        - container
        - envvars
        - jar
        - logs
        - node
        - queue
        - top
        - version
      - Administration commands
        - daemonlog
        - nodemanager
        - proxyserver
        - registrydns
        - resourcemanager
        - rmadmin
        - schedulerconf
        - scmadmin
        - sharedcachemanager
        - timelineserver
  - Configuration parameters
  - Monitoring
    - ADH monitoring metrics
5. Planning guide
  - Software requirements
  - Hardware requirements
    - Depending on node types
      - NameNode
      - DataNode
      - YARN
      - PostgreSQL nodes used with Hive Metastore
    - Depending on workload patterns
  - Filesystem requirements
  - Network requirements
    - ADH network ports
  - Typical configurations
    - Pilot cluster
  - Custom Java settings
6. Concepts
  - Architecture and data model
    - Airflow
    - Flink
      - Overview
    - HBase
      - Architecture
      - Data model
    - HDFS
      - Architecture
        - Scalability
    - Hive
      - TEZ
    - Iceberg
      - Architecture
    - Impala
      - Architecture
    - Kyuubi
      - Architecture
      - Kyuubi vs HiveServer vs Spark Thrift Server
      - Share levels
    - MapReduce
      - Architecture
      - Overview
    - Monitoring
    - Ozone
      - Architecture
      - HDFS vs Ozone
    - Phoenix
      - Overview
    - Solr
      - Overview
    - Spark
      - Spark Connect
    - SSM
      - Architecture
    - Trino
      - Architecture
    - YARN
      - Architecture
    - Zeppelin
    - ZooKeeper
  - Security
    - Kerberos
7. Release notes
  - ADH releases
  - Supported services
  - Version compatibility matrix
  - Known issues

Get started with Arenadata Hyperwave

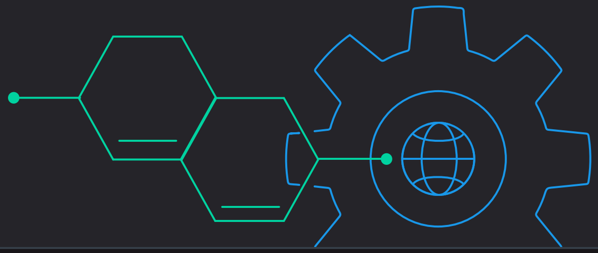
To Table of Contents

This tutorial guides you through the process of installing an Arenadata Hyperwave (ADH) cluster. The easiest way to install it is to use Arenadata Cluster Manager (ADCM) — a separate Arenadata product designed for simple, convenient, and fast software deployment and exploitation.

You can install ADH via ADCM in two ways.

Online installation

on hosts with access to the Internet



Offline installation

on hosts without access to the Internet

Online installation

To Table of Contents

This topic describes **online** installation of ADH. It can be applied to the hosts with access to the Internet.

The process of online installation via ADCM includes the following steps.

Step 1.

Install ADCM

Step 2.

Prepare hosts

ARENADATA
HADOOP

Step 3.

Install ADH cluster

Step 4.

Install monitoring

Install ADCM

Daria Barysheva

Contents

To Table of Contents

- Step 1. Check prerequisites
  - Software and hardware requirements
  - Install Docker
  - Disable SELinux
  - Stop firewalld
  - Allow access to the Docker CLI without root privileges
- Step 2. Install ADCM
- Step 3. Start ADCM
- Step 4. Check installation
  - Check the Docker container status
  - Check the 8000 port availability
  - Check the ADCM web-interface
  - Set the ADCM URL
- Step 5. Update ADCM

ADCM requires a separate host. Before installing ADCM, we recommend to read its documentation.

The installation steps are listed below.

Step 1. Check prerequisites

Software and hardware requirements

ADCM is a software that is distributed in the form of a Docker image. To install it, you need:

- One of the following operating systems:
  - Ubuntu 22.04
  - CentOS 7
  - RHEL 7
  - Alt Linux SP 8
  - Alt Linux SP 10.2
  - Astra Linux SE 1.7 Orel
  - Astra Linux 1.7.6 Voronezh
  - RED OS 7.3 Certified edition
- A user account with `sudo` privileges.
- An access to the official repository of CentOS Extras and CentOS Base. This repository is enabled by default, but if you have disabled it — you need to re-enable it (e.g. via editing the repository configuration file in the */etc/yum.repos.d/* directory).
- The software package installer YUM/APT.
- The hardware that meets the following conditions:

> **IMPORTANT**
>
> The following system requirements are **minimal**. The target sizing should be calculated based on the customer requirements.

  - **CPU**: 4 CPU cores;
  - **RAM**: 16 GB;
  - **HDD**: 50 GB.

> **IMPORTANT**
>
> Starting with ADCM 2.7.0, any Container Registry Client with downloaded images of ADPG, ADCM, and ADCM Installer will be required to install ADPG without the Internet connection and use it as the ADCM database. The installation is done via the ADCM Installer by specifying the URL of the local storage of the ADCM and ADPG images.

To grant the user the necessary rights to install ADCM, do the following:

1. Grant the `root` privileges for the user:

CentOS 7/RHEL 7/RED OS 7.3/Alt Linux SP 8

```
$ sudo usermod -aG wheel <user>
```

where `<user>` is a username.

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
$ sudo usermod -aG sudo <user>
```

where `<user>` is a username.

2. Make sure that the */etc/sudoers* file contains the record specified below depending on the operating system with the following command:

```
$ sudo vi /etc/sudoers
```

To return to the command line, press `Esc`, type `:q!`, and press `Enter`.

CentOS 7/RHEL 7/RED OS 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Install Docker

> **CAUTION**
>
> Do not install Docker from the official Docker site. Instead, use the official repositories of your Linux distribution. The repositories provided by operating system developers tend to be more stable and secure.

Below are the step-by-step procedures of Docker installation on the examples of several operating systems supported by ADCM. These steps may differ for operating system versions other than those specified on tabs.

CentOS 7/RHEL 7

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo yum update -y
```

2. Install the packages required for Docker installation and usage:

```
$ sudo yum install -y yum-utils docker device-mapper-persistent-data lvm2
```

where:

- `yum-utils` is the package that provides the `yum-config-manager` utility;
- `device-mapper-persistent-data` and `lvm2` are packages required by the devicemapper storage driver.

3. Start Docker:

```
$ sudo systemctl start docker
```

4. Enable Docker as a system service:

```
$ sudo systemctl enable docker
```

Ubuntu 22.04

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo apt-get update
```

2. Install the packages required for Docker installation and usage:

```
$ sudo apt-get install ca-certificates curl
```

where:

- `ca-certificates` is the package of Root CA certificates;
- `curl` is the package for working with web resources.

Confirm the use of additional disk space during the execution of the command.

3. Add the official GPG key for the Docker repository:

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

4. Add the Docker repository:

```
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "jammy") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Update all packages, including packages for the Docker repository:

```
$ sudo apt-get update
```

6. Install Docker Engine:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Confirm the use of additional disk space during the execution of the command.

7. Start Docker:

```
$ sudo systemctl start docker
```

8. Enable Docker as a system service:

```
$ sudo systemctl enable docker
```

Alt Linux SP 8

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo apt-get update
```

2. Install Docker Engine:

```
$ sudo apt-get install docker-engine
```

Confirm the use of additional disk space during the execution of the command.

3. Start and enable Docker as a system service:

```
$ sudo systemctl enable --now docker
```

Astra Linux SE 1.7 Orel

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo apt-get update
```

2. Install the packages required for Docker installation and usage:

```
$ sudo apt-get install ca-certificates curl net-tools
```

where:

- `ca-certificates` is the package of Root CA certificates;
- `curl` is the package for working with web resources;
- `net-tools` is the package for controlling the network subsystem of the Linux kernel, in particular for executing the `netstat` command.

Confirm the use of additional disk space during the execution of the command.

3. Add the official GPG key for the Docker repository:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

4. Add the Docker repository:

```
$ echo "deb [arch=amd64] https://download.docker.com/linux/debian stretch stable" | sudo tee -a /etc/apt/sources.list
```

5. Update all packages, including packages for the Docker repository:

```
$ sudo apt-get update
```

6. Install Docker Engine:

```
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

Confirm the use of additional disk space during the execution of the command.

7. Start Docker:

```
$ sudo systemctl start docker
```

8. Enable Docker as a system service:

```
$ sudo systemctl enable docker
```

RED OS 7.3

1. Install Docker Engine:

```
$ sudo dnf install docker-ce docker-ce-cli
```

Confirm the use of additional disk space during the execution of the command.

2. Start and enable Docker as a system service:

```
$ sudo systemctl enable docker --now
```

Disable SELinux

> **NOTE**
>
> It is recommended to disable SELinux only if CentOS 7 or RHEL 7 is used.

To disable SELinux permanently, set `SELINUX=disabled` in the */etc/selinux/config* file, and reboot your system.

You can edit the file via the `vi` or `vim` command:

```
$ sudo vi /etc/selinux/config
```

The content of the changed file should look like this:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

You can also disable SELinux temporarily via the following command:

```
$ sudo setenforce 0
```

Still, it is strongly recommended to disable SELinux permanently, so that it does not restart on each system reboot.

> **CAUTION**
>
> Do not forget to reboot your host after SELinux is disabled.

Stop firewalld

If you use the firewalld service, stop it before installing ADCM.

Before stopping firewalld, make sure that it is installed, started, and enabled:

```
$ sudo systemctl status firewalld
```

The example of the command execution result:

```
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-12-02 10:02:57 UTC; 3min 25s ago
     Docs: man:firewalld(1)
 Main PID: 3023 (firewalld)
    Tasks: 2
   Memory: 22.8M
   CGroup: /system.slice/firewalld.service
           └─3023 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid
```

Stop firewalld:

```
$ sudo systemctl stop firewalld
```

As an alternative, you can disable the firewalld service, so that it will not apply rules to network packets:

```
$ sudo systemctl disable firewalld
```

Allow access to the Docker CLI without root privileges

If in the future you want to access the Docker CLI without using `root` privileges, you should create a `docker` user group (if it does not exist) and add the current user to this group with the certain rights:

```
$ sudo groupadd docker
$ sudo usermod -a -G docker $USER
```

After running these commands, you should re-login under the current user.

Step 2. Install ADCM

> **NOTE**
>
> - It is recommended to configure PostgreSQL as an external database or to deploy an embedded database using ADCM Installer. For more information on configuring PostgreSQL, see the Work with an external database article.
> - Starting with ADCM 2.5.0, the minimum required version of PostgreSQL is 14.
> - To see the full list of images, refer to the ADCM repository on Docker Hub.
> - ADCM has some mandatory persistent information stored in the */adcm/data* directory inside a container. This means that you have to map a volume to that directory and provide backups for that volume.

To install ADCM, perform the following steps:

1. Pull a Docker image from the Arenadata Docker Registry:

```
$ sudo docker pull hub.arenadata.io/adcm/adcm:<version>
```

where `<version>` is a desired ADCM image version in one of the following formats:

- `<major>.<minor>.<patch>` — if you need a specific ADCM patch. For example, `2.0.0`.
- `<major>.<minor>` — if you need the last patch within the selected ADCM version. For example, `2.0`.

> **IMPORTANT**
>
> - `latest` was used for earlier versions of ADCM and is no longer supported starting from version `2.0.0`.
> - Before ADCM 2.0.0, the following format was used for versioning: `YYYY.MM.DD.HH`.

2. Add a container based on that image:

> **NOTE**
>
> The command below creates a container with ADCM on the `8000` port and establishes a connection to an external database. All its data will be stored on the host machine in the */opt/adcm/* directory. The environment variables description is given in the table below.

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="
<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e
DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

If you want to run ADCM **with SELinux enabled**, a Docker container should be created with the `:Z` option in the data catalog parameter:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data:Z hub.arenadata.io/adcm/adcm:<version>
```

Optionally, when you create a container, you can specify the environment variables listed below. Each variable is specified in the following format: `-e <name>="<value>"`. For example:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e LOG_LEVEL="INFO" hub.arenadata.io/adcm/adcm:<version>
```

> **NOTE**
>
> Special environment variables, such as `STATUS_LOG_LEVEL`, `ADCM_LOG_LEVEL`, `AUDIT_LOG_LEVEL`, `LDAP_LOG_LEVEL`, `BACKGROUND_TASKS_LOG_LEVEL`, and `TASK_RUNNER_LOG_LEVEL` have higher priority than the `LOG_LEVEL` variable.

Environment variables

Name	Description
DB_HOST	Name or IP address of the host where the database is deployed
DB_PORT	TCP port number that is used to connect to the database
DB_USER	Username that is used to connect to the database
DB_NAME	Database name
DB_PASS	Password that is used to connect to the database

DB_OPTIONS	Options for connecting to the database:
	- `sslmode` — flag that is used for establishing an SSL connection. Possible values:
	  - `disable` — only try a non-SSL connection.
	  - `allow` — first try a non-SSL connection. If that fails, try an SSL connection.
	  - `prefer` (default) — first try an SSL connection. If that fails, try a non-SSL connection.
	  - `require` — only try an SSL connection. If a root certificate authority (CA) file is present, verify the certificate in the same way as if `verify-ca` was specified.
	  - `verify-ca` — only try an SSL connection, and verify that the server certificate is issued by a trusted CA.
	  - `verify-full` — only try an SSL connection, and verify that the server certificate is issued by a trusted CA and that the requested server host name matches that in the certificate.
	- `sslcert` — specifies the file name of the client SSL certificate. This parameter is ignored if the connection is made without SSL.
	- `sslkey` — specifies the location for the secret key used for the client certificate. This parameter is ignored if the connection is made without SSL.
	- `sslrootcert` — specifies the name of a file containing SSL certificate authority certificate(s). If the file exists, the server certificate will be verified to be signed by one of these authorities. A special system value may be specified instead, in which case the system trusted CA roots will be loaded. The exact locations of these root certificates differ by SSL implementation and platform.

DEFAULT_ADCM_URL	URL of the host where the ADCM container is deployed. Use the following format: `http://<IP>:8000/`. If `DEFAULT_ADCM_URL` is not specified, you should set the URL manually in the ***ADCM's URL*** field on the ***Settings*** page
LOG_LEVEL	Log level for all log types. Possible values: - `DEBUG` - `INFO` - `WARNING` - `ERROR` (default value) - `CRITICAL`
STATUS_LOG_LEVEL	Log level for *status.log*. Possible values: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
ADCM_LOG_LEVEL	Log level for *adcm.log* and *adcm_debug.log*. Possible values: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
AUDIT_LOG_LEVEL	Log level for *audit.log*. Possible values: - `DEBUG` - `INFO` (default value) - `WARNING` - `ERROR` - `CRITICAL`
LDAP_LOG_LEVEL	Log level for *ldap.log*. Possible values: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
BACKGROUND_TASKS_LOG_LEVEL	Log level for *cron_task.log*. Possible values: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
TASK_RUNNER_LOG_LEVEL	Log level for *task_runner.err*. Possible values: - `DEBUG` - `INFO` - `WARNING` - `ERROR` - `CRITICAL`
MIGRATION_MODE	Flag that indicates whether migration mode (ability to migrate ADCM database from SQLite to PostgreSQL) is turned on. For more information, see Migration. Possible values: - `0` — migration mode is off; - `1` — migration mode is on.
CONSUL_URL	HashiCorp Consul server URL. Use the following format: `http://<IP>:8500/`
CONSUL_DATACENTER	HashiCorp Consul Datacenter. If the variable value is not specified, the value specified in HashiCorp Consul by default will be used
CONSUL_CACERT_FILE	Path to the certificate authority (CA) certificate file if encryption is enabled in HashiCorp Consul
CONSUL_CLIENT_CERT_FILE	Path to the client certificate file if encryption is enabled in HashiCorp Consul
CONSUL_CLIENT_KEY_FILE	Path to the client private key file if encryption is enabled in HashiCorp Consul

Step 3. Start ADCM

ADCM is just a container operated by Docker. So you are able to use simple Docker commands to start and stop ADCM:

- To **start ADCM**, use the following command.

```
$ sudo docker start adcm
```

- To **stop ADCM**, use the following command.

```
$ sudo docker stop adcm
```

- In order to provide the **automatic start** of the Docker container after unexpected errors, use the following command.

```
$ sudo docker update --restart=on-failure adcm
```

> **NOTE**
>
> - For more information about Docker restart policies, refer to the Docker documentation.
> - If you allowed your user to access the Docker CLI without `root` privileges, then you can omit the `sudo` keyword from the commands listed above.

Step 4. Check installation

Check the Docker container status

To make sure that the Docker container `adcm` is successfully deployed, run the following command:

```
$ sudo docker container ls
```

The example of the command execution result is listed below. Make sure that the `STATUS` value is equal to `Up`.

```
CONTAINER ID   IMAGE                            COMMAND             CREATED       STATUS          PORTS
NAMES
74b5628146e5   hub.arenadata.io/adcm/adcm:2.0   "/etc/startup.sh"   4 days ago    Up 21 minutes   0.0.0.0:8000-
>8000/tcp   adcm
```

Check the 8000 port availability

After ADCM is installed and started, its web interface should become available on the `8000` port of the deployed container.

For Ubuntu 22.04, first install the packages required for executing the `netstat` command:

```
$ sudo apt-get install net-tools
```

You can check the 8000 port availability using these commands:

- The `netstat` command — to display information about port listening.

```
$ sudo netstat -ntpl | grep 8000
```

The example of the command execution result:

```
tcp6       0      0 :::8000                 :::*                    LISTEN
1514/docker-proxy-c
```

- The `curl` command — to check the URL connection.

```
$ curl http://localhost:8000
```

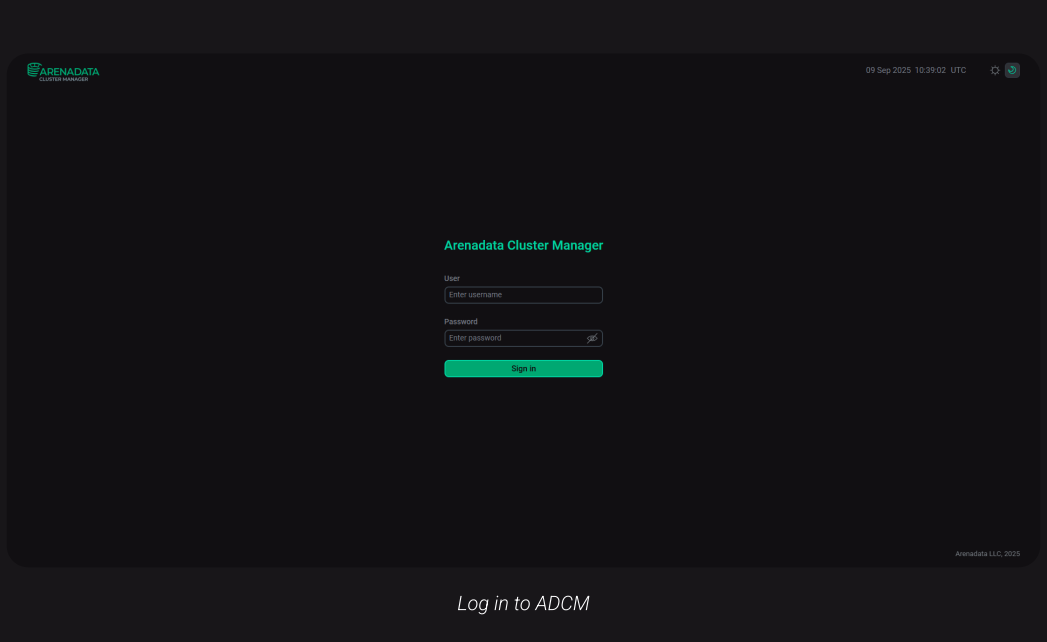
The example of the command execution result:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Arenadata Cluster Manager</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/favicon.ico">
<link rel="stylesheet" href="styles.10db6328264e0907c52f.css"></head>
<body>
  <app-root></app-root>
<script src="runtime-es2015.7eb406ed18bf0258cd35.js" type="module"></script><script src="runtime-es5.7eb406ed18bf0258cd35.js"
nomodule defer></script><script src="polyfills-es5.2e224d70daec4412d3c2.js" nomodule defer></script><script src="polyfills-es2015.
37e0553ac06970d6a5b5.js" type="module"></script><script src="main-es2015.39851da0ebf9ed6fec45.js" type="module"></script><script
src="main-es5.39851da0ebf9ed6fec45.js" nomodule defer></script></body>
</html>
```

Check the ADCM web-interface

To log in to ADCM, use the user with administrator rights that is created by default:

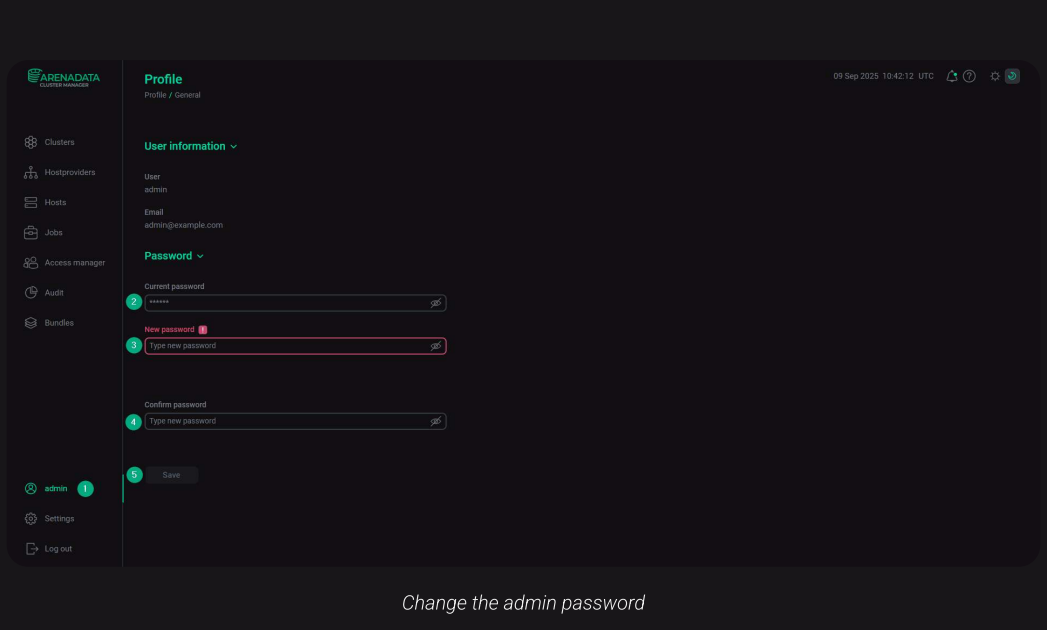
- ***User***: admin
- ***Password***: admin



Log in to ADCM

Later you can change the password if necessary. To do this:

1. Open the user profile in the left navigation menu.
2. Enter the current password in the ***Current password*** field.
3. Fill in the ***New password*** field.
4. Confirm a new password in the ***Confirm password*** field.
5. Click ***Save***.

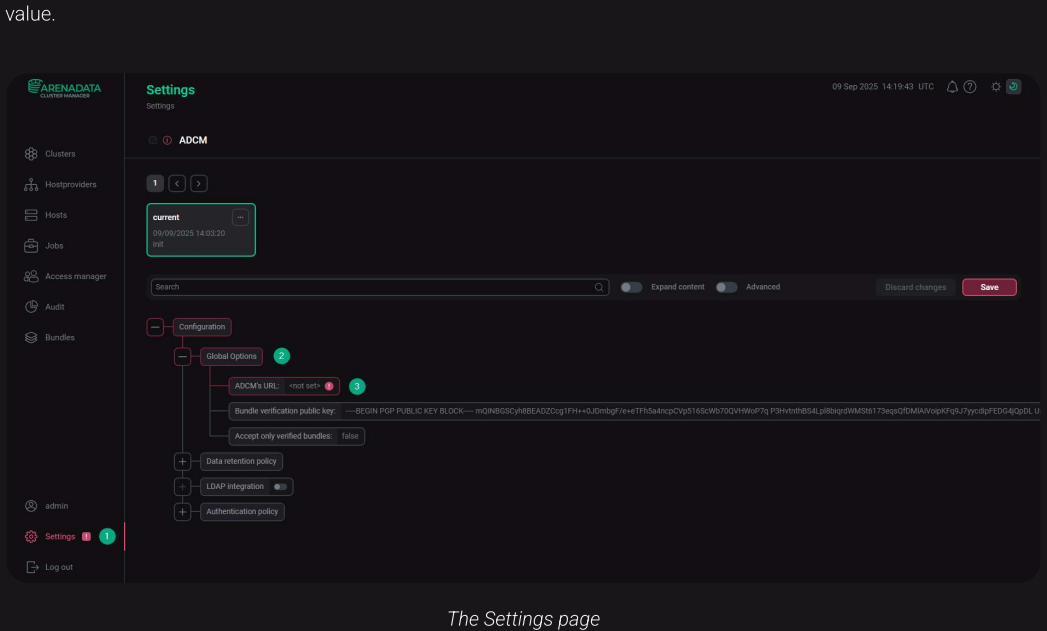


Change the admin password

Set the ADCM URL

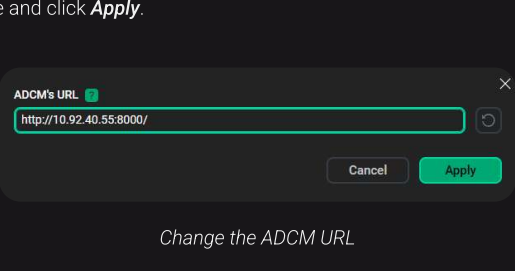
For successful communication between ADCM and product clusters deployed via ADCM, it is important to correctly define the external ADCM address. This address is used by cluster components for sending information about their state to ADCM. To specify the ADCM URL, use the environment variable `DEFAULT_ADCM_URL` when creating the ADCM container. If you did not specify that variable, you should manually install the URL in the ADCM web interface. To do this, follow the steps:

1. Select the ***Settings*** item in the left navigation menu.
2. Expand the ***Global Options*** node in the tree of configuration settings.
3. Click the ***ADCM's URL*** field value.



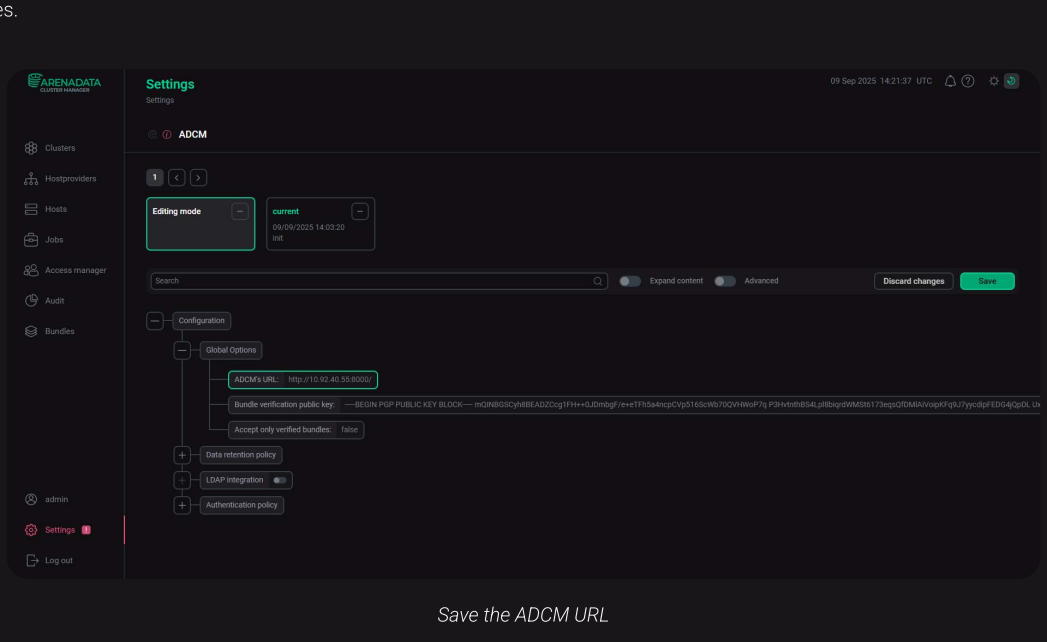
The Settings page

4. In the window that opens, edit the ***ADCM's URL*** field value and click ***Apply***.


ADCM's URL: http://10.92.40.55:8000/

Change the ADCM URL

5. Click ***Save*** to apply changes.



Save the ADCM URL

Step 5. Update ADCM

> **CAUTION**
>
> - Before upgrading, it is strongly recommended to back up ADCM data. For more information, see Backup and restore.
> - Downgrade of the ADCM version is not supported.
> - It is not recommended to update ADCM more than 5 minor versions ahead (see `<minor>` in the versioning format below).
> - Before upgrading ADCM to version 2.7.0, migrate data from SQLite to PostgreSQL. Migration can be performed in any ADCM version (up to 2.6.0).

If you need to upgrade the previously installed version of ADCM, do the following:

1. Stop the `adcm` container:

```
$ sudo docker stop adcm
```

2. Remove the `adcm` container:

```
$ sudo docker container rm adcm
```

3. Download the required Docker image from the Arenadata Docker Registry:

```
$ sudo docker pull hub.arenadata.io/adcm/adcm:<version>
```

where `<version>` is a desired ADCM image version in one of the following formats:

- `<major>.<minor>.<patch>` — if you need a specific ADCM patch. For example, `2.0.0`.
- `<major>.<minor>` — if you need the last patch within the selected ADCM version. For example, `2.0`.

> **IMPORTANT**
>
> - `latest` was used for earlier versions of ADCM and is no longer supported starting from version `2.0.0`.
> - Before ADCM 2.0.0, the following format was used for versioning: `YYYY.MM.DD.HH`.

4. Create a new container based on the downloaded image:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="
<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e
DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

> **NOTE**
>
> For more information on the environment variables (`DB_HOST`, `DB_PORT`, `DB_USER`, `DB_NAME`, `DB_PASS`, and `DB_OPTIONS`), see the Installation article.

5. Start the new `adcm` container:

```
$ sudo docker start adcm
```

Prepare hosts

Contents

To Table of Contents

User permissions for ADH installation

To manage a cluster and its nodes via ADCM, it is necessary to prepare hosts for a cluster and link these hosts to ADCM. This is the function of ADCM **hostproviders** — special plugins for ADCM, which encapsulate the complex mechanism of interaction between ADCM and the hosts controlled by it.

ADCM supports several types of hostproviders. For more information on how to configure hostproviders and use them to create hosts, refer to the following documentation:

- SSH
- Yandex Cloud
- VCloud (VMware)
- Cloud.ru Advanced
- CROC Cloud

CAUTION

When creating ADH hosts via ADCM, you should specify their FQDNs. The maximum FQDN length is 38 symbols.

User permissions for ADH installation

If you specify a custom UNIX user in the hostprovider settings, which will be used to run ADH installation commands, make sure that the specified user can run commands as a superuser.

To grant the user the necessary rights to install ADH, do the following:

1. Grant the `root` privileges for the user:

CentOS 7/RHEL 7/RED OS 7.3/Alt Linux SP 8

```
$ sudo usermod -aG wheel <user>
```

where `<user>` is a username.

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
$ sudo usermod -aG sudo <user>
```

where `<user>` is a username.

2. Make sure that the */etc/sudoers* file contains the record specified below depending on the operating system with the following command:

```
$ sudo vi /etc/sudoers
```

To return to the command line, press `Esc`, type `:q!`, and press `Enter`.

CentOS 7/RHEL 7/RED OS 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Create a cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

Step 1. Download a cluster bundle

Step 2. Upload a cluster bundle to ADCM

Step 3. Create a cluster based on the uploaded bundle

In ADCM a **cluster** means a set of services running on a set of hosts.

Clusters distributions for ADCM are made in the form of so-called bundles. Generally a bundle is a regular archive that includes a description and software logic of cluster deployment and exploitation.

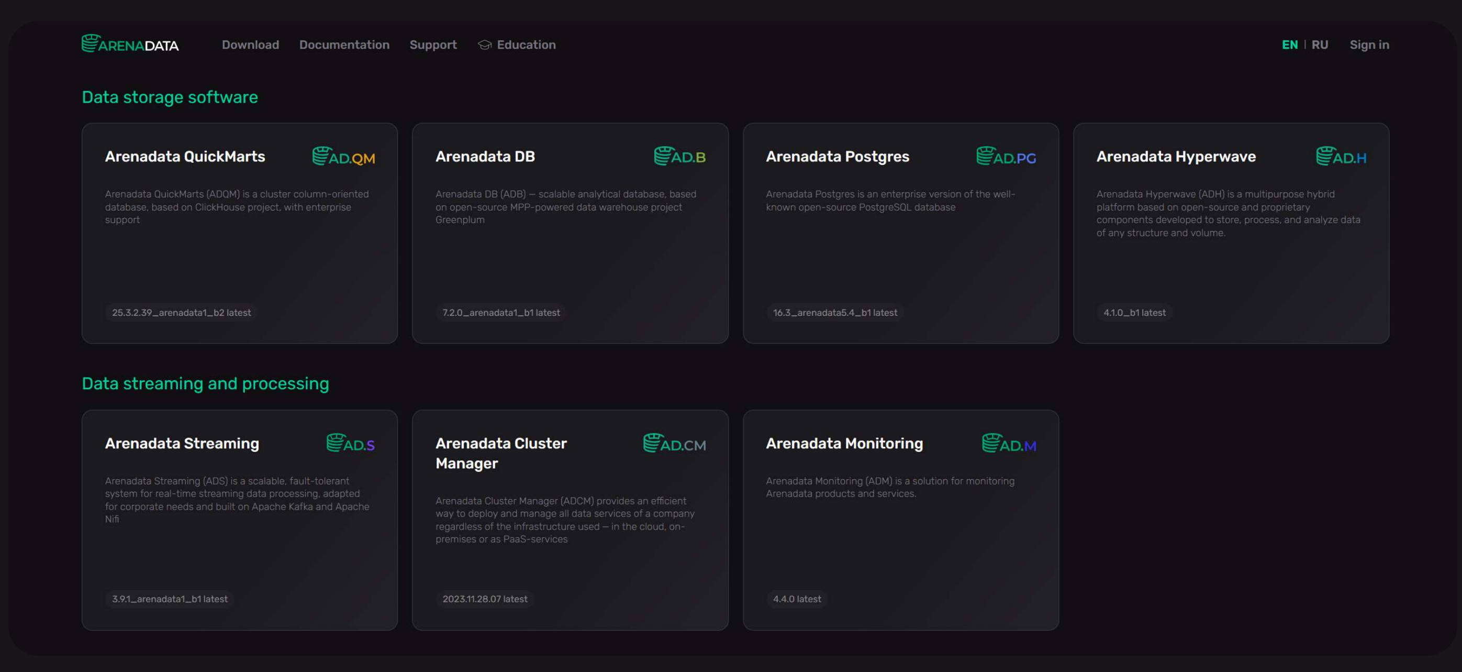
The steps for adding a cluster to ADCM are listed below.

NOTE

If you use the Enterprise Edition, skip the step 1, get a bundle from the Arenadata support team, and move to the step 2.

Step 1. Download a cluster bundle

1. Go to https://network.arenadata.io/ and choose ***Arenadata Hyperwave***.



ARENADATA
Download
Documentation
Support
Education
EN | RU
Sign in

Data storage software

Arenadata QuickMarts	Arenadata QuickMarts (ADQM) is a cluster column-oriented database, based on ClickHouse project, with enterprise support	25.3.2.39_arenadata1_b2 latest
Arenadata DB	Arenadata DB (ADB) — scalable analytical database, based on open-source MPP-powered data warehouse project Greenplum	7.2.0_arenadata1_b1 latest
Arenadata Postgres	Arenadata Postgres is an enterprise version of the well-known open-source PostgreSQL database	16.3_arenadata5.4_b1 latest
Arenadata Hyperwave	Arenadata Hyperwave (ADH) is a multipurpose hybrid platform based on open-source and proprietary components developed to store, process, and analyze data of any structure and volume.	4.1.0_b1 latest

Data streaming and processing

Arenadata Streaming	Arenadata Streaming (ADS) is a scalable, fault-tolerant system for real-time streaming data processing, adapted for corporate needs and built on Apache Kafka and Apache Nifi	3.9.1_arenadata1_b1 latest
Arenadata Cluster Manager	Arenadata Cluster Manager (ADCM) provides an efficient way to deploy and manage all data services of a company regardless of the infrastructure used — in the cloud, on-premises or as PaaS-services	2023.11.28.07 latest
Arenadata Monitoring	Arenadata Monitoring (ADM) is a solution for monitoring Arenadata products and services.	4.4.0 latest

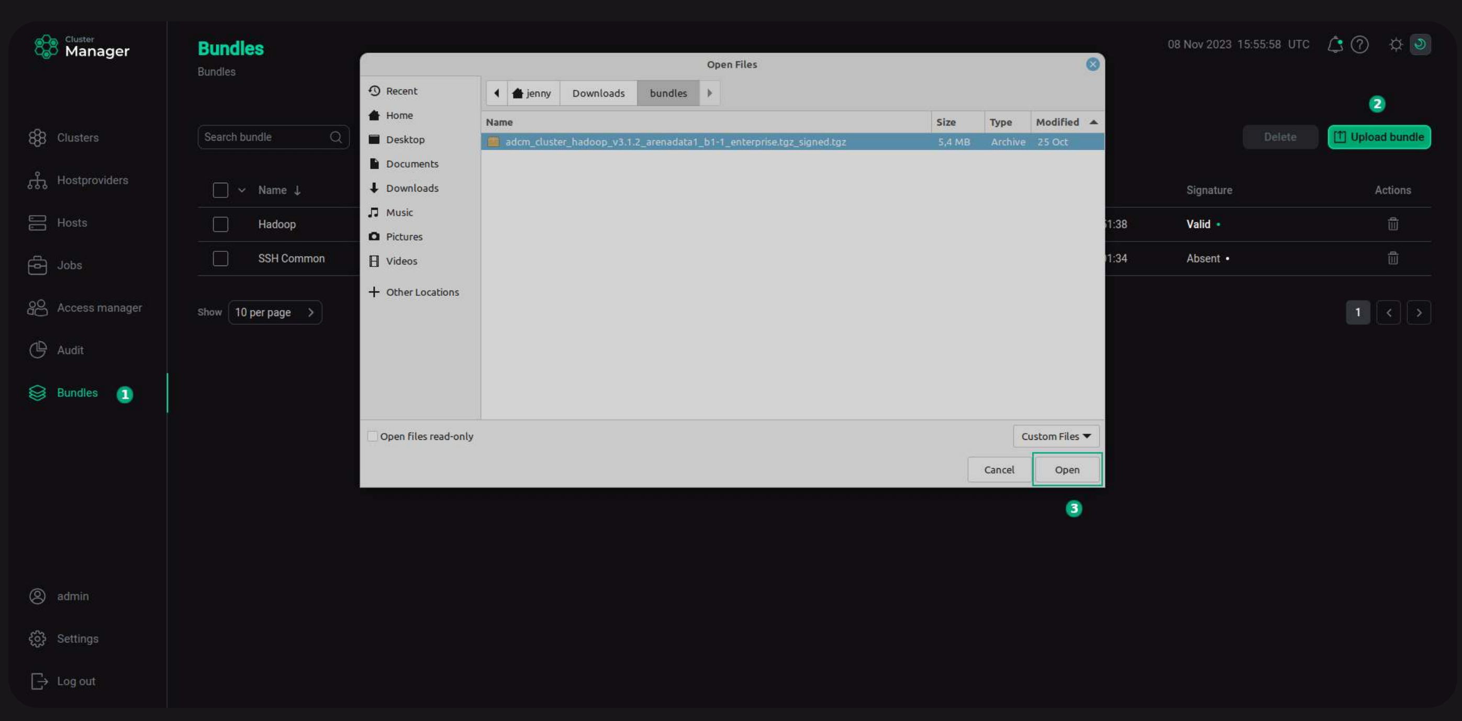
Switch to the bundle download page

2. On the next page, download a bundle for the required version of ADH.

Step 2. Upload a cluster bundle to ADCM

To upload a bundle to ADCM, follow the steps:

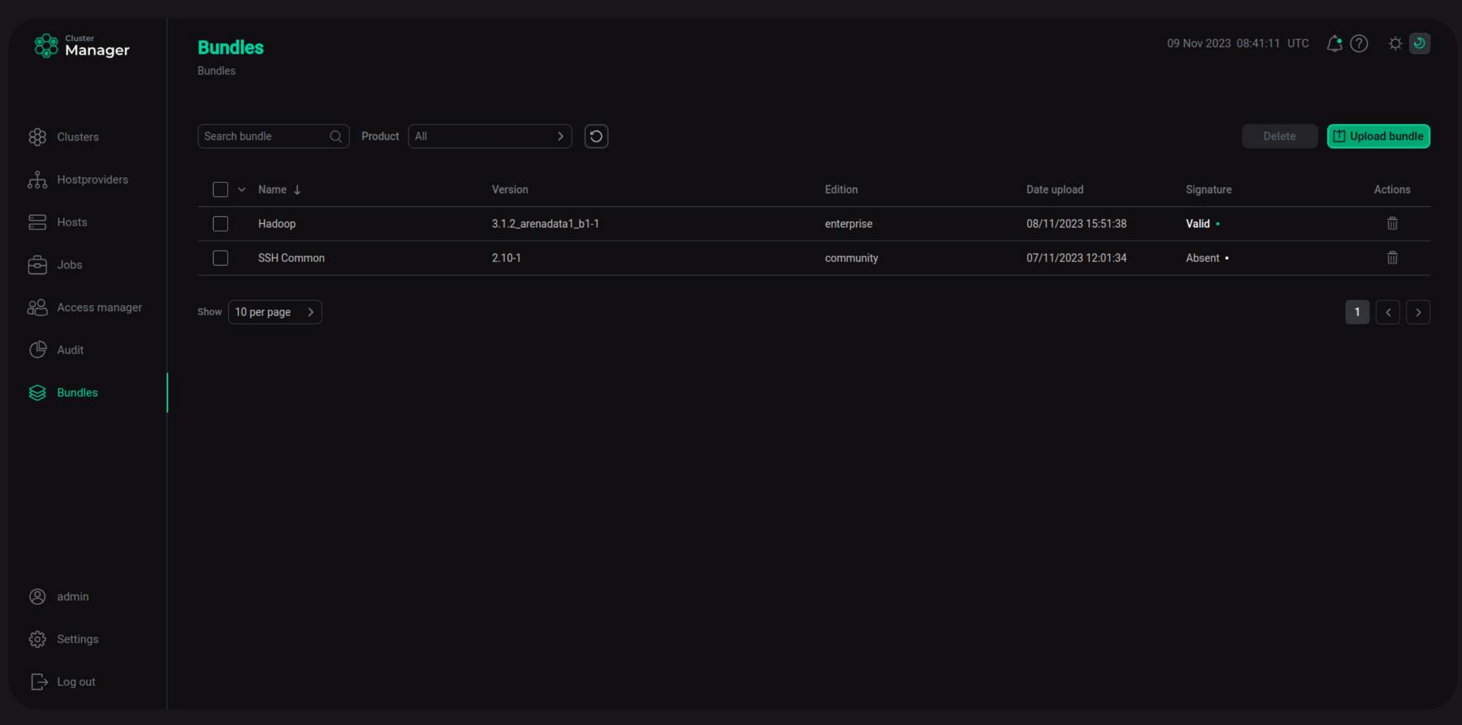
1. Select the ***Bundles*** item in the left navigation menu and click ***Upload bundle***.
2. Select a bundle in the Open File Dialog.



Manager
Bundles
Open Files
Cancel
Open

Upload a bundle

3. As a result of the performed actions, a bundle is displayed on the ***Bundles*** page.



Manager
Bundles
Bundles
09 Nov 2023 08:41:11 UTC
Search bundle
Product
All
Delete
Upload bundle
Show
10 per page
1

Name	Version	Edition	Date upload	Signature	Actions
Hadoop	3.1.2_arenadata1_b1-1	enterprise	08/11/2023 15:51:38	Valid	
SSH Common	2.10-1	community	07/11/2023 12:01:34	Absent	

The result of successful uploading a bundle

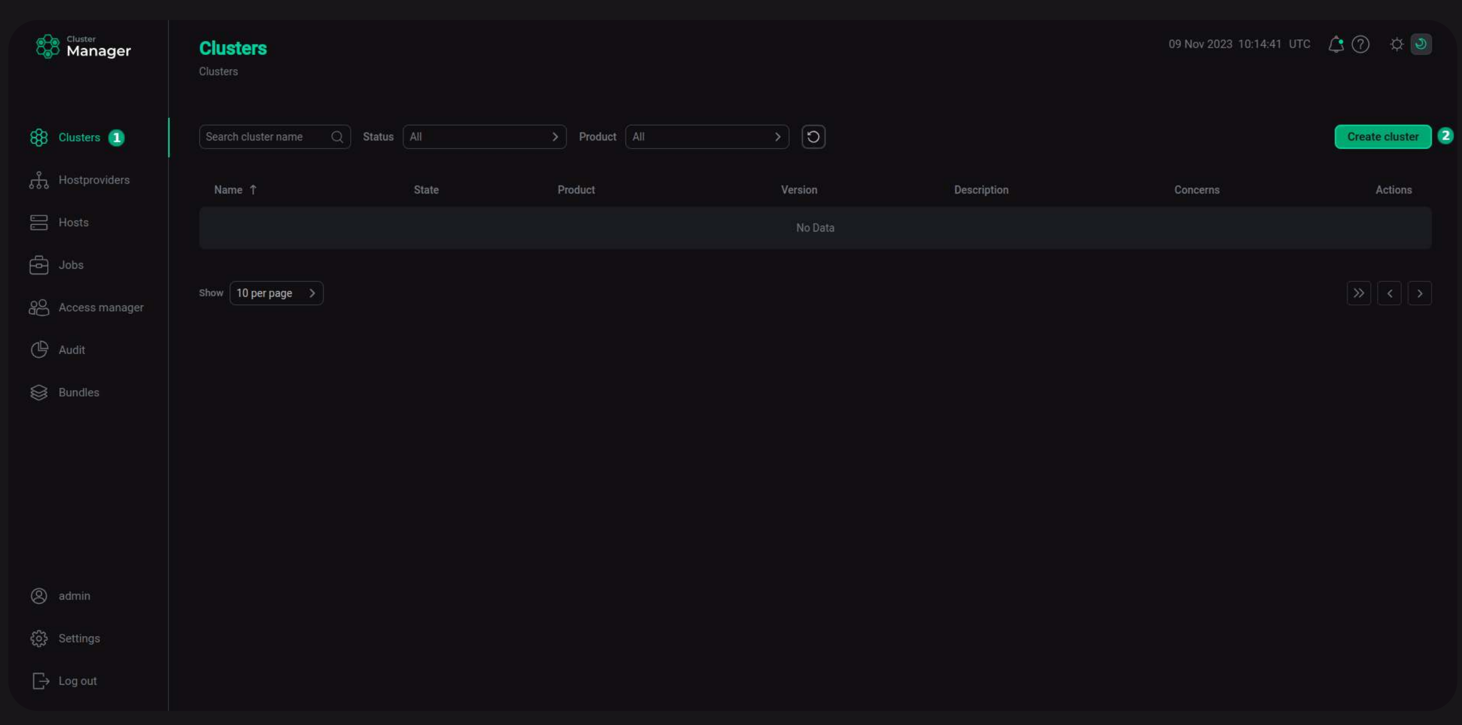
Step 3. Create a cluster based on the uploaded bundle

IMPORTANT

When creating a cluster via the ADCM web interface, you generate a new ADCM cluster instance. It means only adding the cluster to ADCM — at this stage no services are deployed.

The steps for cluster creation are given below:

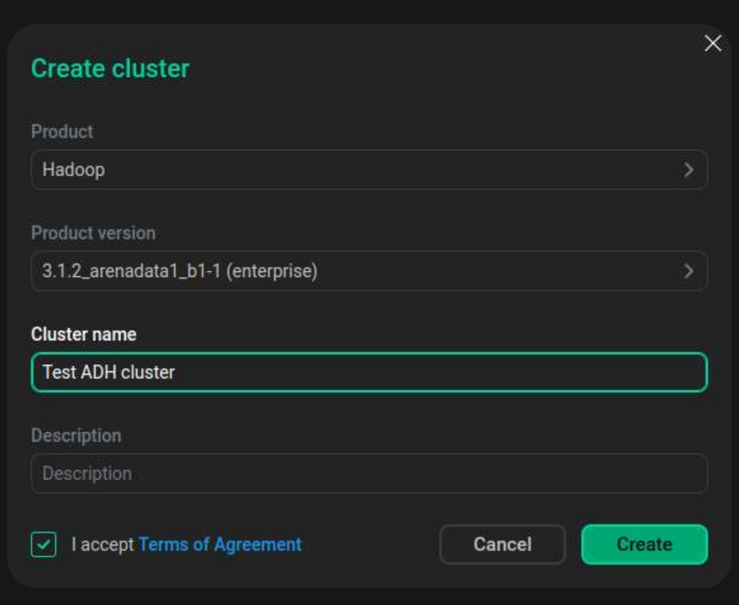
1. Select the ***Clusters*** item in the left navigation menu and click ***Create cluster***.



Manager
Clusters
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
No data
Show
10 per page

Switch to cluster creation

2. In the opened dialog, describe a new cluster instance.
   - Select an uploaded cluster bundle in the ***Product*** field.
   - Select a bundle version in the ***Product version*** field. Several versions become available in the case of different versions of the same bundle being uploaded.
   - Enter a cluster name in the ***Cluster name*** field.
   - Enter a cluster description in the ***Description*** field if necessary.
   - Set the ***I accept Terms of Agreement*** flag to sign the license agreement. This flag is displayed only for Enterprise bundles. To read the agreement text, refer to the ***Terms of Agreement*** link. You can also sign the license agreement on the ***Bundles*** page by clicking the desired product bundle.
   - Click ***Create***.



Create cluster

Product: Hadoop

Product version: 3.1.2_arenadata1_b1-1 (enterprise)

Cluster name: Test ADH cluster

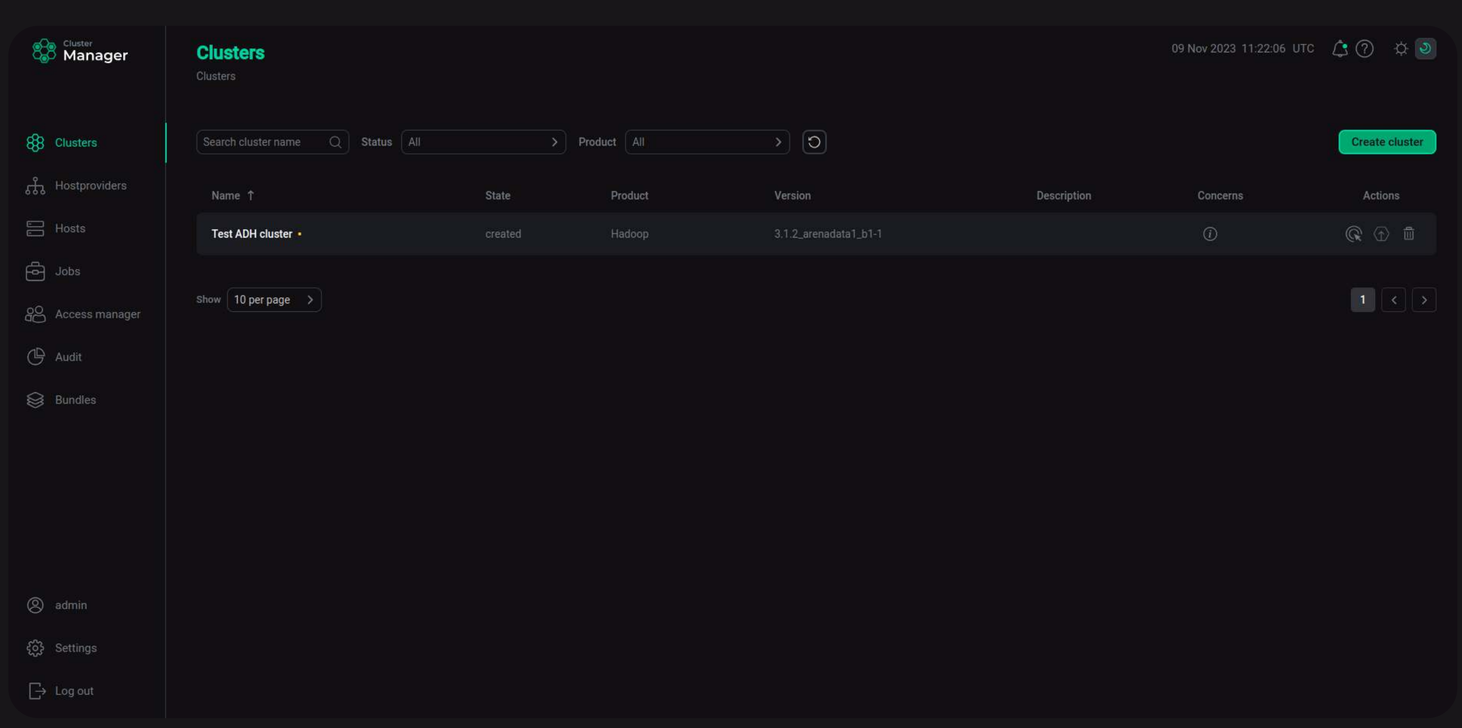
Description: Description

☑ I accept Terms of Agreement

Cancel
Create

Create a cluster

3. As a result, the created cluster instance is displayed on the ***Clusters*** page.



Manager
Clusters
Clusters
09 Nov 2023 11:22:06 UTC
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

The result of successful cluster creation

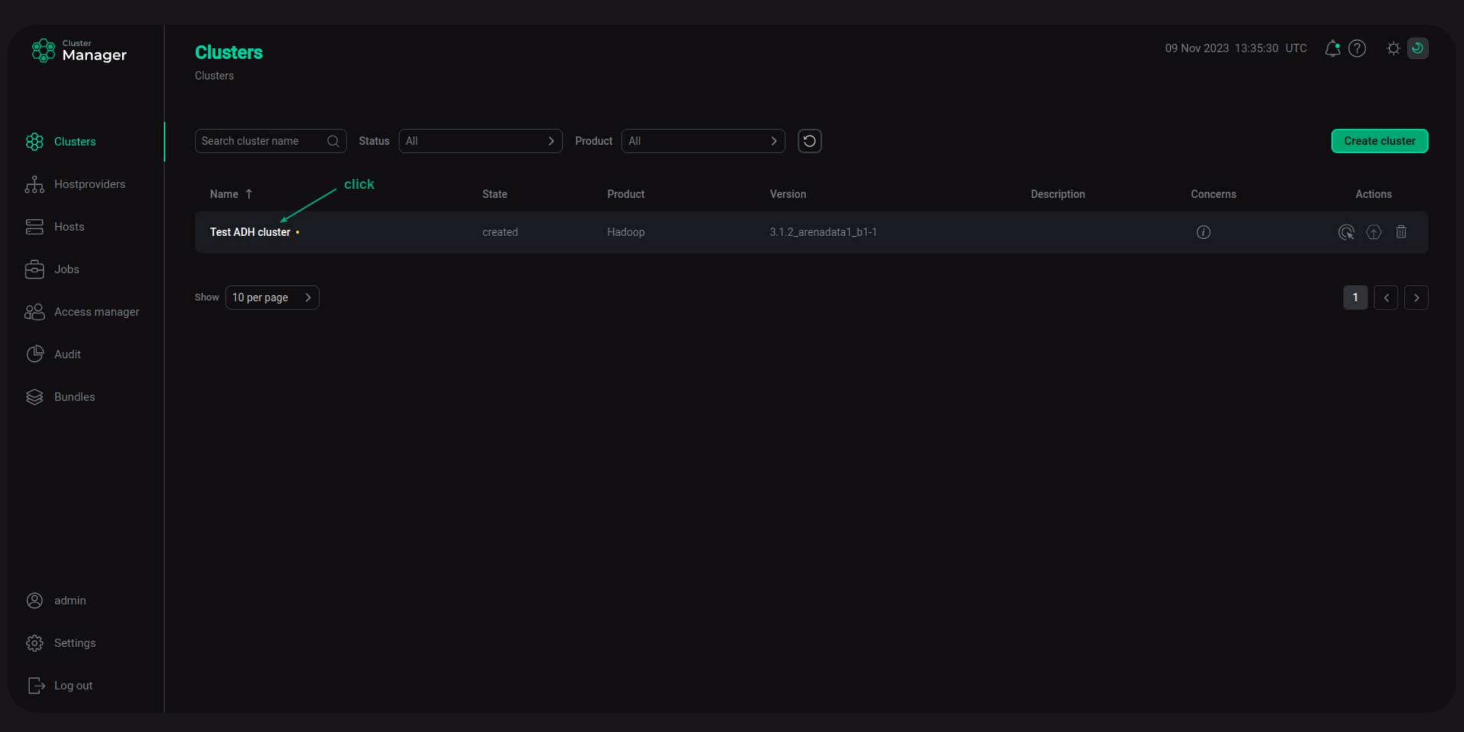
Add services

Eugenia Kuzina, Daria Barysheva

To Table of Contents

In ADCM a **service** means a software that performs some function. Examples of services for ADH clusters: HDFS, HBase, Hive, etc. The steps for adding services to a cluster are listed below:

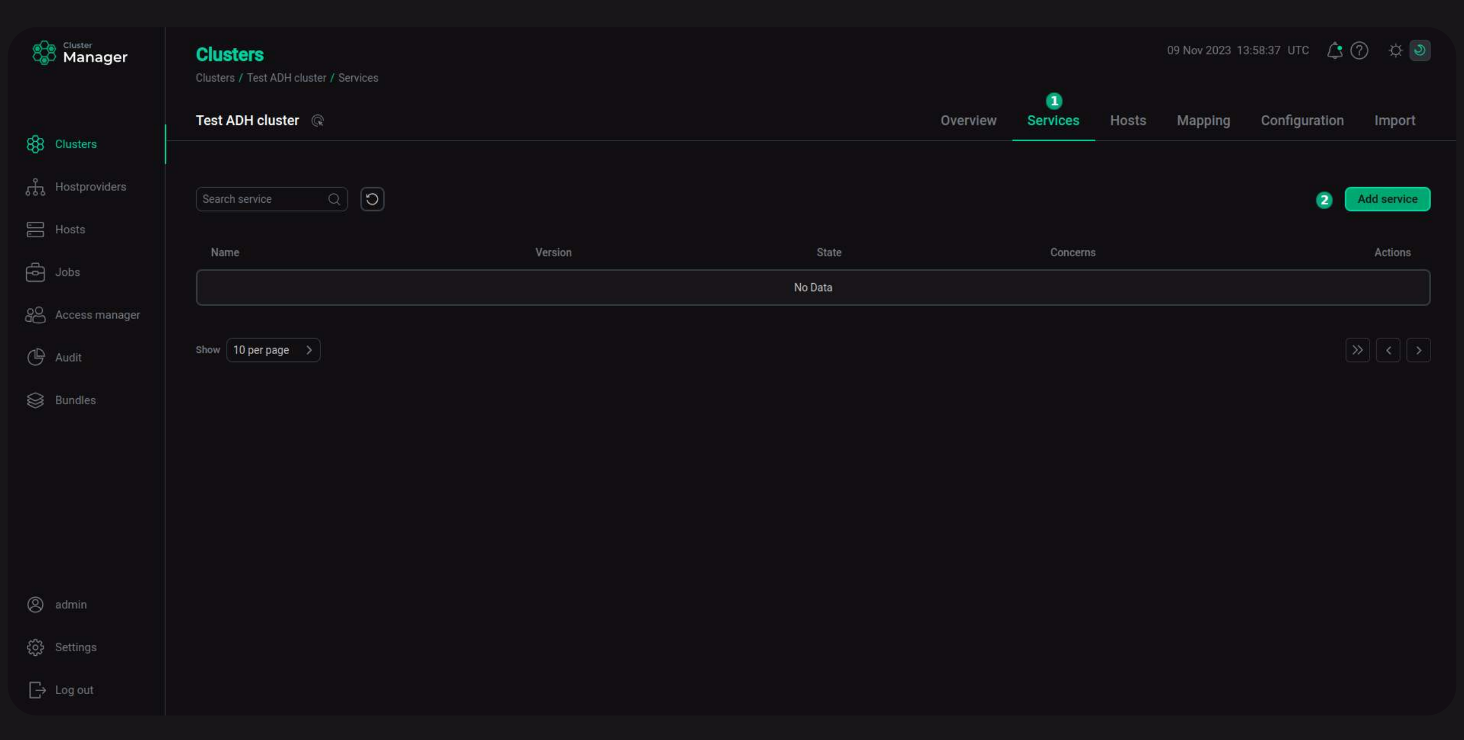
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

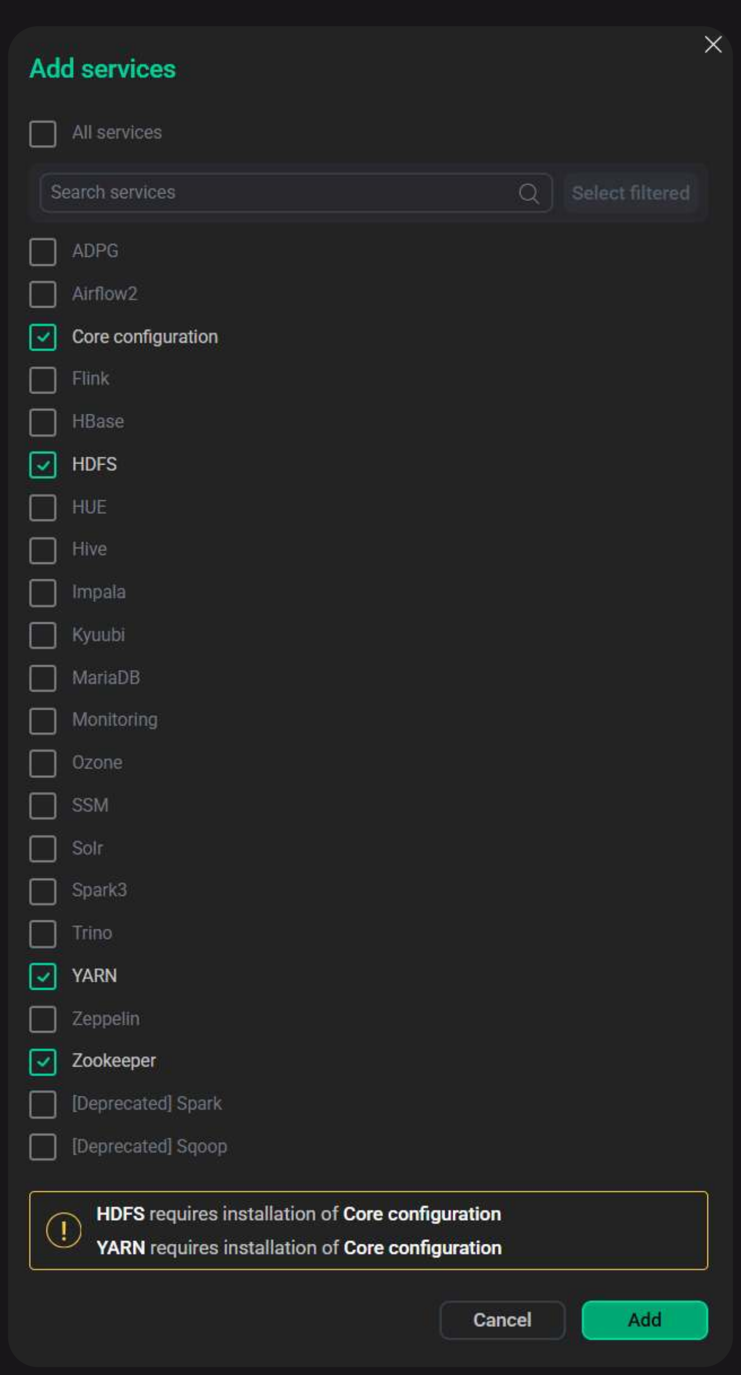
Select a cluster

2. Open the ***Services*** tab on the cluster page and click ***Add service***.


Cluster Manager
Clusters
Clusters / Test ADH cluster / Services
09 Nov 2023 13:58:37 UTC
Test ADH cluster
1
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Switch to adding services

3. In the opened dialog, select services that should be added to the cluster and click ***Add***.


Add services
Search services
Select filtered
Cancel
Add

☐ All services
☐ ADPG
☐ Airflow2
☑ Core configuration
☐ Flink
☐ HBase
☑ HDFS
☐ HUE
☐ Hive
☐ Impala
☐ Kyuubi
☐ MariaDB
☐ Monitoring
☐ Ozone
☐ SSM
☐ Solr
☐ Spark3
☐ Trino
☑ YARN
☐ Zeppelin
☑ Zookeeper
☐ [Deprecated] Spark
☐ [Deprecated] Sqoop

HDFS requires installation of **Core configuration**
YARN requires installation of **Core configuration**

Select services

The brief description of available services is listed below.

Services that can be added to the ADH cluster

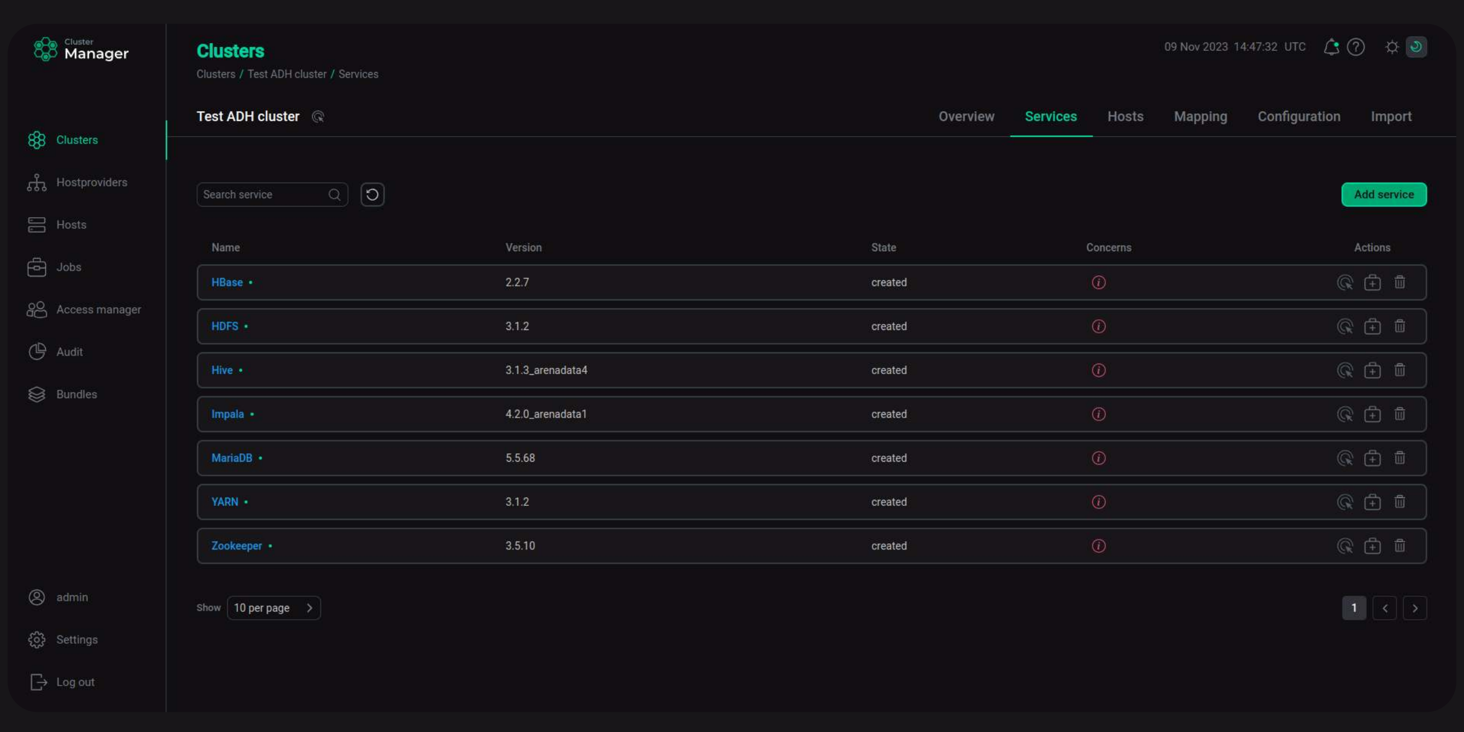
Service	Purpose
Airflow2	In comparison with the Airflow 1.x version, it offers the following features: High Availability, lowered task latency, full REST API, TaskFlow API, task groups, independent providers, and others
Core configuration	A service that contains common Hadoop packages required by every other service except ADPG, Solr, Zookeeper, and Monitoring. This service is installed on all nodes. If an already installed Hadoop service is expanded to another host, the Core configuration service is automatically installed on the host. The service is supported since ADH version 3.3.6.2.b1. Configuration parameters of this service are described in the Configuration parameters article. After applying any changes to this service configuration, you need to run the ***Update Core configuration*** action for this service
Flink	A distributed platform used in high-load Big Data applications for analyzing data stored in Hyperwave clusters. It can be used in different streaming use cases: event-driven applications, stream and batch analytics, data pipelines and ETL, etc.
Flink2	Flink-based service with version 2.x. This service is in the technical preview mode, may have a number of limitations, and is not recommended for production use
HBase	A non-relational, distributed database written in Java and used on the top of HDFS. Belongs to the class of column-oriented key-value storages. It is useful for random, real-time read/write access to Big Data
HDFS	A distributed file system used in Hyperwave clusters for storing large files. Provides the possibility of the streaming access to the information distributed block-by-block across cluster nodes
Hive	A software designed for building data warehouses (DWH) and analyzing Big Data. It runs on the top of HDFS and other compatible systems, such as Apache HBase. It facilitates writing, reading, and managing large datasets stored in distributed systems
HUE	Apache HUE is a UI-based SQL assistant that provides interpreters for various data sources, like Impala, Hive, Kyuubi, and many more. It has a handy query editor with intelligent autocomplete, drag and drop, quick browse, and other features
Impala	Impala provides fast, interactive SQL queries on data stored in HDFS, HBase, or S3. In addition to the unified storage platform, Impala also uses the same metadata, SQL syntax (Hive SQL), and JDBC driver as Apache Hive. It makes Impala a unified platform for real-time or batch-oriented queries
Kyuubi	Apache Kyuubi is a distributed and multi-tenant gateway to provide SQL on Data Warehouses and Lakehouses. It builds distributed SQL query engines on top of various kinds of modern computing frameworks, e.g. Apache Spark, Flink, Hive, Impala, etc., to query massive datasets distributed over fleets of machines from heterogeneous data sources
Monitoring	A service that should be added if monitoring of the ADH cluster is planned
Ozone	Apache Ozone (O3) is a distributed object storage that is optimized for working with both Hadoop services and S3 storages. It’s highly scalable, can share the same cluster and security policies with HDFS, and is capable of effectively handling files no matter the size
Solr	A search platform based on the Apache Lucene project. Its main features include full-text search, faceted search, highlighting search results, distributed indexing, integration with databases, processing documents with a complex format (Word, PDF, etc.), load-balanced querying, centralized configuration, and others
Spark3	A fast analytics engine used for large-scale data processing and is compatible with Hadoop data. It can run in Hyperwave clusters using YARN or Spark’s standalone mode. It can process data in HDFS, HBase, Cassandra, Hive, and other Hadoop input formats. It supports both batch processing and new workloads like streaming, machine learning, interactive queries, etc. Spark 3.x offers such new features as adaptive execution of Spark SQL, dynamic partition pruning (DPP), graph processing, enhanced support for deep learning, and others
Spark4	Spark-based service with version 4.x. This service is in the technical preview mode, may have a number of limitations, and is not recommended for production use
SSM	Smart Storage Manager is a service that aims to optimize the efficiency of storing and managing data in the Hadoop Distributed File System. SSM collects HDFS operation data and system state information, and based on the collected metrics can automatically use methodologies such as cache, storage policies, heterogeneous storage management (HSM), data compression, and Erasure Coding. In addition, SSM provides the ability to configure asynchronous replication of data and namespaces to a backup cluster for the purpose of organizing DR
Trino	Trino is an open-source SQL query engine used for processing data in parallel, distributed over multiple storages, such as object storages, databases, and file systems
YARN	A service needed for managing cluster resources and scheduling/monitoring jobs. Uses a special daemon (ResourceManager) that abstracts all the computing resources of the cluster and manages their provision to distributed applications
Zeppelin	A service that plays a role of a web-based notebook and enables interactive data analytics. Allows to create queries to data in Hyperwave clusters and display the results in the form of tables, graphs, charts, etc.
Zookeeper	A centralized coordination service for distributed applications. It is used in Hyperwave clusters for failure detection, active NameNode election, health monitoring, session management, etc.

The minimal set of services recommended for ADH clusters is described below:

- Core configuration;
- HDFS/Ozone;
- YARN;
- Zookeeper (optional for the Community Edition of ADH).

These services make up the core of Hyperwave and are sufficient to organize distributed data storage and processing. The full list of services depends on the requirements of a particular project.

4. As a result, the added services are displayed at the *Services* tab.


Cluster Manager
Clusters
Clusters / Test ADH cluster / Services
09 Nov 2023 14:47:32 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
HBase	2.2.7	created		
HDFS	3.1.2	created		
Hive	3.1.3_arenadata4	created		
Impala	4.2.0_arenadata1	created		
MariaDB	5.5.68	created		
YARN	3.1.2	created		
Zookeeper	3.5.10	created		

The result of successful adding services to a cluster

NOTE

You can also add services later. The process of adding new services to already running cluster does not differ from installing a service from scratch.

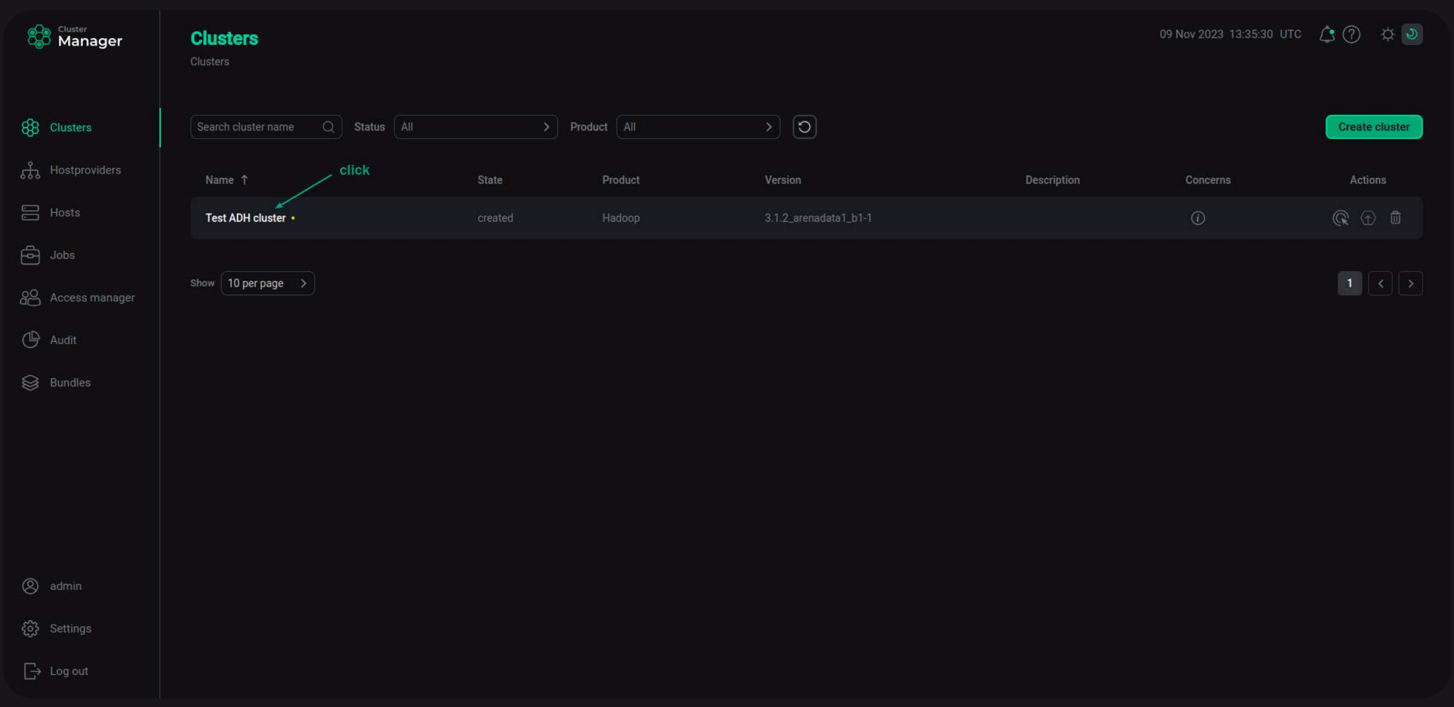
Add hosts to a cluster

Eugenia Kuzina, Daria Barysheva

To Table of Contents

The steps for adding hosts to a cluster are listed below:

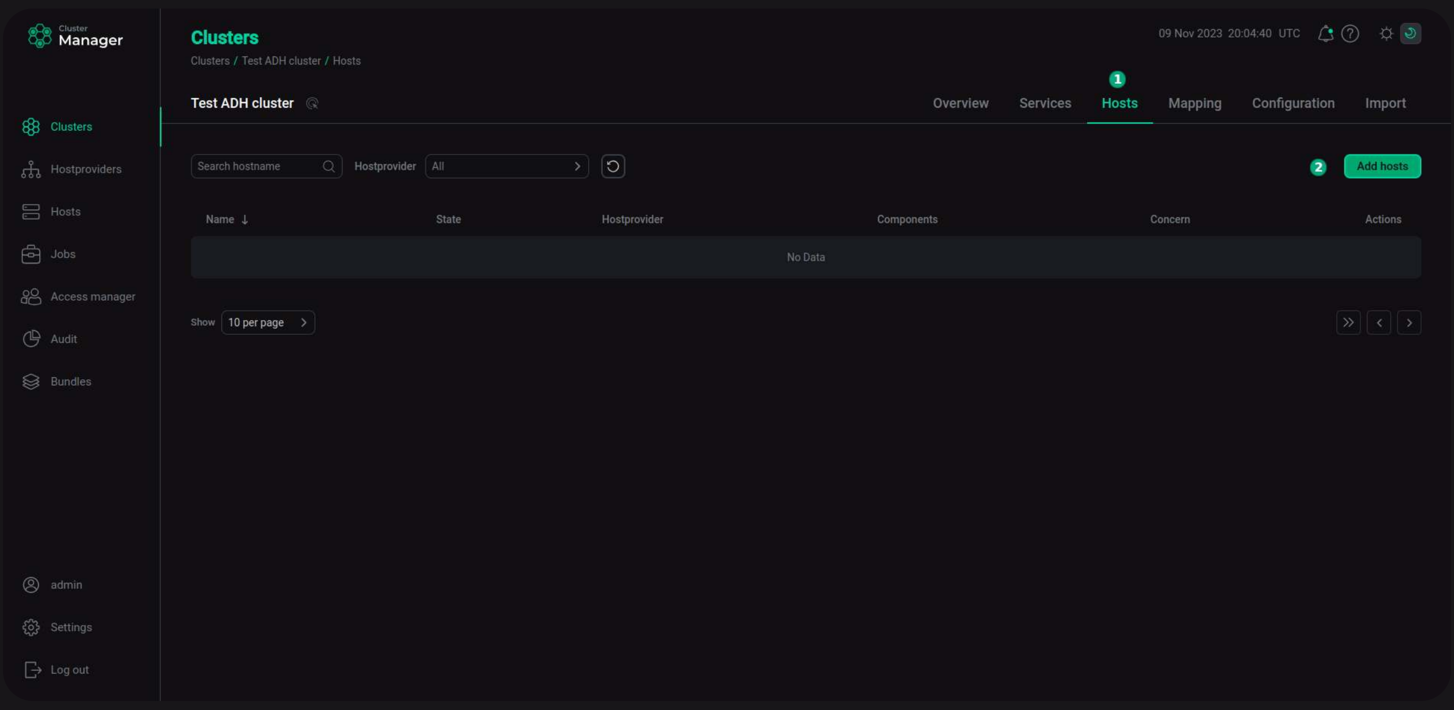
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

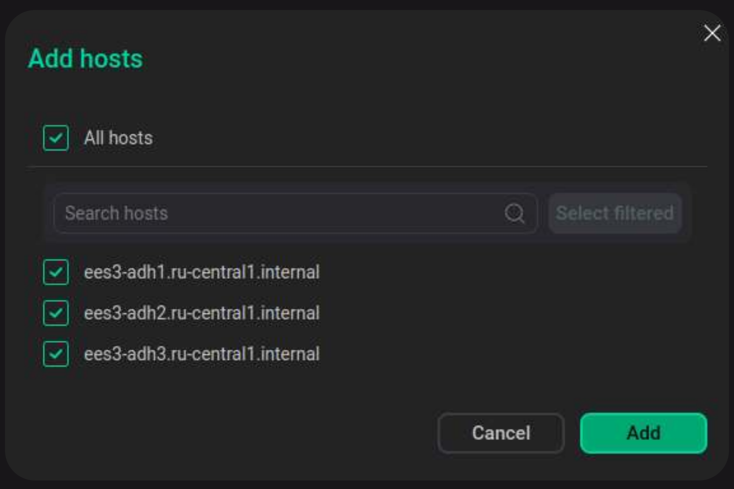
Select a cluster

2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Hosts
09 Nov 2023 20:04:40 UTC
Test ADH cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
2
Add hosts
Name
State
Hostprovider
Components
Concern
Actions
No Data
Show
10 per page

Switch to adding hosts

3. In the opened dialog, select hosts that should be added to the cluster and click ***Add***.


Add hosts
Search hosts
Select filtered
Cancel
Add

☑ All hosts

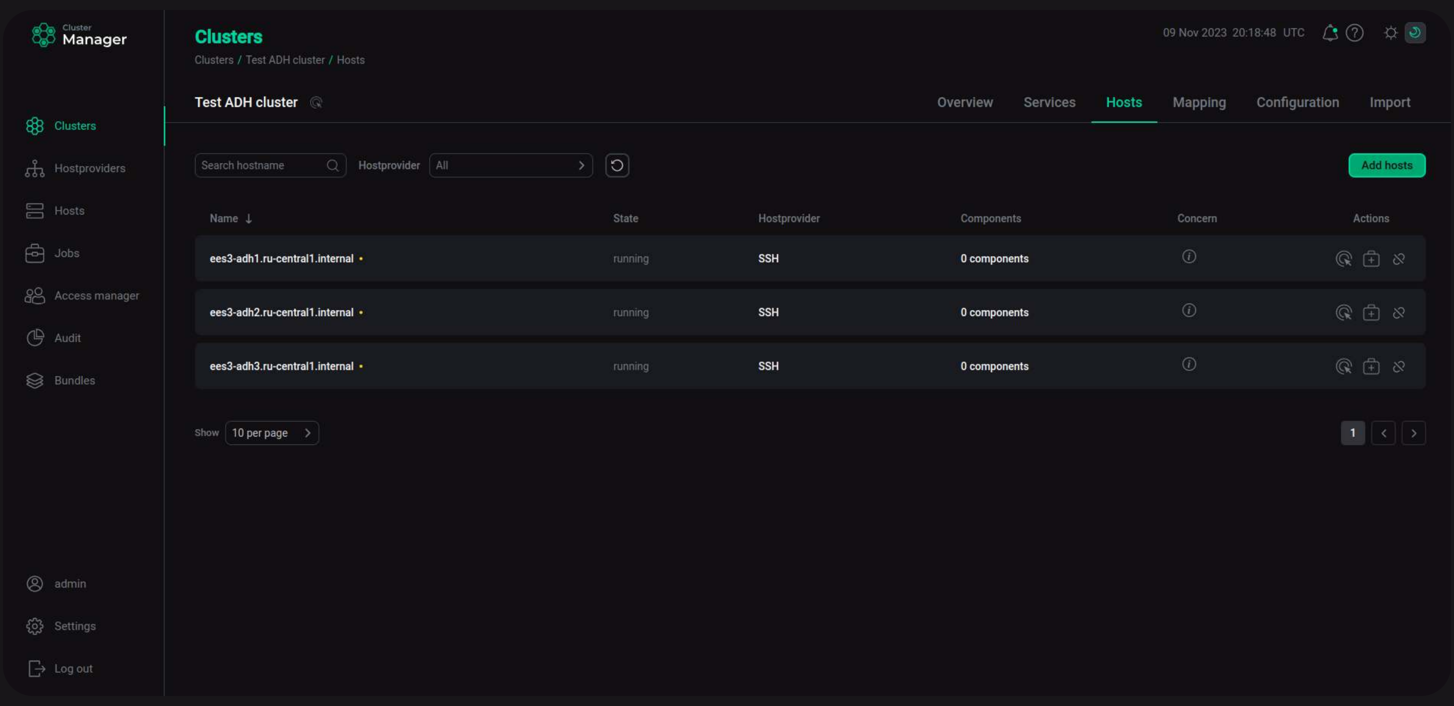
☑ ees3-adh1.ru-central1.internal

☑ ees3-adh2.ru-central1.internal

☑ ees3-adh3.ru-central1.internal

Select hosts

4. As a result, the added hosts are displayed on the ***Hosts*** tab.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Hosts
09 Nov 2023 20:18:48 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Add hosts
Show
10 per page
1

Name	State	Hostprovider	Components	Concern	Actions
ees3-adh1.ru-central1.internal	running	SSH	0 components		
ees3-adh2.ru-central1.internal	running	SSH	0 components		
ees3-adh3.ru-central1.internal	running	SSH	0 components		

The result of successful adding hosts to a cluster

Add components

Eugenia Kuzina, Daria Barysheva

To Table of Contents

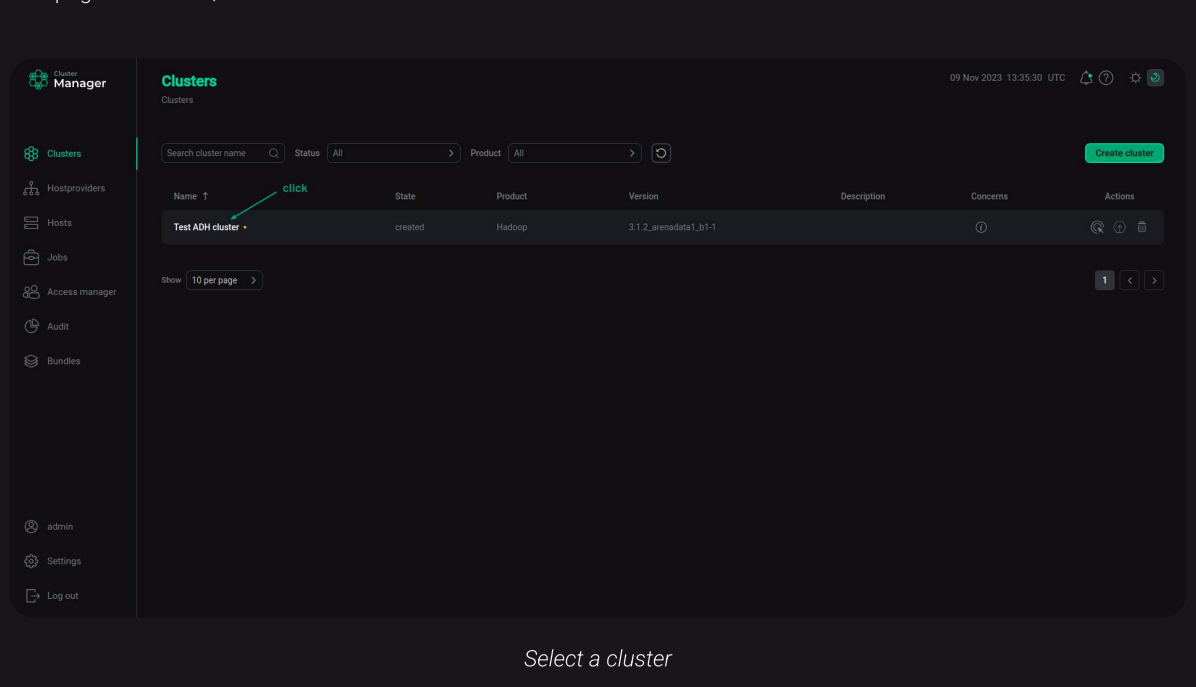
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts. For example, HDFS service can consist of such components as DataNode, NameNode, Client, and others.

IMPORTANT

Initially, there are no components on any of cluster hosts. You should map components to hosts manually.

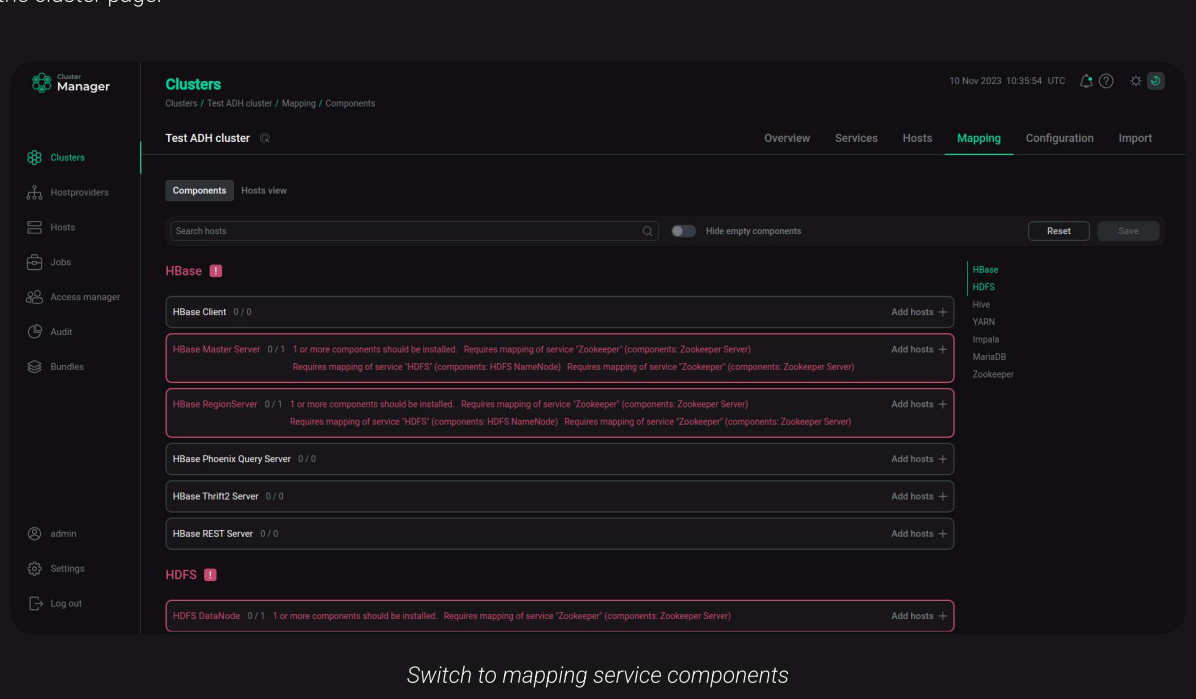
The steps for adding components to hosts are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.



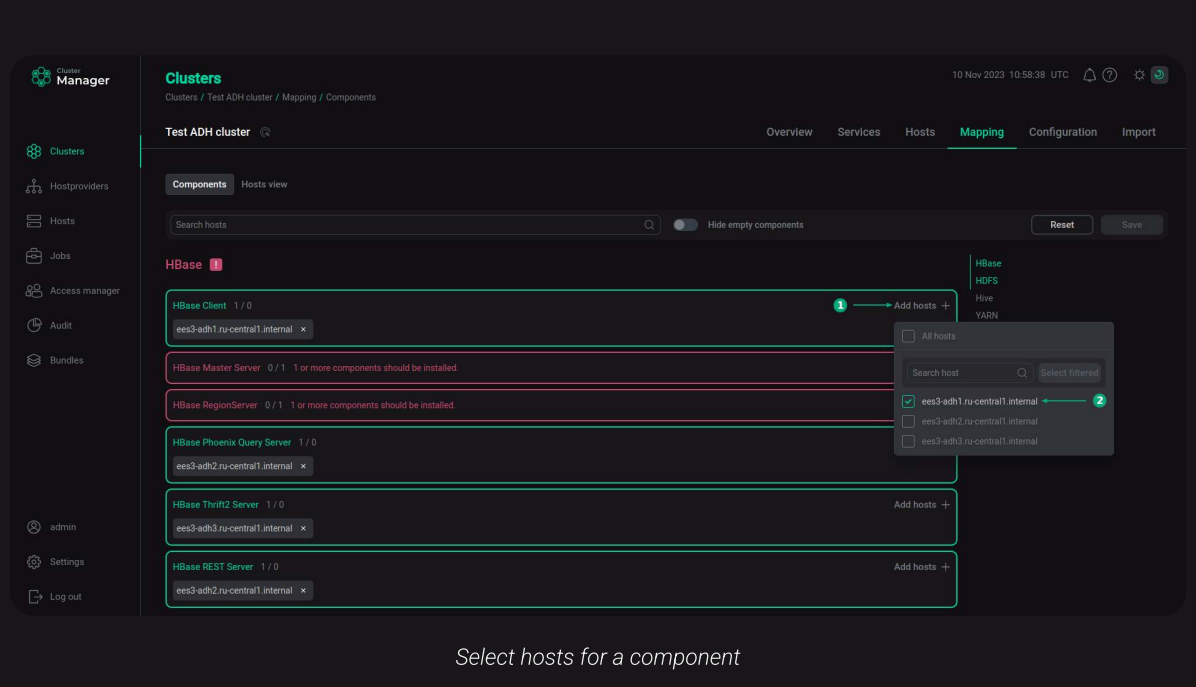
Select a cluster

2. Open the ***Mapping*** tab on the cluster page.



Switch to mapping service components

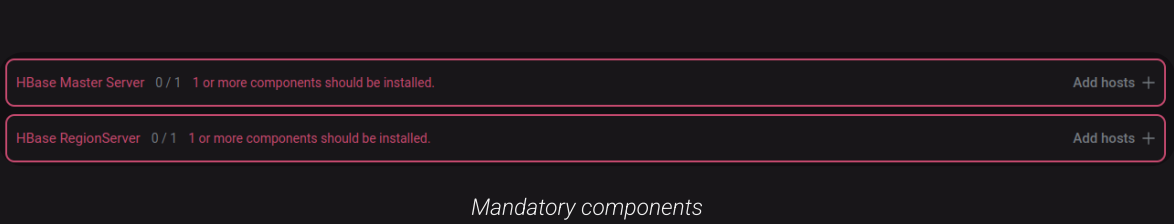
3. In the window that opens, map components to hosts. To assign hosts for each component, click ***Add hosts*** and select hosts in the pop-up window.



Select hosts for a component

Mandatory components

Each service can have mandatory and optional components. Mandatory components are highlighted in red. The ADCM UI also displays how many mandatory components should be installed.


HBase Master Server 0 / 1 1 or more components should be installed.
HBase RegionServer 0 / 1 1 or more components should be installed.

Mandatory components

CAUTION

Without assigning a necessary number of hosts to mandatory components, the component mapping cannot be saved.

The examples of distribution components in ADH clusters consisting of 3, 6, and 9 hosts are described in the table below. Examples are based on the Enterprise Edition of ADH.

Examples of components distribution

3 hosts

Host	Service	Component
Host 1	Core configuration	Configuration server
	Flink	Flink Client Flink JobManager Flink TaskManager Flink SQL-Gateway
	HBase	HBase Client HBase RegionServer
	HDFS	HDFS Client HDFS DataNode HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Client Hive Metastore Hive Tez
	Impala	Impala Daemon Impala Statestore Impala Catalog Service
	Monitoring	Grafana Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Recon Ozone HttpFS Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client Spark3 History server
	Trino	Trino Coordinator Trino Client
	YARN	MapReduce History server YARN Client YARN NodeManager
	ZooKeeper	ZooKeeper Server
Host 2	Airflow2	Airflow2 Broker Airflow2 Flower Airflow2 Scheduler Airflow2 Server Airflow2 Worker
	Flink	Flink Client Flink History Server
	HBase	HBase Client HBase Master server HBase Phoenix query server HBase RegionServer
	HDFS	HDFS Client HDFS DataNode HDFS HttpFS server HDFS JournalNode
	Hive	Hive Client Hive HiveServer2 Hive Tez Hive TezUI
	Impala	Impala Daemon
	Monitoring	Prometheus Server Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Datanode
	Solr	Solr Server Solr prometheus exporter
	Spark3	Spark3 Client Spark3 Livy server
	Trino	Trino Worker Trino Client
	YARN	YARN Client YARN NodeManager YARN ResourceManager
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 3	Arenadata PostgreSQL	Arenadata PostgreSQL
	Flink	Flink Client
	HBase	HBase Client HBase RegionServer HBase Thrift2 server HBase REST Server
	HDFS	HDFS Client HDFS DataNode HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Client Hive Tez
	HUE	HUE Server
	Impala	Impala Daemon Impala Client
	Kyuubi	Kyuubi Server
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone S3G Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client
	SSM	SSM Server SSM Agent
	Trino	Trino Worker Trino Client
	YARN	YARN Client YARN NodeManager YARN Timeline server
	ZooKeeper	ZooKeeper Server

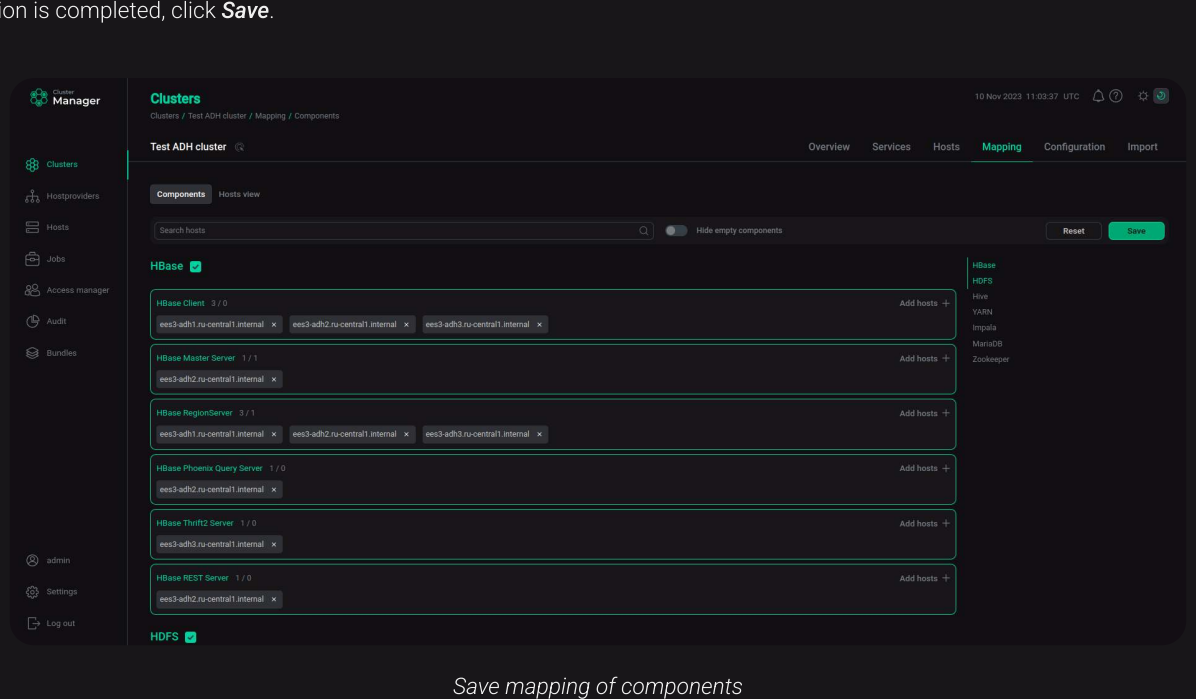
6 hosts

Host	Service	Component
Host 1	Core configuration	Configuration server
	Flink	Flink JobManager Flink TaskManager Flink SQL-Gateway
	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Metastore
	Impala	Impala Statestore Impala Catalog Service
	Monitoring	Grafana Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Recon Ozone HttpFS
	Solr	Solr Server
	Spark3	Spark3 History server
	Trino	Trino Coordinator Trino Client
	YARN	MapReduce History server
	ZooKeeper	ZooKeeper Server
Host 2	Airflow2	Airflow2 Broker Airflow2 Flower Airflow2 Scheduler Airflow2 Server Airflow2 Worker
	HBase	HBase Master server HBase Phoenix query server
	HDFS	HDFS JournalNode HDFS HttpFS server
	Hive	Hive HiveServer2 Hive TezUI
	Monitoring	Prometheus Server Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server Solr prometheus exporter
	Spark3	Spark3 Livy server
	Trino	Trino Worker Trino Client
	YARN	YARN ResourceManager
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 3	Arenadata PostgreSQL	Arenadata PostgreSQL
	HBase	HBase Thrift2 server HBase REST Server
	HDFS	HDFS NameNode HDFS ZKFC HDFS JournalNode
	HUE	HUE Server
	Kyuubi	Kyuubi Server
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	SSM	SSM Server SSM Agent
	YARN	YARN Timeline server
	ZooKeeper	ZooKeeper Server
Host 4	Flink	Flink Client Flink History Server
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon Impala Client
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 5	Flink	Flink Client
	HBase	HBase Client HBase RegionServer
	HDFS	HDFS DataNode HDFS Client
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 6	Flink	Flink Client
	HBase	HBase Client HBase RegionServer
	HDFS	HDFS DataNode HDFS Client
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone S3G
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server

9 hosts

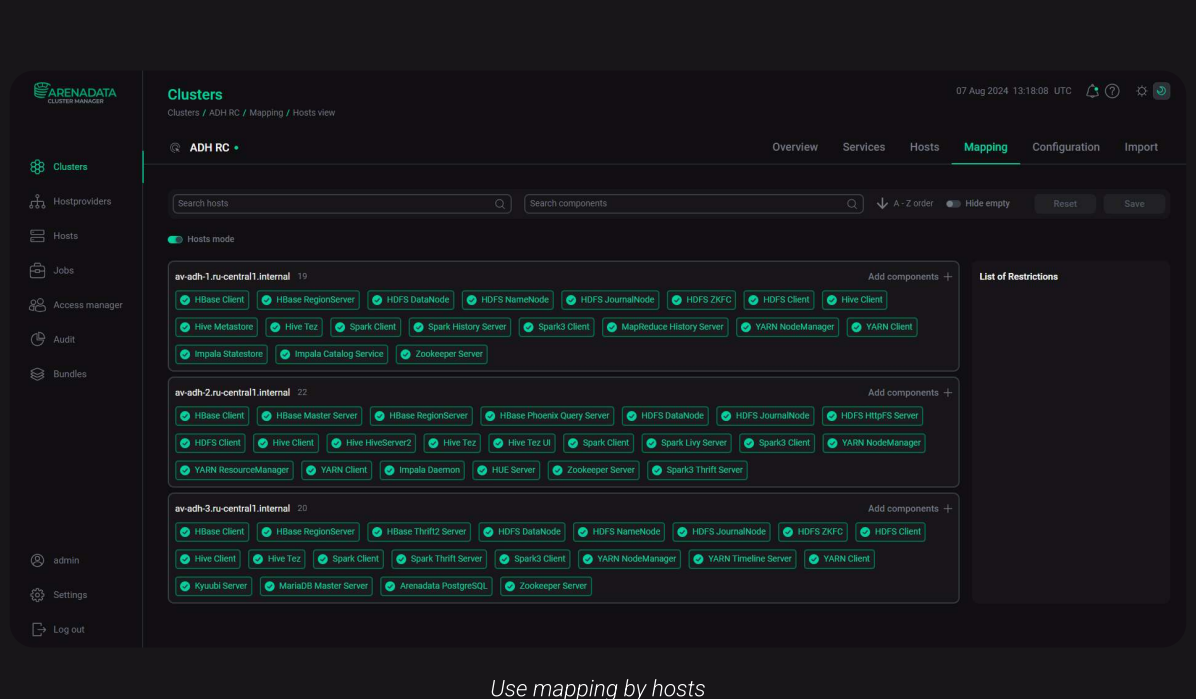
Host	Service	Component
Host 1	Core configuration	Configuration server
	Flink	Flink JobManager Flink TaskManager Flink SQL-Gateway
	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Metastore
	Impala	Impala Statestore Impala Catalog Service
	Monitoring	Grafana Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Recon Ozone HttpFS
	Solr	Solr Server
	Spark3	Spark3 History server
	Trino	Trino Coordinator Trino Client
	YARN	MapReduce History server
	ZooKeeper	ZooKeeper Server
Host 2	Airflow2	Airflow2 Broker Airflow2 Flower Airflow2 Scheduler Airflow2 Server Airflow2 Worker
	Flink	Flink TaskManager
	HBase	HBase Master server HBase Phoenix query server
	HDFS	HDFS JournalNode HDFS HttpFS server
	Hive	Hive HiveServer2 Hive TezUI
	Monitoring	Prometheus Server Node Exporter
	Solr	Solr Server Solr prometheus exporter
	Spark3	Spark3 Livy server
	YARN	YARN ResourceManager
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 3	HBase	HBase Thrift2 server HBase REST Server
	HDFS	HDFS NameNode HDFS ZKFC HDFS JournalNode
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Datanode
	Solr	Solr Server
	Trino	Trino Worker Trino Client
	YARN	YARN Timeline server
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 4	Arenadata PostgreSQL	Arenadata PostgreSQL
	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC HDFS HttpFS server
	HBase	HBase Master server
	HUE	HUE Server
	Kyuubi	Kyuubi Server
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	SSM	SSM Server SSM Agent
	Trino	Trino Worker Trino Client
	ZooKeeper	ZooKeeper Server
Host 5	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC
	HBase	HBase Master server
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone S3G
	Solr	Solr Server
	Trino	Trino Worker Trino Client
	ZooKeeper	ZooKeeper Server
Host 6	Flink	Flink Client Flink History Server
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon Impala Client
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 7	Flink	Flink Client
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 8	Flink	Flink Client
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 9	Flink	Flink Client
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server

4. After component distribution is completed, click ***Save***.



Save mapping of components

Starting with ADCM 2.2.0, you can use another mapping mode — by hosts. To do this, switch on the ***Hosts mode*** toggle and map components to hosts by clicking ***Add components*** for each host sequentially.



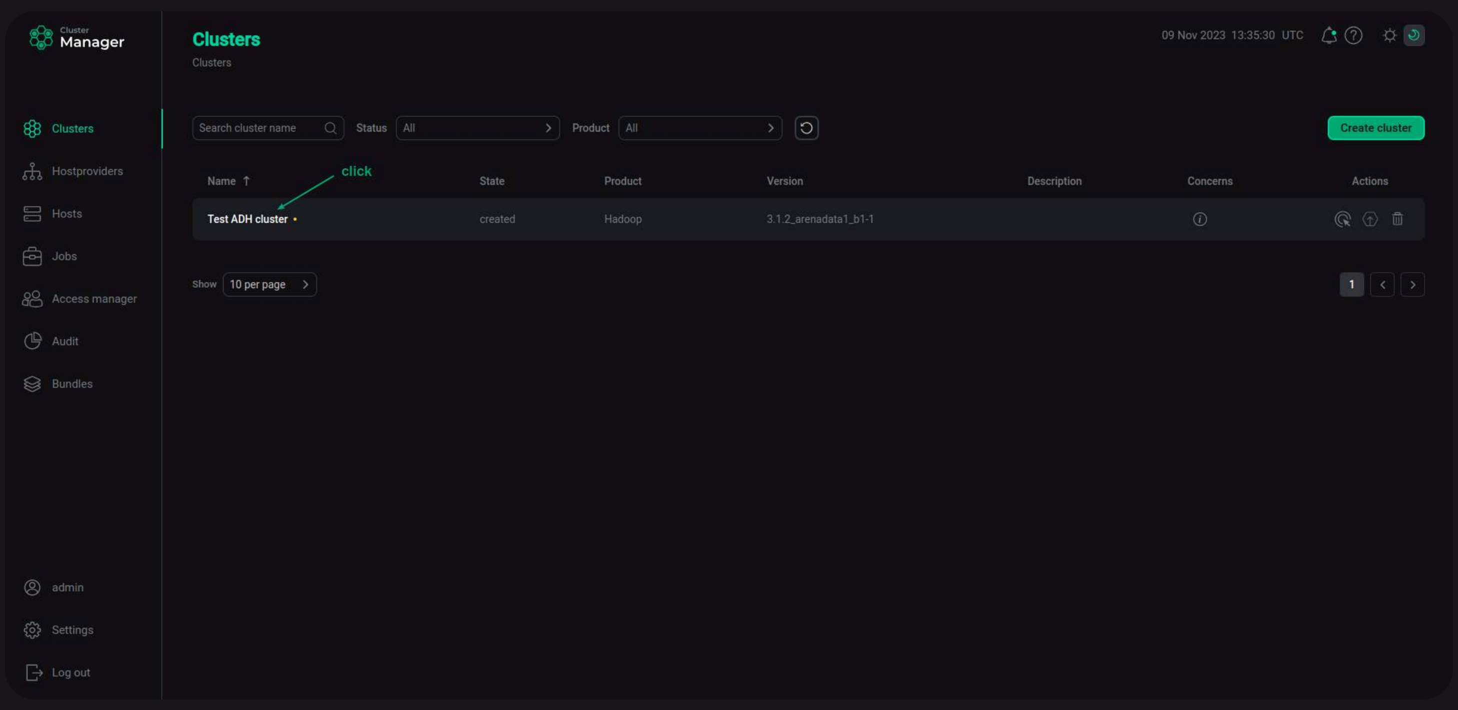
Use mapping by hosts

Configure services

Eugenia Kuzina, Daria Barysheva

To Table of Contents

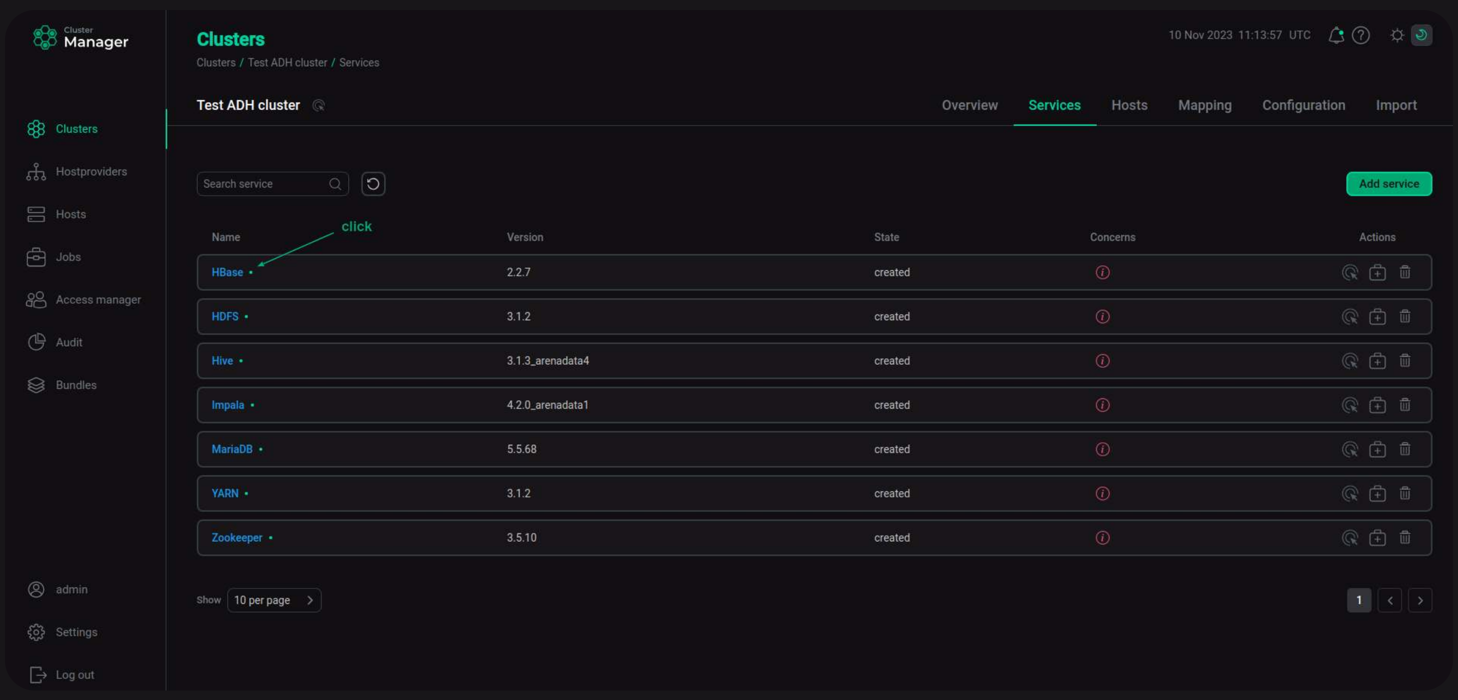
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name ↑	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

Select a cluster

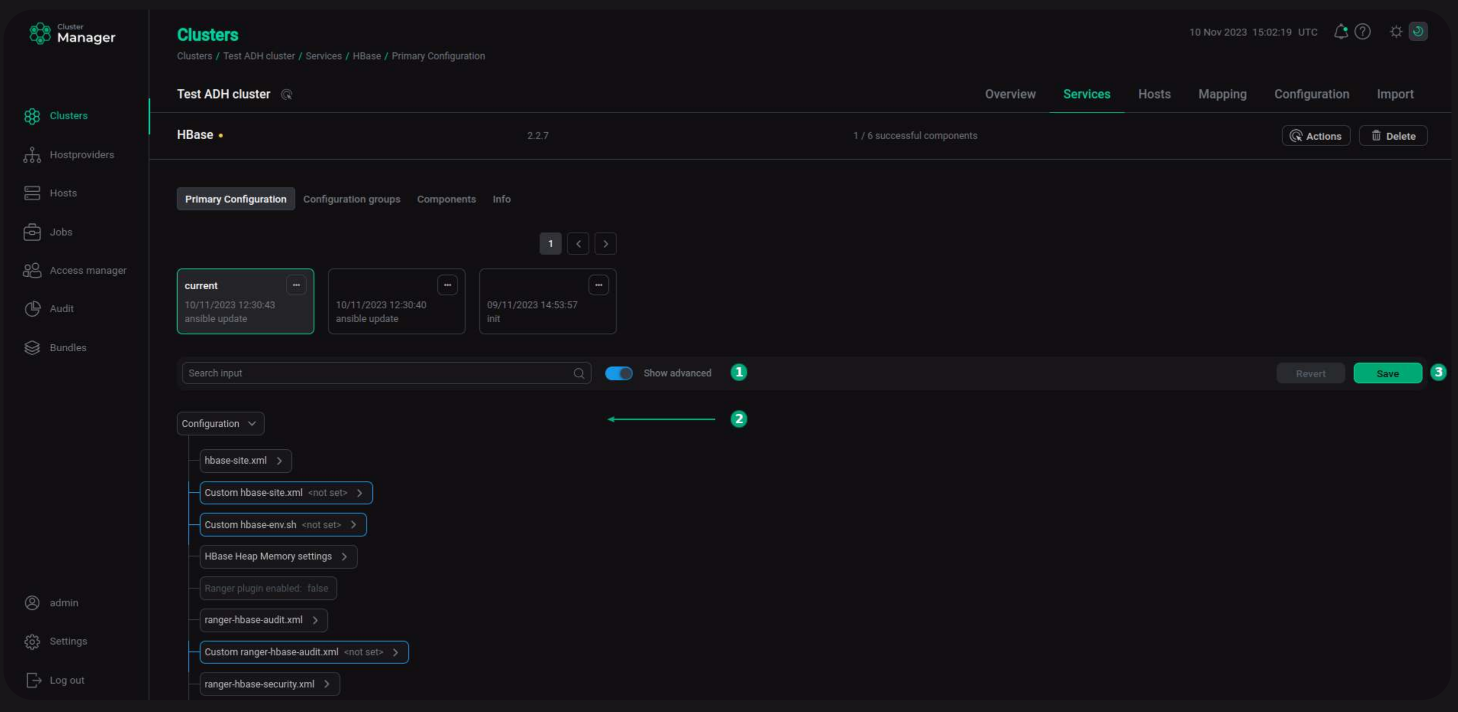
2. Open the ***Services*** tab on the cluster page, find the row that contains the desired service, and click its name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Services
10 Nov 2023 11:13:57 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
click
Show
10 per page
1

Name	Version	State	Concerns	Actions
HBase	2.2.7	created		
HDFS	3.1.2	created		
Hive	3.1.3_arenadata4	created		
Impala	4.2.0_arenadata1	created		
MariaDB	5.5.68	created		
YARN	3.1.2	created		
Zookeeper	3.5.10	created		

Switch to configuring a service

3. In the window that opens:
   - Open the ***Primary Configuration*** tab.
   - Switch on the ***Show advanced*** toggle.
   - Fill in configuration parameters of the selected service. The fields highlighted in red are mandatory.
   - Click ***Save***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Services / HBase / Primary Configuration
10 Nov 2023 15:02:19 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
HBase
2.2.7
1 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1
current
10/11/2023 12:30:43
ansible update
10/11/2023 12:30:40
ansible update
09/11/2023 14:53:57
init
Search input
Show advanced
1
Revert
Save
3
Configuration
2
hbase-site.xml
Custom hbase-site.xml <not set>
Custom hbase-env.sh <not set>
HBase Heap Memory settings
Ranger plugin enabled: false
ranger-hbase-audit.xml
Custom ranger-hbase-audit.xml <not set>
ranger-hbase-security.xml

Configure a service

NOTE

For more information about services configuration parameters, see Configuration parameters in the References.

Configure a cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

Primary configuration

Configuration groups

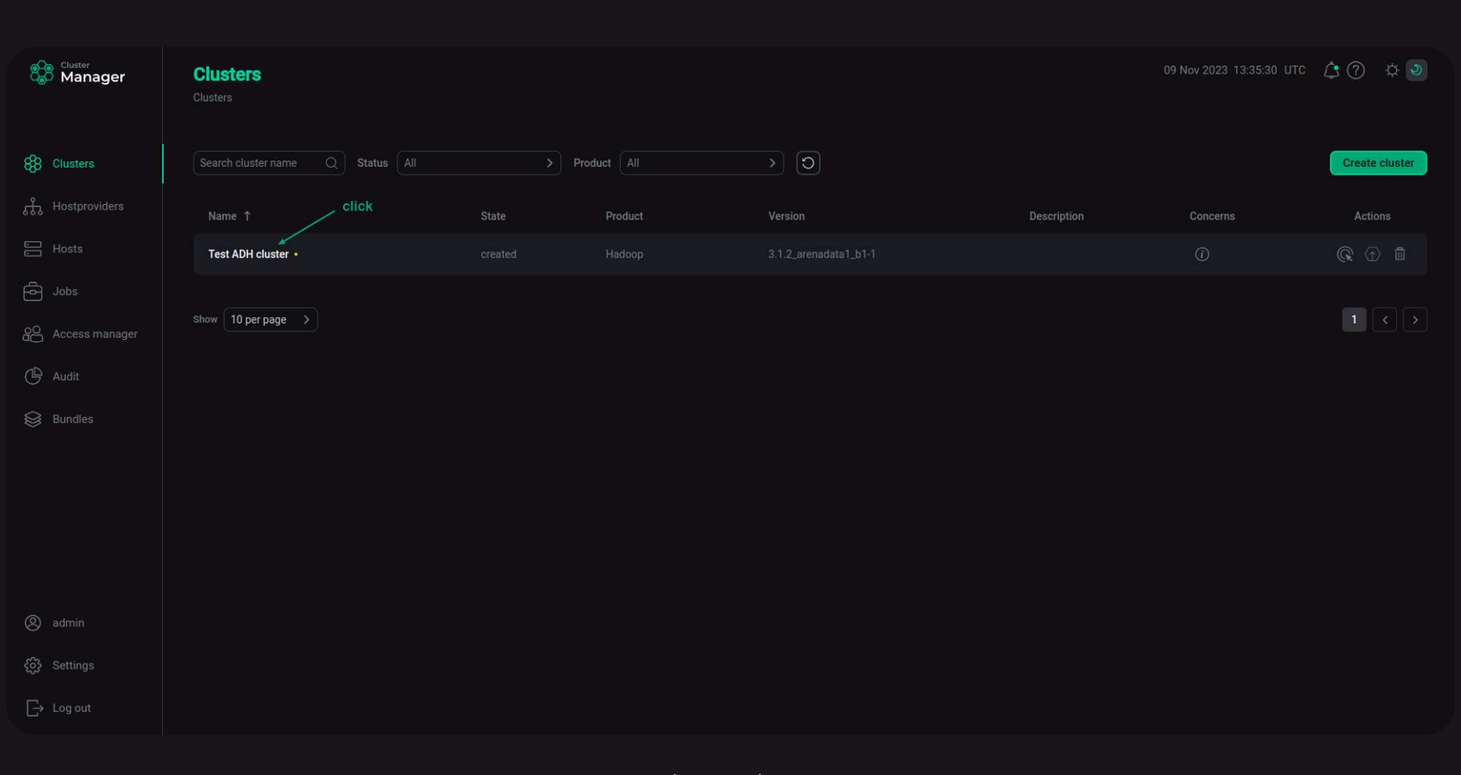
Ansible settings

IMPORTANT

There is usually no need to change cluster configuration parameters. You can leave all parameters at the default values.

After creation a new cluster, you can configure it by performing the following steps:

1. Select a cluster on the *Clusters* page. To do this, click a cluster name in the *Name* column.



Cluster Manager
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search cluster name
Status
All
Product
All
Create cluster
click
Name
State
Product
Version
Description
Concerns
Actions
Show
10 per page
1
admin
Settings
Log out

Test ADH cluster created Hadoop 3.1.2_arenadata1_b1-1

Select a cluster

2. Open the *Configuration* tab on the cluster page. The *Configuration* tab includes the following sections: *Primary configuration*, *Configuration groups*, *Ansible settings*.
3. Fill in all necessary parameters on the selected tab and click *Save*.

Primary configuration

The *Configuration* → *Primary configuration* tab allows you to configure the general cluster settings.

Available configuration parameters are listed below:

- The *Reliability control* section. It is displayed if the *Show advanced* switch is activated. The section contains the following parameters:
  - *retries* — the number of retries for cluster operations (20 by default);
  - *delay* — the delay between retries in seconds (10 by default);
  - *timeout* — timeout for checks of network ports in seconds (60 by default).
- Paths to the repositories that are used in the process of installing — depending on the cluster operating system (CentOS 7, Red Hat 7, Ubuntu 22, ALT Linux 8, Astra Linux, RedOS 7):
  - *monitoring* — monitoring components repository;
  - *arenadata* — Arenadata YUM repository;
  - *arenadata_postgres* — Arenadata PostgreSQL YUM repository;
  - *prometheus_repo* — Prometheus YUM repository;
  - *arenadata_enterprise* — Arenadata Enterprise YUM repository;
- The *Docker registry* parameter that specifies a Docker Registry address. The default value is *hub.adsw.io* — Docker Registry from the Arenadata repository.
- The *Kerberos* section that includes settings listed in the table below. Some options are visible only with the *Show advanced* option enabled.

Parameter	Description	Default value
Enable Kerberos	Enables the Kerberos authentication	False
Authentication on WEB UIs	Enables the Kerberos authentication on WEB UIs	False
SPNEGO Signature Value	A secret key that is used to generate and verify the signature in the HTTP authentication protocol SPNEGO (in the Base64 format)	—
Kerberos KDC type	A KDC type. For more information, see Kerberos	MIT
KDC hosts	One or more KDC hosts with running FreeIPA server(s)	—
Realm	A Kerberos realm to connect to the FreeIPA server	—
Domains	One or more domains associated with FreeIPA	—
Kadmin server	A host where `kadmin` is running	—
Kadmin principal	A principal name used to connect via `kadmin`, for example `admin@RU-CENTRAL1.INTERNAL`	—
Kadmin password	An IPA Admin password	—
Keytabs directory	A directory with keytab files	/etc/security/keytabs
Additional realms	Additional Kerberos realms	—
Delay between kinit invocation attempts	Delay between kinit invocation attempts	5
Number of retries for kinit invocation attempts	Number of retries for kinit invocation attempts	10
Trusted Active Directory server	An Active Directory server for one-way cross-realm trust from the MIT Kerberos KDC	—
Trusted Active Directory realm	An Active Directory realm for one-way cross-realm trust from the MIT Kerberos KDC	—
Custom krb5.conf	Enables adding custom parameters to the *krb5.conf* file	False
krb5.conf	Additional parameters to write to the *krb5.conf* file	—
Admin DN	A full DistinguishedName of the admin user with rights to `create/modify/delete/pwdchange` user accounts in the target Organizational Unit	—
LDAP URL	LDAP URL that consists of the `ldap://` or `ldaps://` protocol prefix, hostname or IP address and port of an AD server. For example, `ldaps://192.168.4.2:636`	—
Container DN	A container distinguished name	—
TLS CA certificate Path	A CA certificate path on the host filesystem that is pre-located or should be written from the *TLS CA certificate* field	—
TLS CA certificate (optional)	A CA certificate that is written on a host by the path from *TLS CA certificate Path* during the execution of the *Manage Kerberos* action. The certificate is not saved in ADCM	—
Custom ldap.conf	Enables adding custom parameters to the *ldap.conf* file	False
ldap.conf	Additional parameters that are written to the *ldap.conf* file	—
IpaClient No NTP Autoconf	Disables the NTP configuration during the IPA client installation	False
IpaClient No DNS Lookup	Disables the DNS lookup for the FreeIPA server during the IPA client installation	True
Custom smb.conf	Usage of custom configuration file *smb.conf*	False
smb.conf	Custom configuration file *smb.conf*	—
Set up Kerberos utils	Enables installation or removal of Kerberos clients and utils. Affects the *Expand* and *Install* actions	True
Configure Kerberos on hosts	Enables cluster configuration, including *krb5.conf*, *ldap.conf*	True
Set up principals and keytabs	Enables creation, recreation, or removal of principals and keytabs. Passwords for principals are generated randomly before keytab creation. Affects the *Expand* and *Install* actions. ADCM bundle will set up owner and permissions for keytabs only if this checkbox is selected in the cluster configuration. In case of absence of admin permissions, a customer should provide the prepared keytabs with correctly set owner and permissions (see Custom keytab recommendations)	True
Configure services and clients	Enables updating of services and clients configuration	True
Run service checks	Enables service check runs	True

- The *Credential encryption* section with the encryption provider settings:
  - *Credstore password* — encryption provider password;
  - *Credstore options* — the way of storing the encryption provider password: `no password`, `password-file`, or `password in the environment`.
- The *SSL* section with the settings listed in the table below.

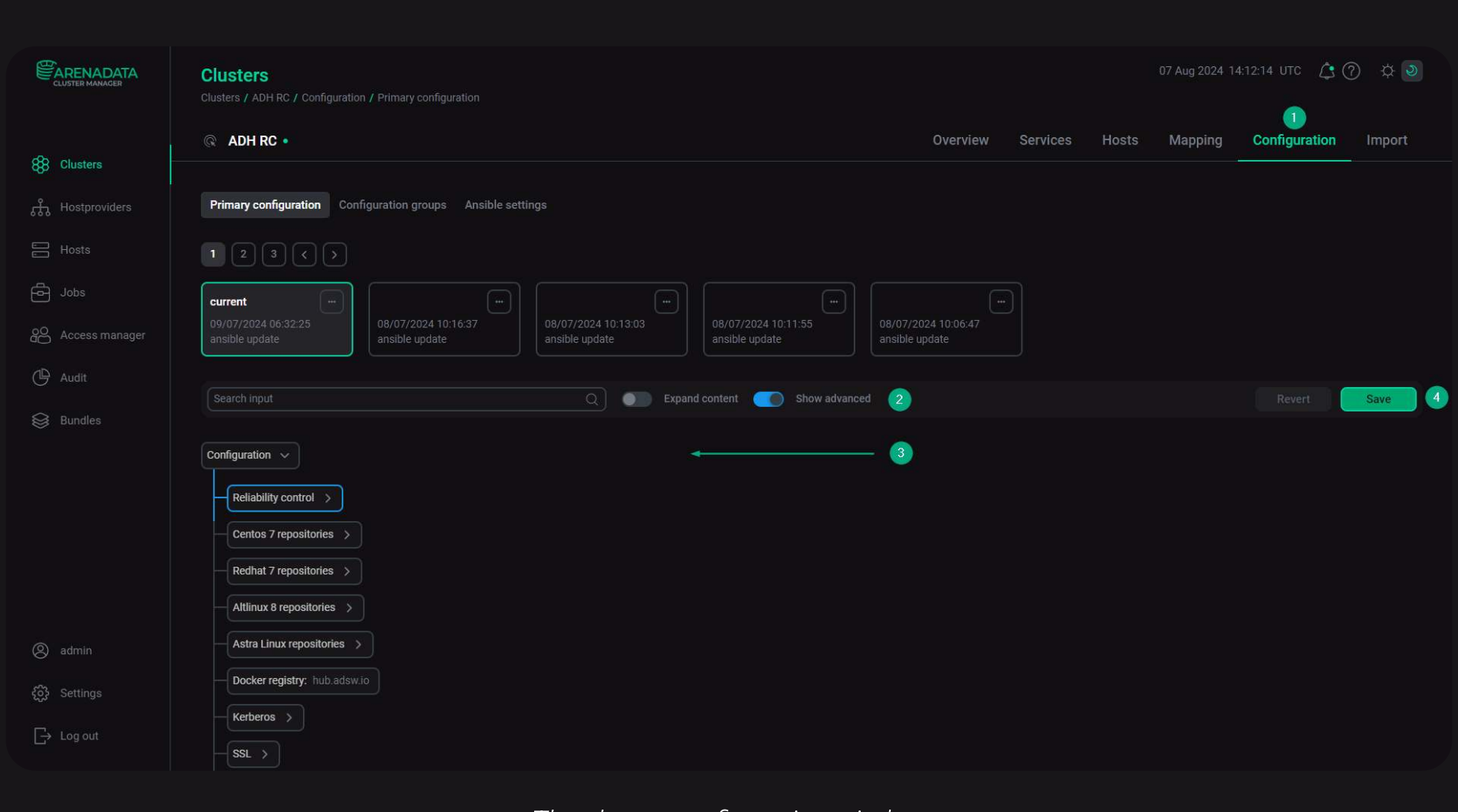
Parameter	Description	Default value
Enable SSL	Enables SSL	false
Keystore path	A path to the keystore file	/etc/ssl/keystore.jks
Keystore password	A password for the keystore file	—
Truststore path	A path to the truststore file	/etc/ssl/truststore.jks
Truststore password	A password for the truststore file	—
TLS Version	Version of the TLS protocol. See Set TLS version for an ADH cluster	TLSv1.2
Run service checks	Enables service check runs	true

- The *java.io.tmpdir* parameter that specifies an environment variable to control `java.io.tmpdir` for YARN, HBase, Hive, Spark3.
- The *ad-runtime-utils* parameter contains a configuration file template that specifies the runtime detection installation path settings for a given runtime (Java, Python, etc.). The template contains cluster settings as well as service-specific settings for Trino, Flink, and Ozone. You can use this parameter to explicitly define the Java version. For more information, see Software Requirements — Java.
  - The `default` section configures the default behavior for the cluster.
  - The `autodetect` section defines the algorithm for automatically detecting different Java versions. For each version (8, 17, 21, 23, 24), the following is configured:
    - `env_var` — an environment variable that specifies version (for example, `JAVA8_HOME`, `JAVA17_HOME`);
    - `paths` — a list of paths to search for the Java installation.
  - The `services` section defines services for which specific Java versions are required.

ad-runtime-utils

```
default:
  runtimes:
    java:
      version: "8"
      env_var: JAVA_HOME

autodetect:
  runtimes:
    java:
      "8":
        env_var: JAVA8_HOME
        paths:
          - /usr/java/jdk1.8
          - /usr/java/jre1.8
          - /usr/lib/jvm/j2sdk1.8-oracle
          - /usr/lib/jvm/j2sdk1.8-oracle/jre
          - /usr/lib/jvm/java-8-oracle
          - /usr/lib/jdk8-latest
          - /usr/lib/jvm/java-1.8.0
          - /usr/lib/jvm/java-1.8.0-oracle
          - /usr/lib/jvm/bellsoft-java8-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-8

      "17":
        env_var: JAVA17_HOME
        paths:
          - /usr/lib/jvm/bellsoft-java17-amd64
          - /usr/lib/jvm/bellsoft-java17
          - /usr/lib/jvm/liberica-jdk-17*
          - /usr/lib/jvm/liberica-jre-17*
          - /usr/lib/jvm/zulu-17*
          - /usr/lib/jvm/temurin-17*
          - /usr/lib/jvm/adoptopenjdk-17*
          - /usr/lib/jvm/jdk-17*
          - /usr/lib/jvm/java-17*
          - /usr/lib/jvm/openjdk-17*
          - /usr/lib/jvm/openjdk-17*
          - /opt/java/bellsoft-java17-amd64
          - /opt/java/temurin-17*
          - /usr/local/java/bellsoft-java17-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-17
          - /usr/lib64/jvm/java-17*

      "21":
        env_var: JAVA21_HOME
        paths:
          - /usr/lib/jvm/java-21*
          - /usr/lib/jvm/jdk-21*
          - /usr/lib/jvm/java-arenadata-openjdk-21

      "23":
        env_var: JAVA23_HOME
        paths:
          - /usr/lib/jvm/bellsoft-java23-amd64
          - /usr/lib/jvm/bellsoft-java23
          - /usr/lib/jvm/liberica-jdk-23*
          - /usr/lib/jvm/liberica-jre-23*
          - /usr/lib/jvm/zulu-23*
          - /usr/lib/jvm/temurin-23*
          - /usr/lib/jvm/adoptopenjdk-23*
          - /usr/lib/jvm/jdk-23*
          - /usr/lib/jvm/java-23-openjdk
          - /usr/lib/jvm/openjdk-23*
          - /opt/java/bellsoft-java23-amd64
          - /opt/java/temurin-23*
          - /usr/local/java/bellsoft-java23-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-23

      "24":
        env_var: JAVA24_HOME
        paths:
          - /usr/lib/jvm/bellsoft-java24-amd64
          - /usr/lib/jvm/bellsoft-java24
          - /usr/lib/jvm/liberica-jdk-24*
          - /usr/lib/jvm/liberica-jre-24*
          - /usr/lib/jvm/zulu-24*
          - /usr/lib/jvm/temurin-24*
          - /usr/lib/jvm/adoptopenjdk-24*
          - /usr/lib/jvm/jdk-24*
          - /usr/lib/jvm/java-24-openjdk
          - /usr/lib/jvm/openjdk-24*
          - /opt/java/bellsoft-java24-amd64
          - /opt/java/temurin-24*
          - /usr/local/java/bellsoft-java24-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-24

services:
  TRINO:
      path: /etc/trino/conf/trino-java.yaml
  FLINK2:
      path: /etc/flink2/conf/flink2-java.yaml
  OZONE:
      path: /etc/ozone/conf/ozone-java.yaml
  SPARK4:
      path: /etc/spark4/conf/spark4-java.yaml
```

- The *Precheck packages* parameter that specifies whether the availability of the packages in the repository should be checked before installation. `False` by default.
- *Cluster UUID* — displays the universally unique identifier of the cluster.
- The *Ranger plugins version* parameter that allows you to override the version number for the Ranger plugins.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH RC / Configuration / Primary configuration
07 Aug 2024 14:12:14 UTC
ADH RC
1
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Primary configuration
Configuration groups
Ansible settings
current
Search input
Expand content
Show advanced
2
Revert
Save
4
3
Configuration
Reliability control
Centos 7 repositories
Redhat 7 repositories
Altlinux 8 repositories
Astra Linux repositories
Kerberos
SSL
admin
Settings
Log out

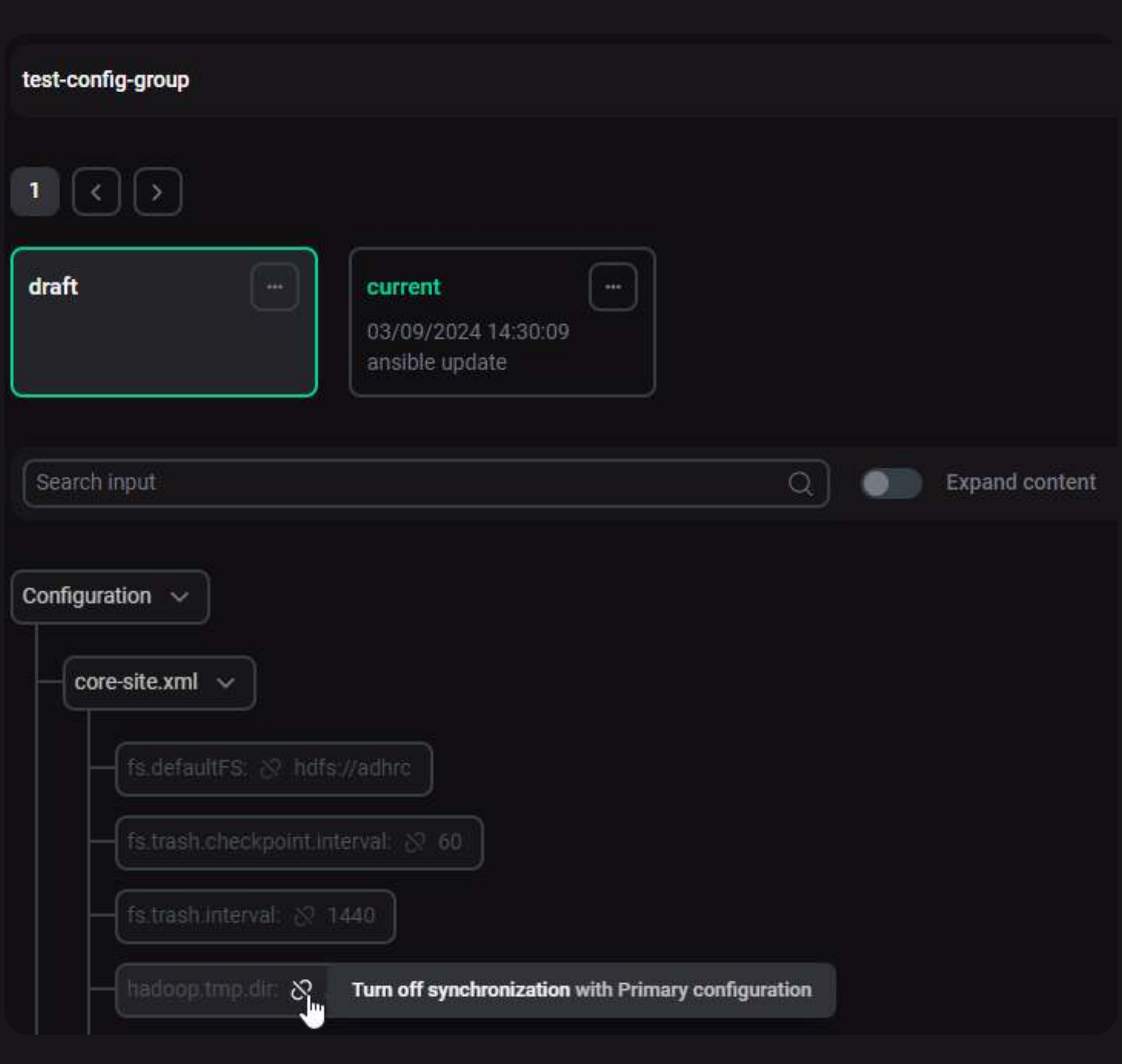
Docker registry: hub.adsw.io

The cluster configuration window

Configuration groups

The *Configuration* → *Configuration groups* tab is designed to set cluster configuration groups.

To configure parameters in a configuration group, hover over the parameters that you wish to configure differently, click the *Turn off synchronization* icon, edit them, and click *Save*.



test-config-group
1
draft
current
03/09/2024 14:30:09
ansible update
Search input
Expand content
Configuration
core-site.xml
Turn off synchronization with Primary configuration

fs.defaultFS: hdfs://adhrc

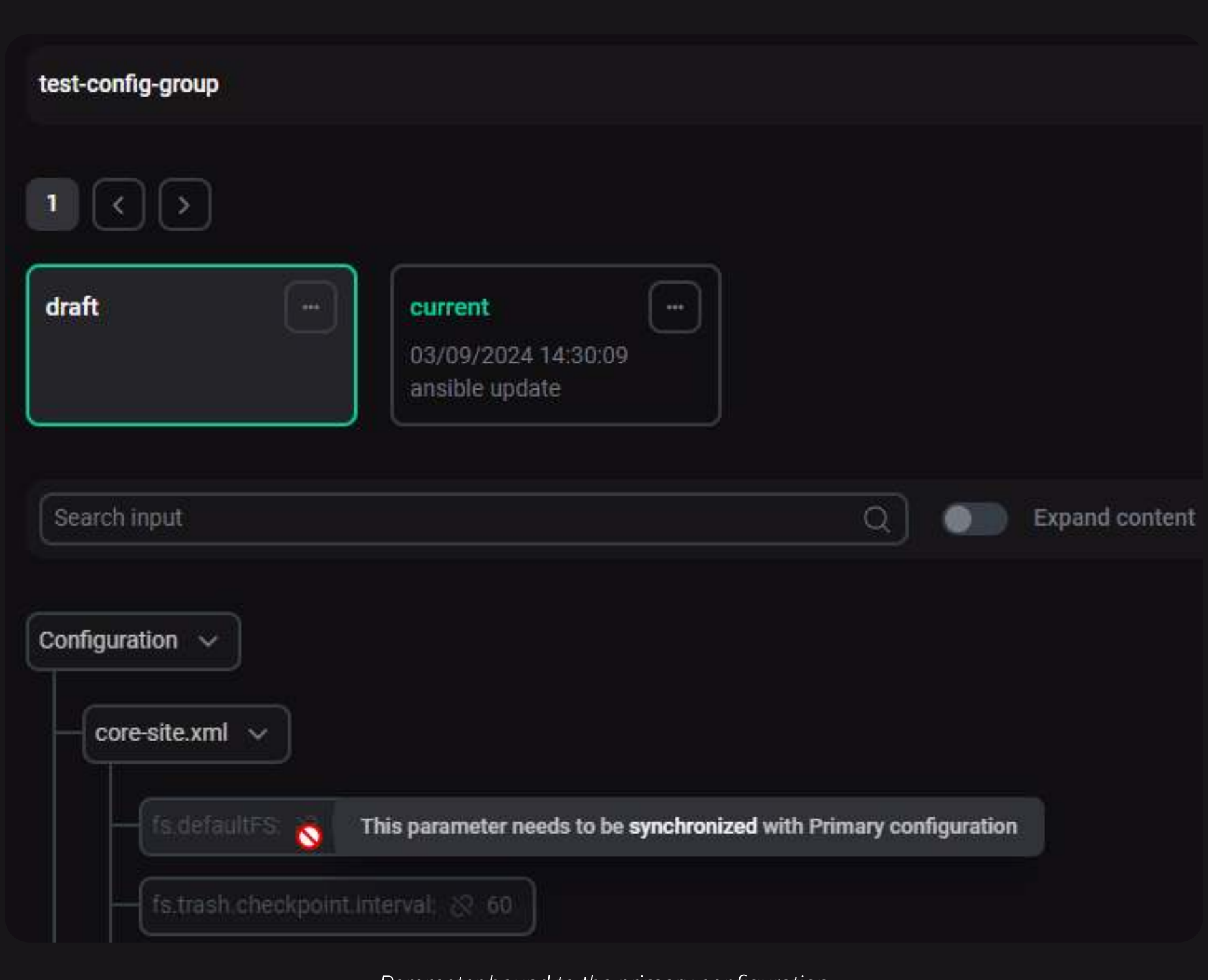
fs.trash.checkpoint.interval: 60

fs.trash.interval: 1440

hadoop.tmp.dir:

Configuration tab on the config group page

Some parameters you cannot change within a configuration group as they must always be synchronized with the primary configuration.



test-config-group
1
draft
current
03/09/2024 14:30:09
ansible update
Search input
Expand content
Configuration
core-site.xml
fs.defaultFS:
This parameter needs to be synchronized with Primary configuration

fs.trash.checkpoint.interval: 60

Parameter bound to the primary configuration

Ansible settings

The *Configuration* → *Ansible settings* tab allows you to set Ansible configuration options at the cluster level. The tab is available starting with ADCM 2.2.0.

Ansible settings

Name	Description	Default
forks	The number of parallel processes to spawn when communicating with remote hosts	5

Install a cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

- Step 1. Installation
  - Whole cluster
  - Single services
- Step 2. View installation results
- Step 3. Check the cluster state after the installation

Step 1. Installation

There are two ways to install cluster services:

1. **Whole cluster**. In this method all services are installed automatically one by one after applying the ***Install*** action to a cluster.
2. **Single services**. In this method each service is installed manually after applying one or more actions to it. This way is also suitable for installing new services in the already deployed cluster.

In both cases each service starts automatically after installation.

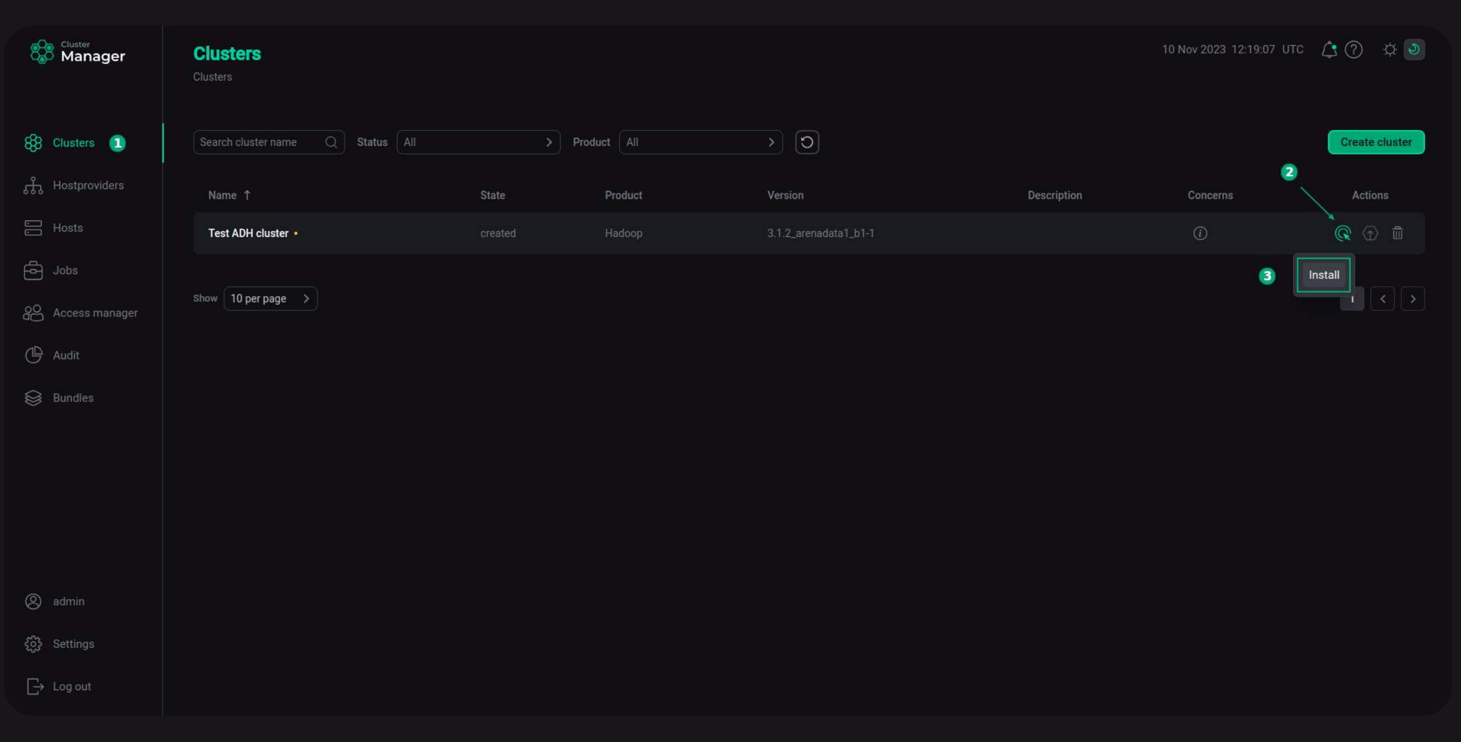
IMPORTANT

The installation must be performed on a clean host without prior installation of related or similar software.

Whole cluster

To install all cluster services within a single action, follow the steps:

1. Apply the ***Install*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.

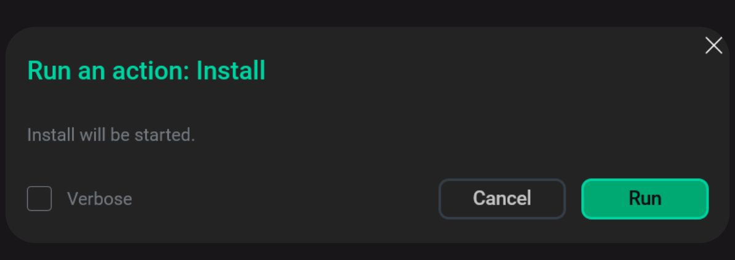


Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

Cluster Manager
Clusters
Clusters 1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
2
3
Install
Show 10 per page
10 Nov 2023 12:19:07 UTC

Switch to the whole cluster installation

2. Verify the requested action in the opened window.



Run an action: Install

Install will be started.

☐ Verbose

Verify the action

3. Wait until the job is completed.

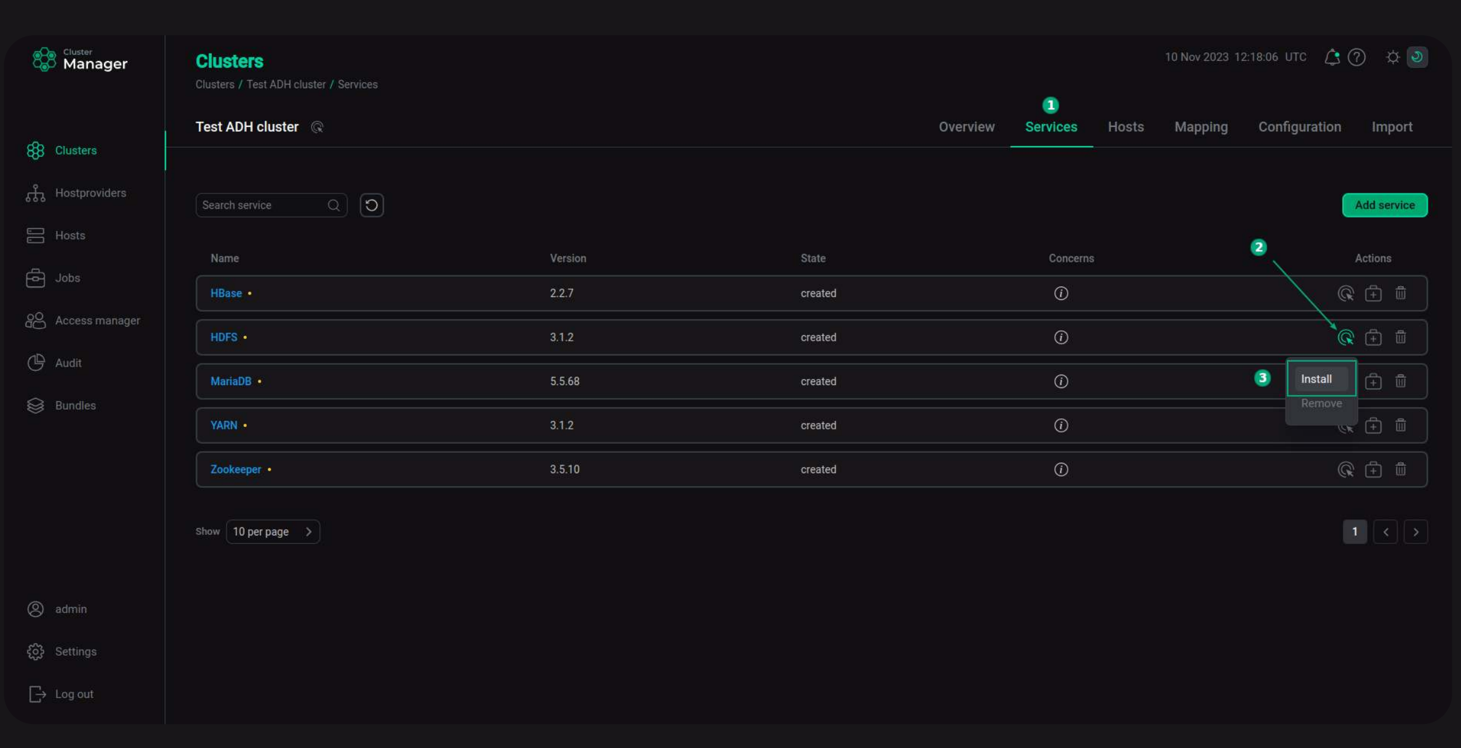
Single services

To install one or several cluster services manually, add these services to the cluster and then perform the actions described below.

CAUTION

Notice that services are installed one by one. Do not install another service until the installation of the previous one is completed.

1. Open the ***Services*** tab on the cluster page. For each service that should be installed, click the icon in the ***Actions*** column and select the ***Install*** action.



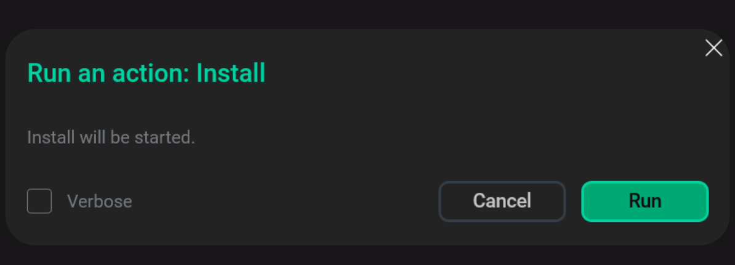
Clusters / Test ADH cluster / Services

Name	Version	State	Concerns	Actions
HBase	2.2.7	created		
HDFS	3.1.2	created		
MariaDB	5.5.68	created		
YARN	3.1.2	created		
Zookeeper	3.5.10	created		

Cluster Manager
Clusters
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
2
3
Install
Remove
Search service
Add service
Show 10 per page
10 Nov 2023 12:18:06 UTC
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Switch to installation of single services

2. Verify the action in the opened window.



Run an action: Install

Install will be started.

☐ Verbose

Verify the action

3. Wait until the job is completed. Then check that the service state has changed from `created` to `installed`.
4. Repeat the previous actions for other services that should be installed.

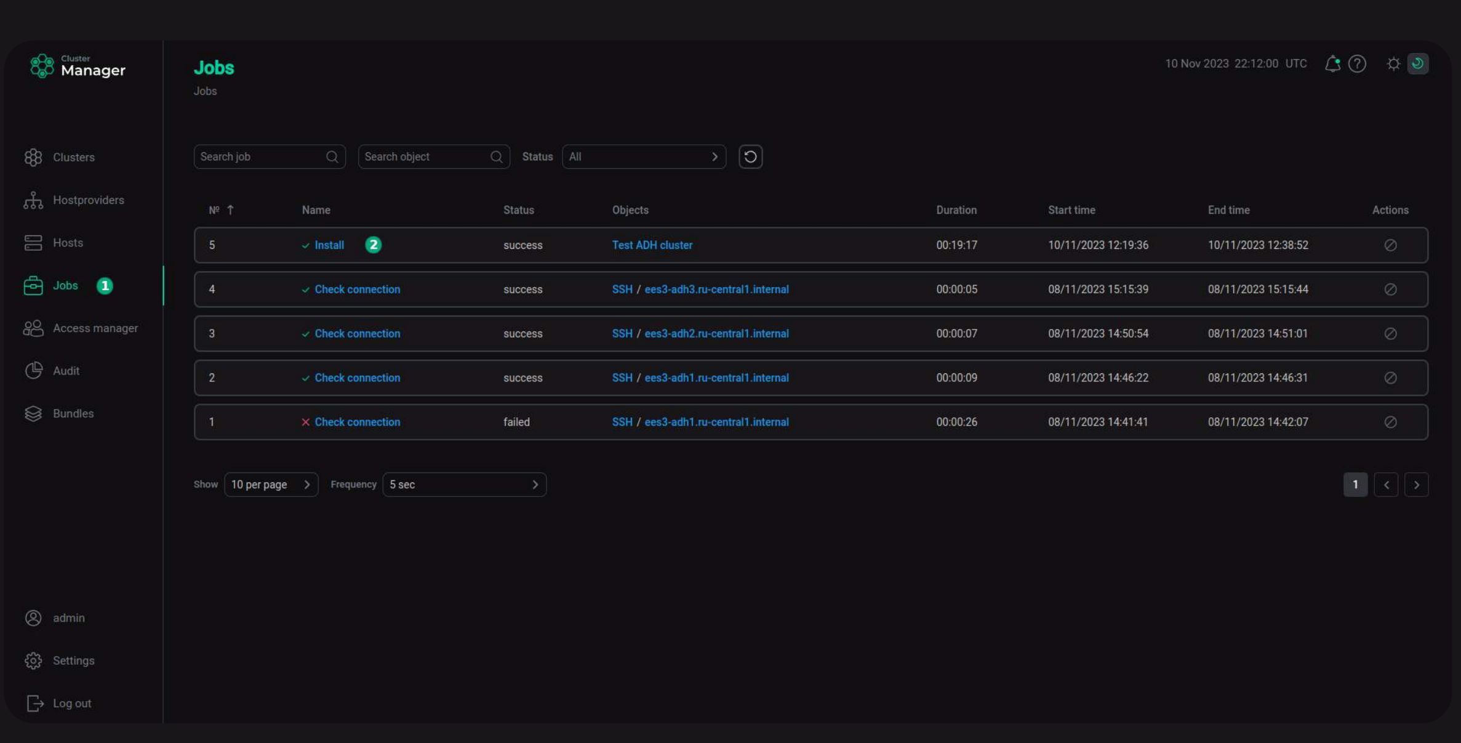
IMPORTANT

When you install services manually, one by one, use the following order:

1. Zookeeper
2. HDFS
3. YARN
4. HBase
5. Hive
6. Spark3
7. Monitoring

Step 2. View installation results

ADCM starts a single job for installation process, as well as for any other task. You can find out about the status of jobs more specifically on the ***Jobs*** page.

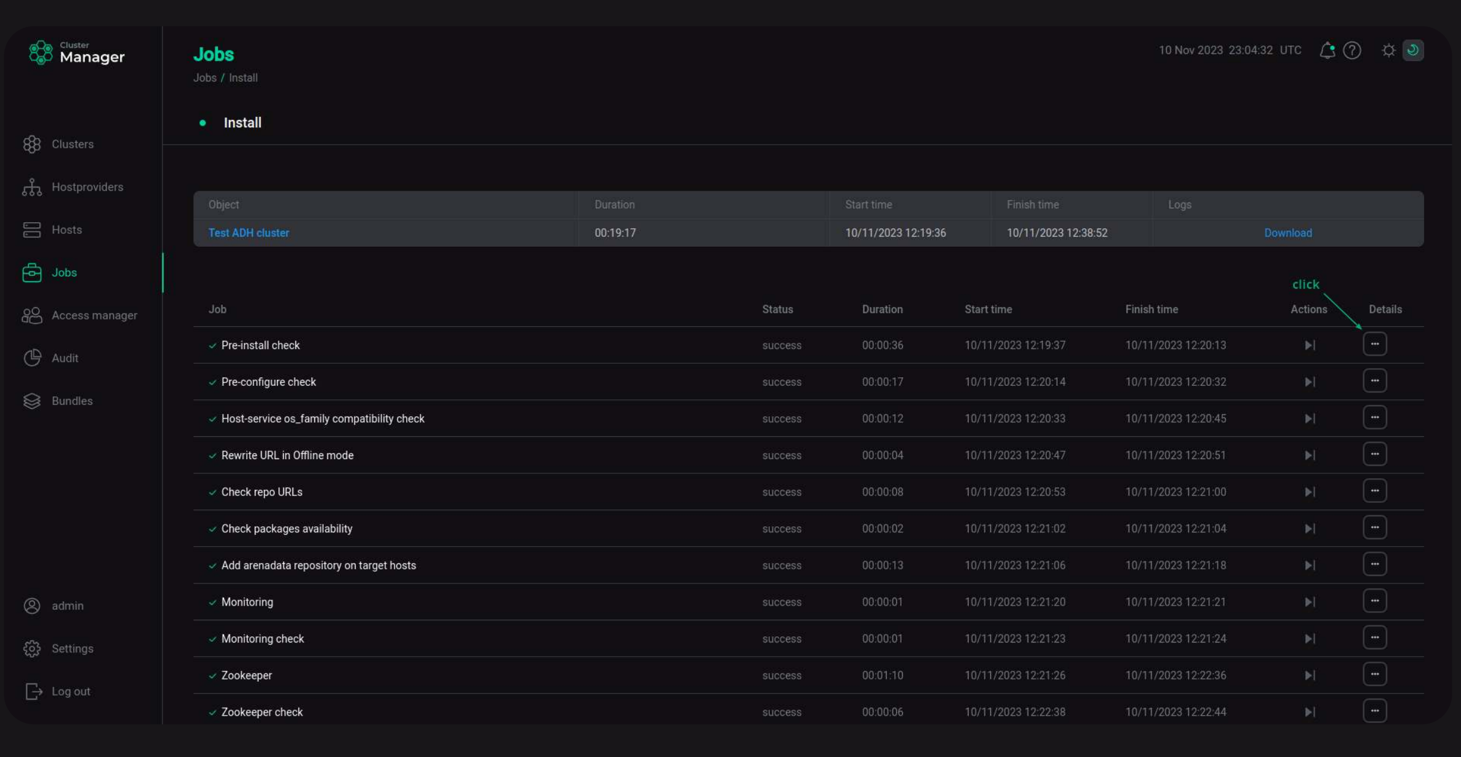


№	Name	Status	Objects	Duration	Start time	End time	Actions
5	✓ Install	success	Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	
4	✓ Check connection	success	SSH / ees3-adh3.ru-central1.internal	00:00:05	08/11/2023 15:15:39	08/11/2023 15:15:44	
3	✓ Check connection	success	SSH / ees3-adh2.ru-central1.internal	00:00:07	08/11/2023 14:50:54	08/11/2023 14:51:01	
2	✓ Check connection	success	SSH / ees3-adh1.ru-central1.internal	00:00:09	08/11/2023 14:46:22	08/11/2023 14:46:31	
1	✗ Check connection	failed	SSH / ees3-adh1.ru-central1.internal	00:00:26	08/11/2023 14:41:41	08/11/2023 14:42:07	

Cluster Manager
Jobs
Clusters
Hostproviders
Hosts
Jobs 1
Access manager
Audit
Bundles
admin
Settings
Log out
Search job
Search object
Status All
2
Show 10 per page
Frequency 5 sec
10 Nov 2023 22:12:00 UTC

The Jobs page

The successful completion of the service installation is determined by the transition of the job from the `running` status to the `success` status on the ***Jobs*** page. If the job fails, it switches to the `failed` status. In this case it is possible to see more detailed information about the errors that occurred via clicking the failed job on the ***Jobs*** page. The window with job inner steps is opened. Click the ellipsis in the ***Details*** column for additional information on the current step.



Jobs / Install

Install

Object	Duration	Start time	Finish time	Logs
Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	Download

Job	Status	Duration	Start time	Finish time	Actions	Details
✓ Pre-install check	success	00:00:36	10/11/2023 12:19:37	10/11/2023 12:20:13		
✓ Pre-configure check	success	00:00:17	10/11/2023 12:20:14	10/11/2023 12:20:32		
✓ Host-service os_family compatibility check	success	00:00:12	10/11/2023 12:20:33	10/11/2023 12:20:45		
✓ Rewrite URL in Offline mode	success	00:00:04	10/11/2023 12:20:47	10/11/2023 12:20:51		
✓ Check repo URLs	success	00:00:08	10/11/2023 12:20:53	10/11/2023 12:21:00		
✓ Check packages availability	success	00:00:02	10/11/2023 12:21:02	10/11/2023 12:21:04		
✓ Add arenadata repository on target hosts	success	00:00:13	10/11/2023 12:21:06	10/11/2023 12:21:18		
✓ Monitoring	success	00:00:01	10/11/2023 12:21:20	10/11/2023 12:21:21		
✓ Monitoring check	success	00:00:01	10/11/2023 12:21:23	10/11/2023 12:21:24		
✓ Zookeeper	success	00:01:10	10/11/2023 12:21:26	10/11/2023 12:22:36		
✓ Zookeeper check	success	00:00:06	10/11/2023 12:22:38	10/11/2023 12:22:44		

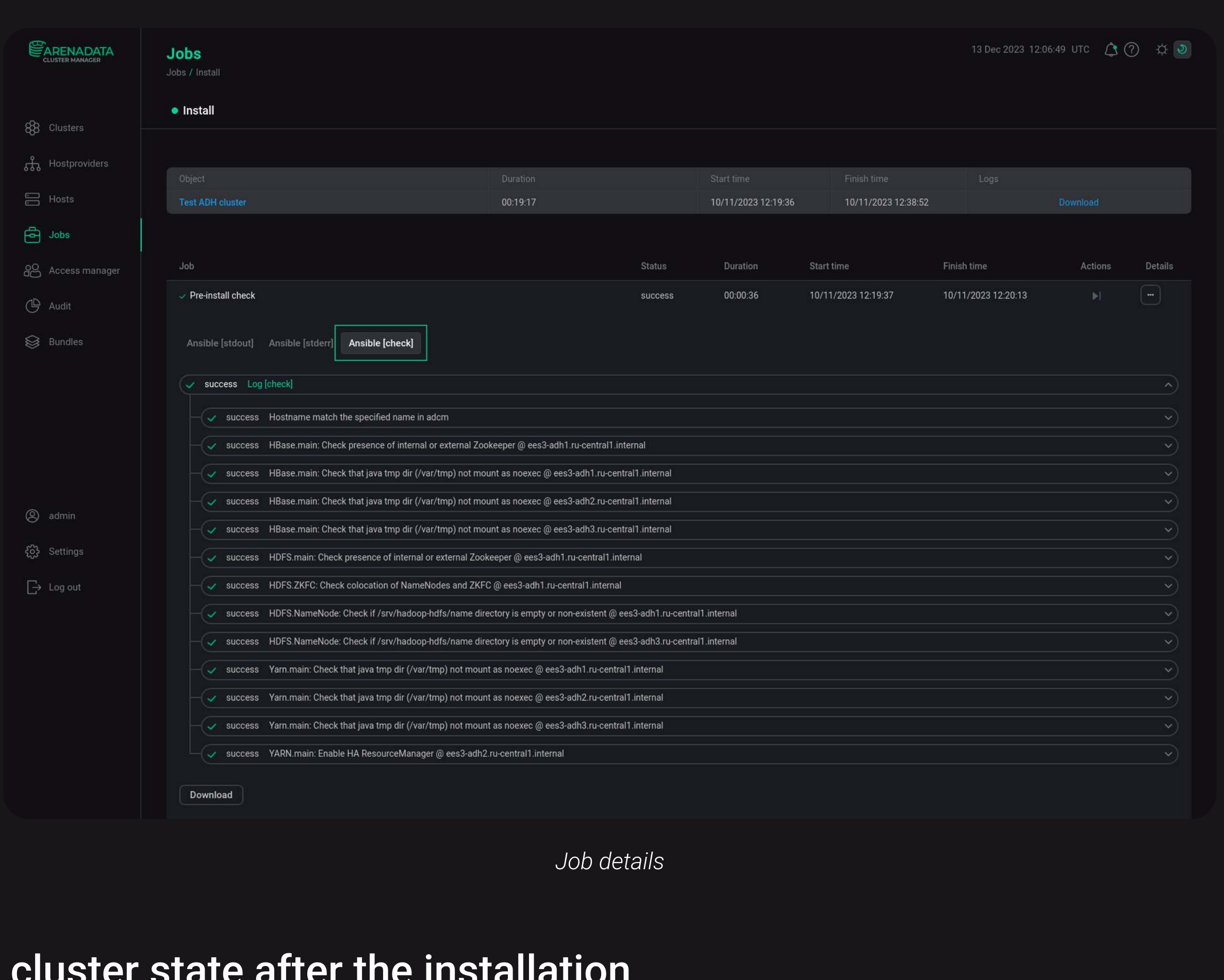
Cluster Manager
Jobs
click
10 Nov 2023 23:04:32 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Switch to job details

A job page contains two sections: ***ansible [stdout]*** and ***ansible [stderr]***. These are technical logs of the job that include information from standard I/O streams stdout/stderr. These logs can help to understand what caused the problems.

There can also be the optional third section ***ansible [check]***. These are the logs of the most frequent errors checks. The description of these errors is more simple and specific, than in two previous technical logs.

The contents of all three sections should be studied if errors occur.



Jobs / Install

Install

Object	Duration	Start time	Finish time	Logs
Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	Download

Job	Status	Duration	Start time	Finish time	Actions	Details
✓ Pre-install check	success	00:00:36	10/11/2023 12:19:37	10/11/2023 12:20:13		

```
success  Log [check]
  success  Hostname match the specified name in adcm
  success  HBase.main: Check presence of internal or external Zookeeper @ ees3-adh1.ru-central1.internal
  success  HBase.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh1.ru-central1.internal
  success  HBase.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh2.ru-central1.internal
  success  HBase.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh3.ru-central1.internal
  success  HDFS.main: Check presence of internal or external Zookeeper @ ees3-adh1.ru-central1.internal
  success  HDFS.ZKFC: Check colocation of NameNodes and ZKFC @ ees3-adh1.ru-central1.internal
  success  HDFS.NameNode: Check if /srv/hadoop-hdfs/name directory is empty or non-existent @ ees3-adh1.ru-central1.internal
  success  HDFS.NameNode: Check if /srv/hadoop-hdfs/name directory is empty or non-existent @ ees3-adh3.ru-central1.internal
  success  Yarn.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh1.ru-central1.internal
  success  Yarn.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh2.ru-central1.internal
  success  Yarn.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh3.ru-central1.internal
  success  YARN.main: Enable HA ResourceManager @ ees3-adh2.ru-central1.internal
```

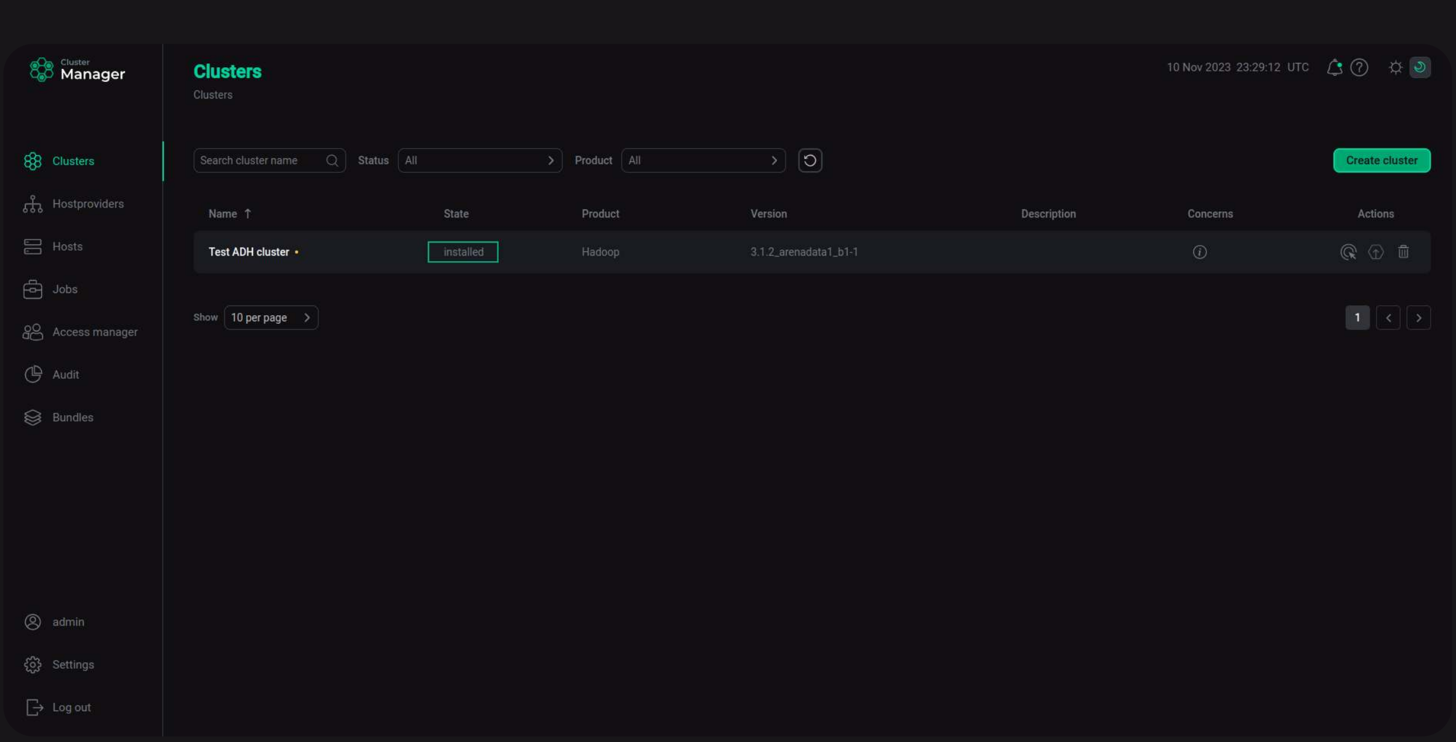
ARENADATA CLUSTER MANAGER
Jobs
13 Dec 2023 12:06:49 UTC
Ansible [stdout]
Ansible [stderr]
Ansible [check]
Download
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Job details

Step 3. Check the cluster state after the installation

As a result of successful installation, the cluster and its services change their state according to the following rules:

- The cluster state, displayed in the ***State*** field on the ***Clusters*** page, is being changed from `created` to `installed`.

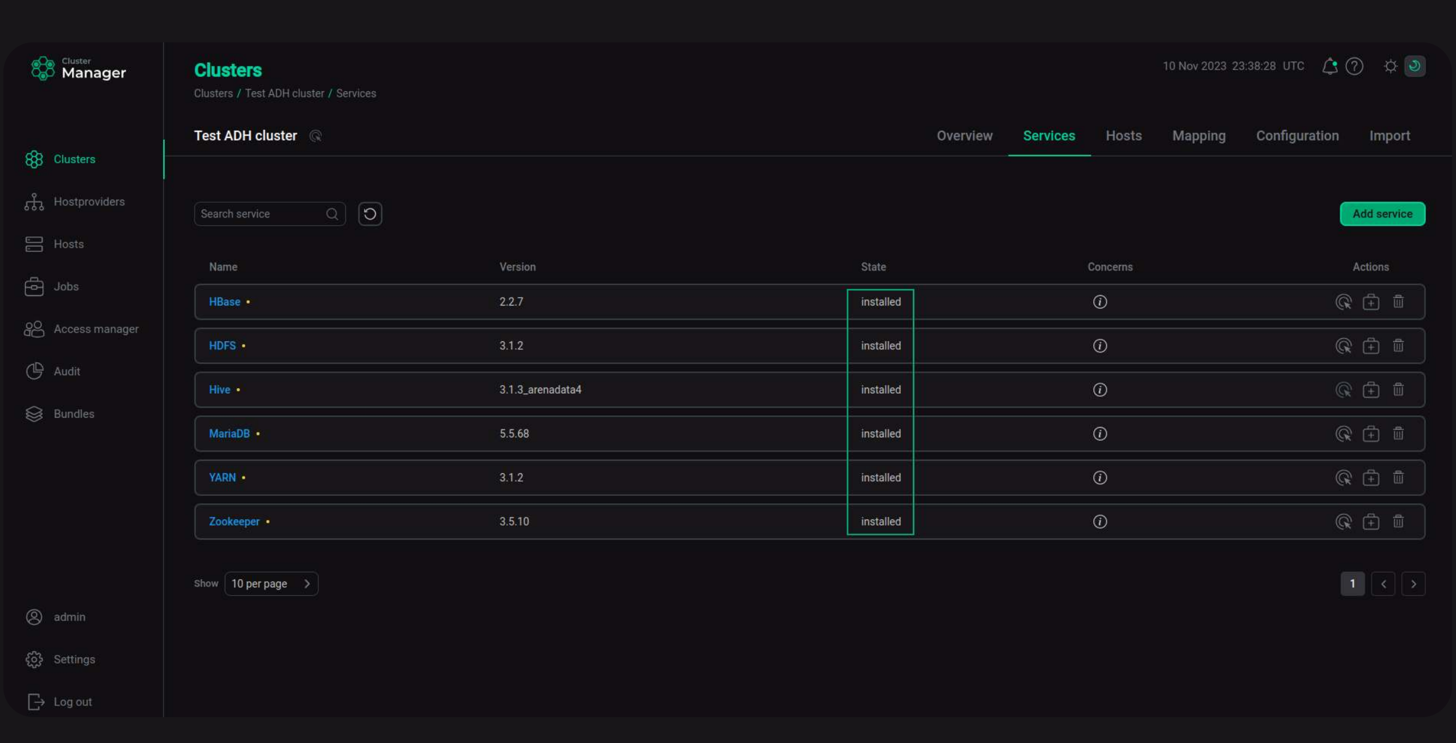


Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			

Cluster Manager
Clusters
10 Nov 2023 23:29:12 UTC
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

The cluster state after successful installation

- The state of cluster services, displayed on the ***Services*** tab of the cluster page, is being changed from `created` to `installed`.



Clusters / Test ADH cluster / Services

Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

Cluster Manager
Clusters
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
10 Nov 2023 23:38:28 UTC
Search service
Add service
Show 10 per page
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

The state of cluster services after successful installation

Install monitoring

Daria Barysheva

To Table of Contents

Monitoring is an optional but strongly recommended ADH component that you can configure in one of the following ways:

- add the Monitoring service to the ADH cluster (available starting with ADH 4.0.0);
- create a monitoring cluster and integrate it with the ADH cluster (deprecated).

Monitoring service

Elena Dvoryadkina, Elena Kostyuchenko

Contents

To Table of Contents

Overview
Step 1. Add the service
Step 2. Add components
Step 3. Configure the service
Step 4. Install the service
Step 5. View results
  Metrics in the Prometheus format
  Prometheus web interface
  Grafana web interface

The Monitoring service deploys the Prometheus server inside ADH to collect and store cluster monitoring metrics, and also supports the ability to use the Grafana web application for visualization and analysis of information. This article describes steps required to add this service to an ADH cluster.

Overview

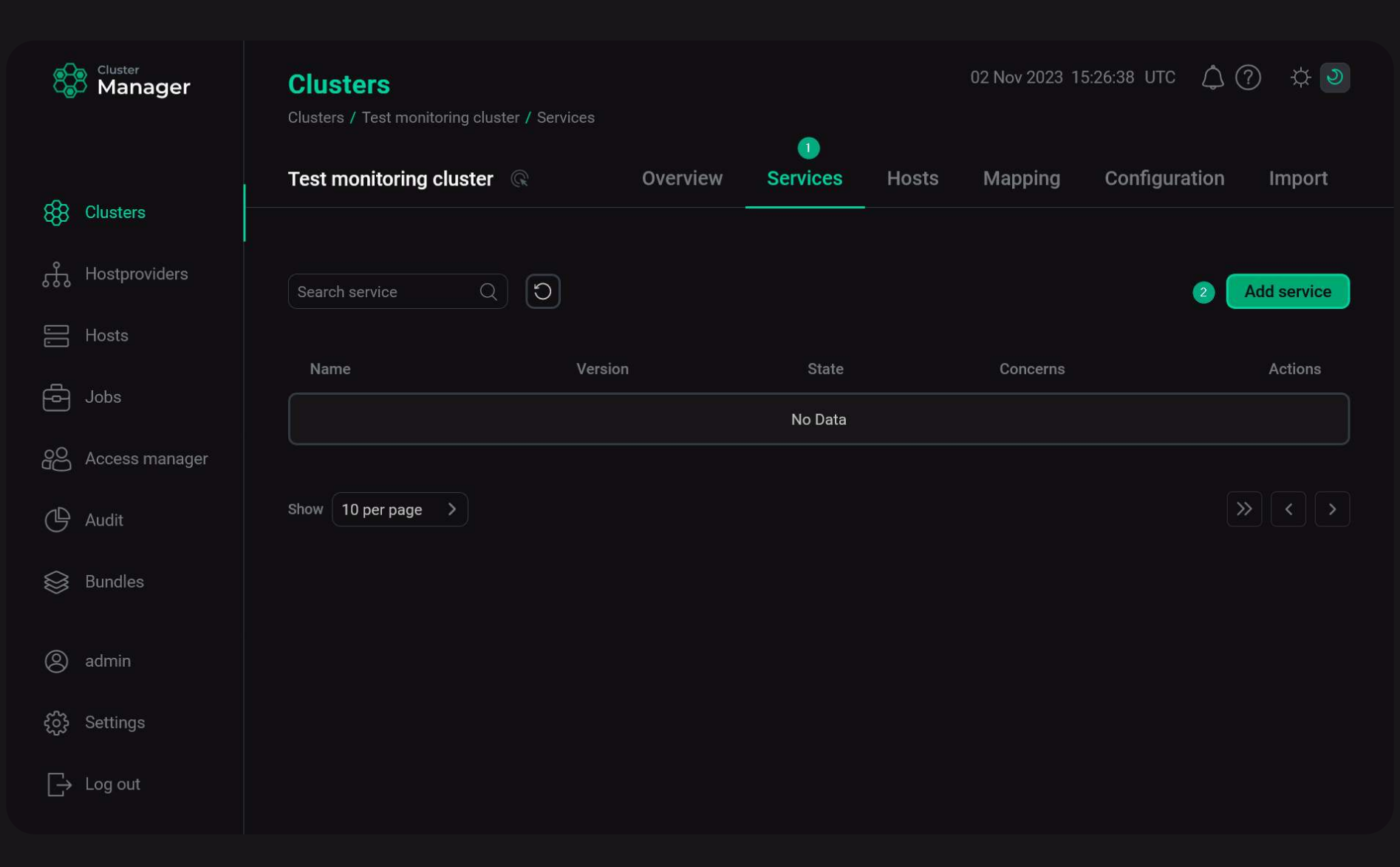
When adding the Monitoring service to ADH, you can install the Node Exporter on all or selected hosts of your cluster. It is a monitoring agent that reads host system metrics that Prometheus collects. Prometheus can also collect metrics from ADH services. These metrics will be available in the Prometheus format on the ports and endpoints specified in the service configurations. You can also use the Prometheus and Grafana web interfaces to view and analyze data that the monitoring service collects.

NOTE

- If you already have a Prometheus-compatible monitoring system set up (for example, your own Prometheus server or VictoriaMetrics), you can use it to collect ADH metrics. To do this, configure access to ADH metrics in your monitoring system using ADH's Prometheus parameters specified on the Monitoring service page.
- You can also use the Federation mechanism to migrate all metrics from the Prometheus server deployed in ADH to your Prometheus.

Step 1. Add the service

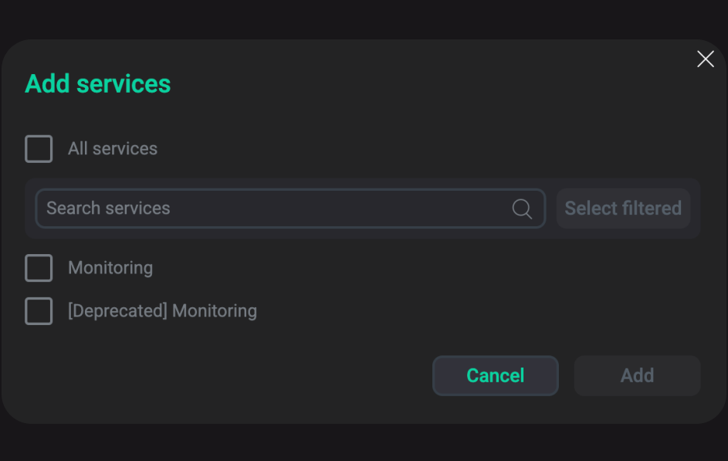
1. In the ADCM interface, open the ***Clusters*** page and click your ADH cluster name. On the cluster page that opens, switch to the ***Services*** tab and click ***Add services***.



Cluster Manager
Clusters
Clusters / Test monitoring cluster / Services
02 Nov 2023 15:26:38 UTC
Test monitoring cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show 10 per page

Switch to adding services

2. In the opened dialog, select the ***Monitoring*** service and click ***Add***.

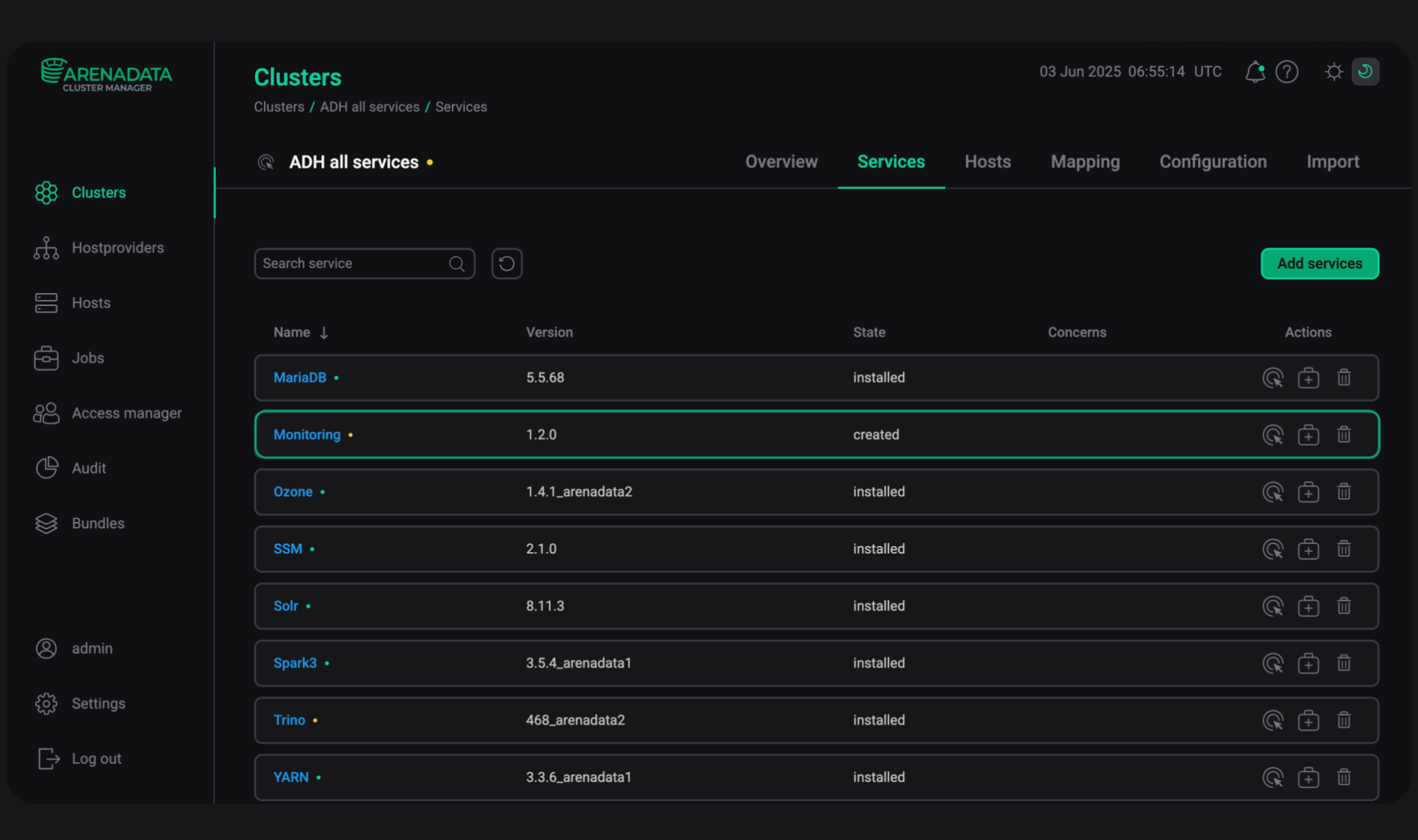


Add services
All services
Search services
Select filtered
Cancel
Add

☐ All services
☐ Monitoring
☐ [Deprecated] Monitoring

Select the service

As a result, the added service is displayed on the ***Services*** tab.



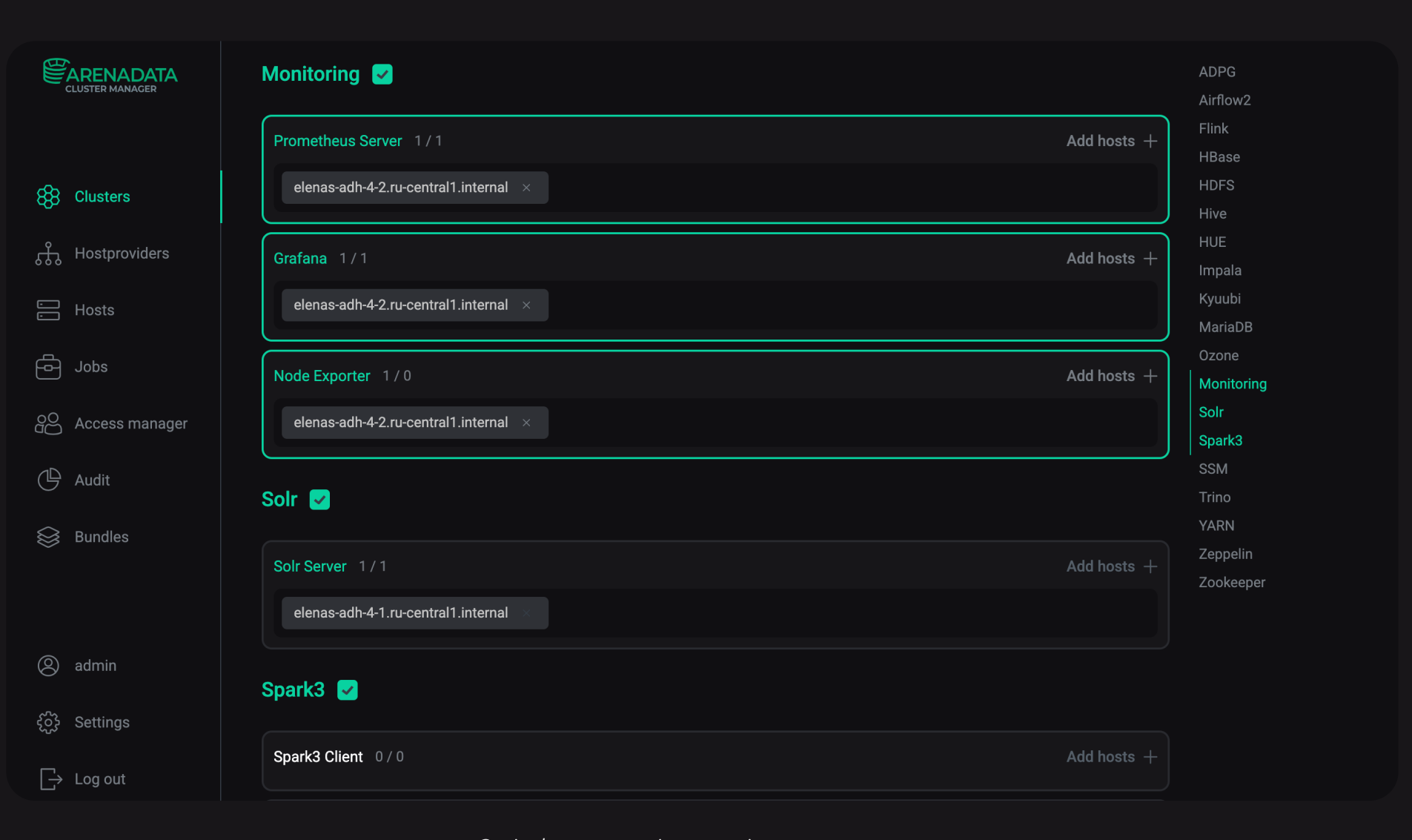
ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH all services / Services
03 Jun 2025 06:55:14 UTC
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	created		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

The result of successful adding the service to the cluster

Step 2. Add components

1. On the cluster page, open the ***Mapping*** tab to proceed to mapping service components to cluster hosts.



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Add hosts
ADPG
Airflow2
Flink
HBase
HDFS
Hive
HUE
Impala
Kyuubi
MariaDB
Ozone
Monitoring
Solr
Spark3
SSM
Trino
YARN
Zeppelin
Zookeeper

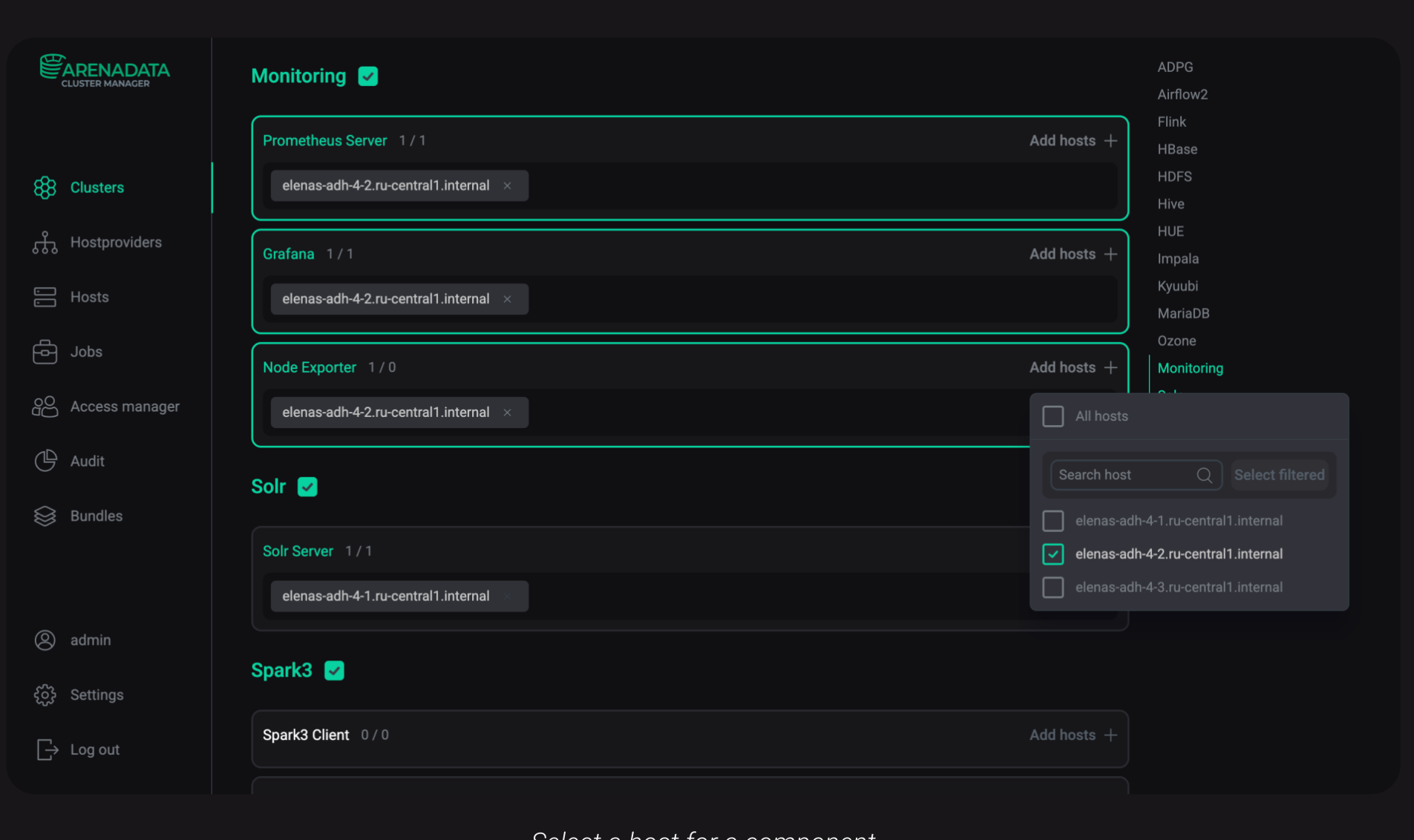
Monitoring ☑
Prometheus Server 1 / 1: elenas-adh-4-2.ru-central1.internal
Grafana 1 / 1: elenas-adh-4-2.ru-central1.internal
Node Exporter 1 / 0: elenas-adh-4-2.ru-central1.internal
Solr ☑
Solr Server 1 / 1: elenas-adh-4-1.ru-central1.internal
Spark3 ☑
Spark3 Client 0 / 0

Switch to mapping service components

Monitoring service components

Component	Description
Prometheus Server	Stores system metrics generated by an ADH cluster and allows Grafana to request them
Grafana	Visualizes ADH metrics as graphs and charts organized into dashboards
Node Exporter	Collects system metrics and sends them to Prometheus

2. Assign hosts to components of the Monitoring service — click ***Add hosts*** and select the desired hosts in the pop-up window.



ARENADATA CLUSTER MANAGER
Monitoring
Add hosts
All hosts
Search host
Select filtered

☐ elenas-adh-4-1.ru-central1.internal
☑ elenas-adh-4-2.ru-central1.internal
☐ elenas-adh-4-3.ru-central1.internal

Select a host for a component

CAUTION

It is not recommended to install the Prometheus Server on hosts with ADH components — use separate hosts for it. Otherwise, if an ADH host fails and/or the load on it is critically high, information about the corresponding problems will not be saved.

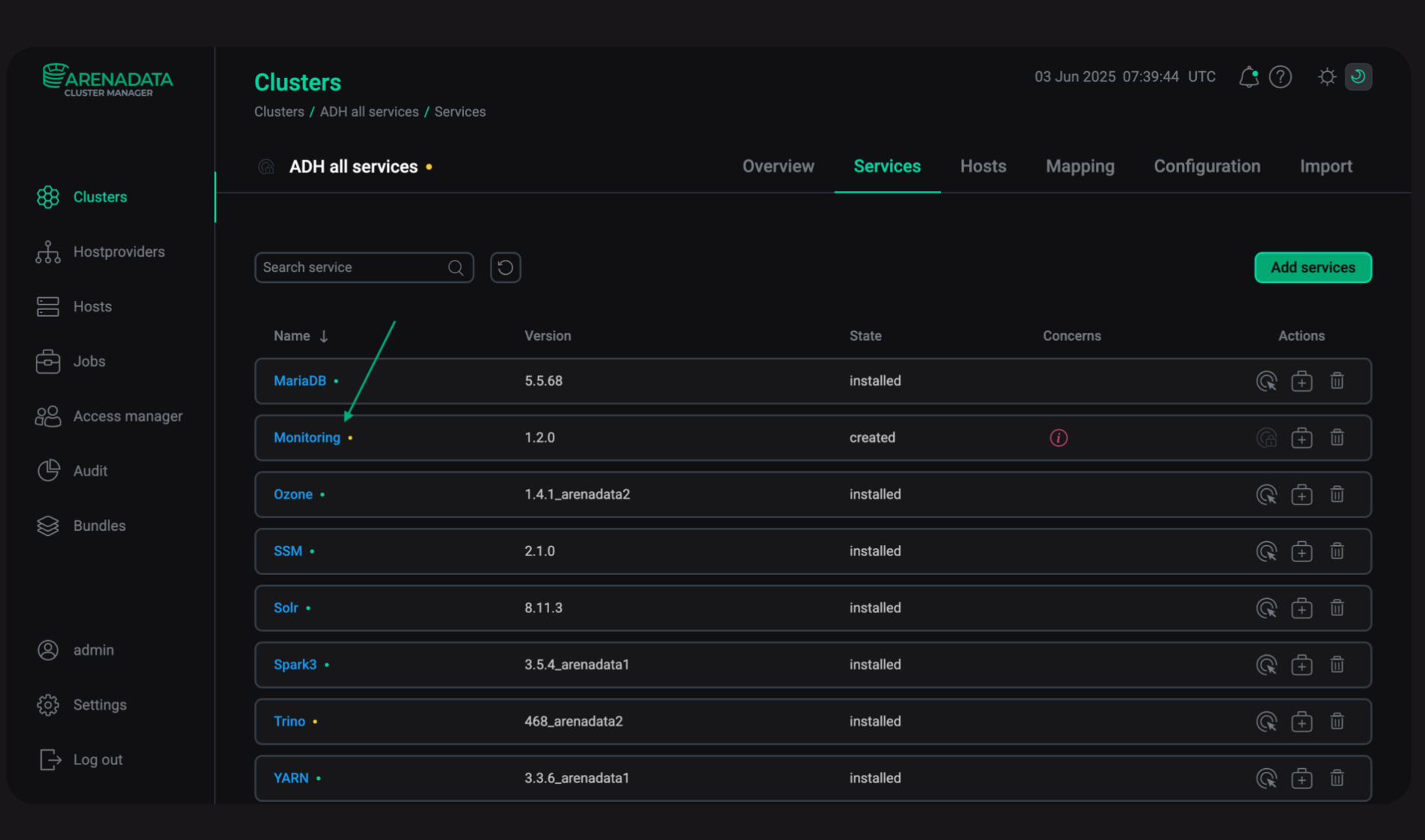
3. After the distribution of components is completed, click ***Save***.

NOTE

When changing the topology of services/configurations related to monitoring, it is necessary to restart the Prometheus component in the Monitoring service.

Step 3. Configure the service

1. Open the ***Services*** tab on the cluster page and click the ***Monitoring*** service name in the ***Name*** column.

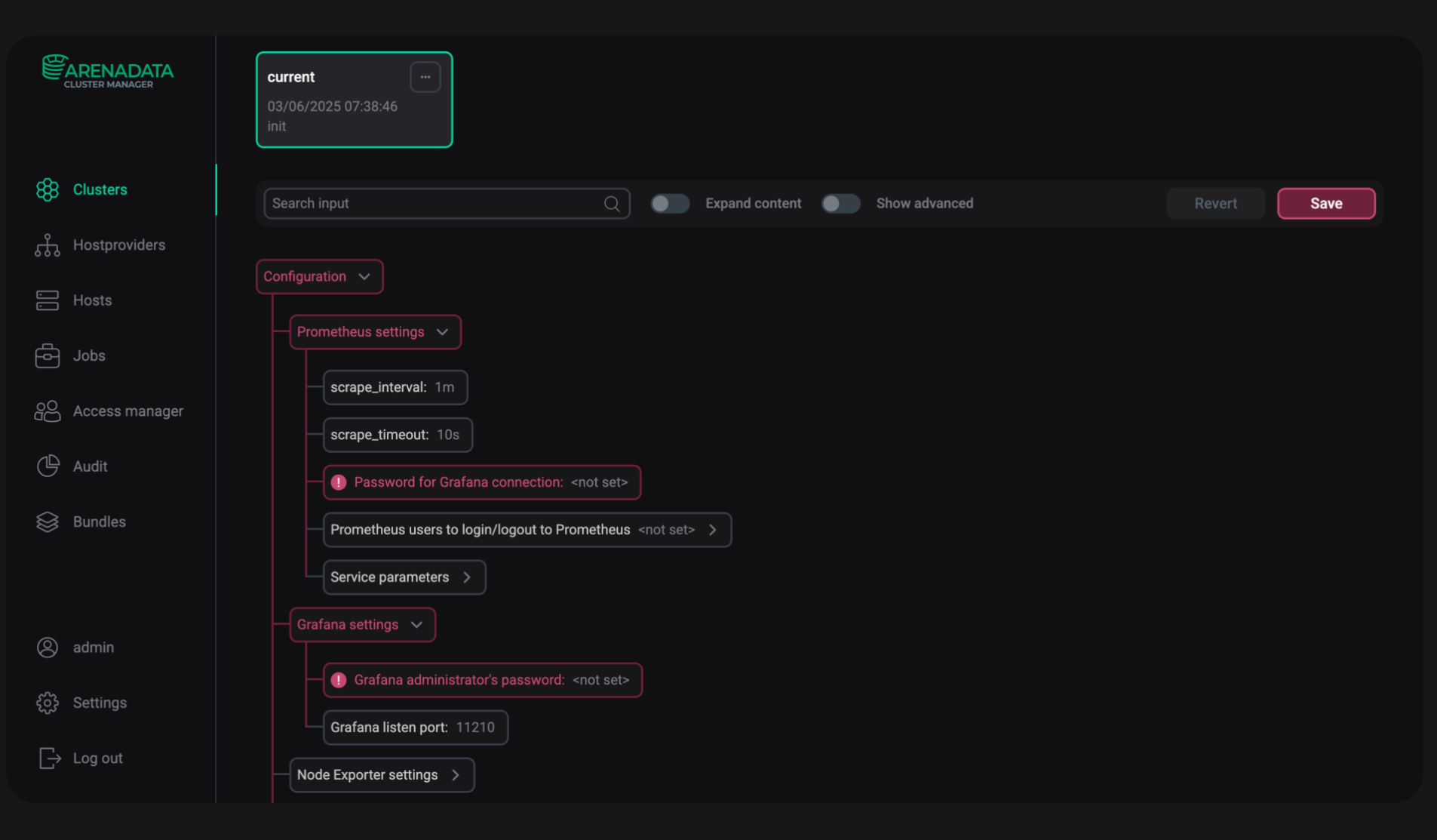


ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH all services / Services
03 Jun 2025 07:39:44 UTC
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	created	ⓘ	
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

Switch to the service configuration

2. On the page that opens, fill in the service's configuration parameters — see the Monitoring section for parameter descriptions. Fields highlighted in red are required.



ARENADATA CLUSTER MANAGER
current
03/06/2025 07:38:46
Init
Search input
Expand content
Show advanced
Revert
Save

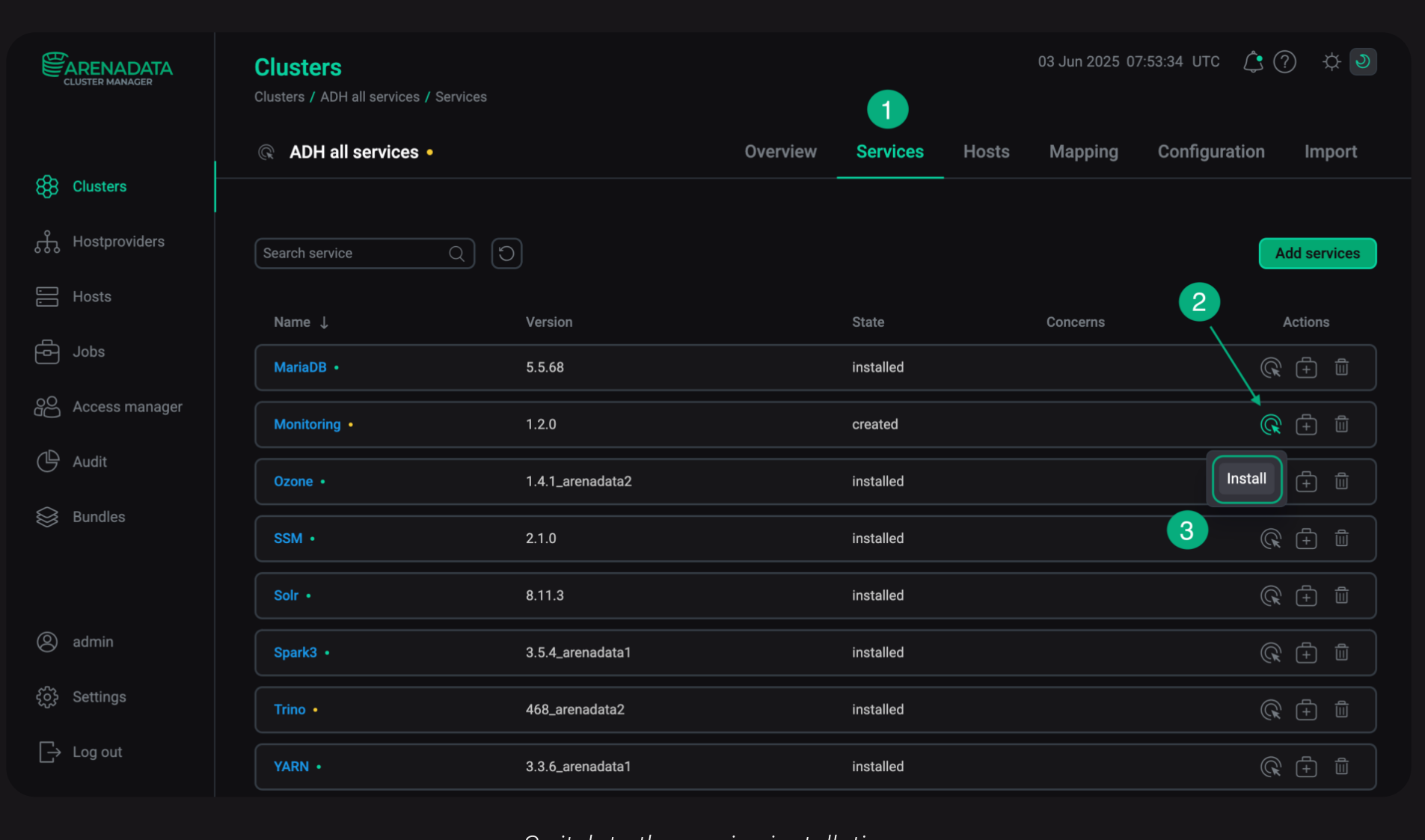
Configuration
Prometheus settings
scrape_interval: 1m
scrape_timeout: 10s
Password for Grafana connection: <not set>
Prometheus users to login/logout to Prometheus <not set>
Service parameters
Grafana settings
Grafana administrator's password: <not set>
Grafana listen port: 11210
Node Exporter settings

Configure the Monitoring service

After specifying all necessary parameters, click ***Save***.

Step 4. Install the service

1. On the ***Services*** tab, click the icon for the Monitoring service in the ***Actions*** column and run the ***Install*** action.

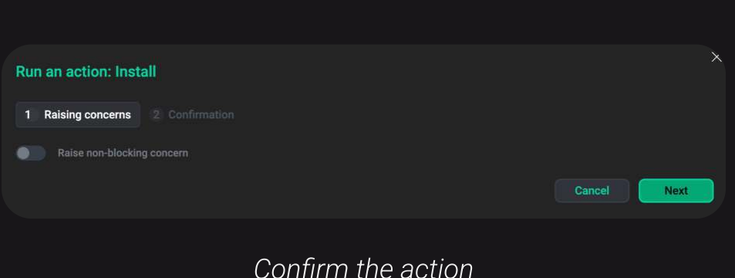


ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH all services / Services
03 Jun 2025 07:53:34 UTC
1
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
2
Install
3

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	created		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

Switch to the service installation

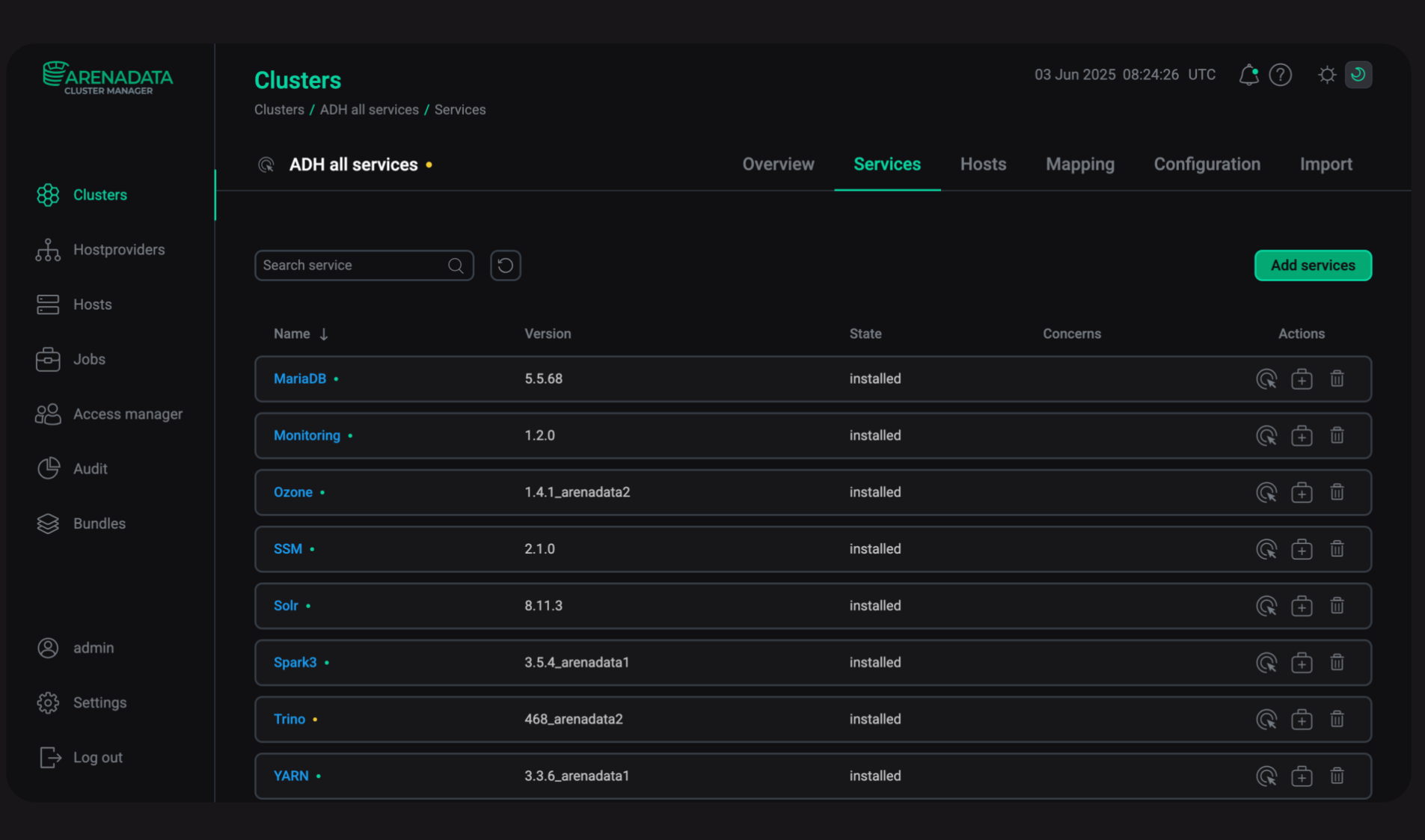
2. Confirm the action in the opened window.



Run an action: Install
1 Raising concerns
2 Confirmation
Raise non-blocking concern
Cancel
Next

Confirm the action

3. Wait until the installation is completed. Then check that the service state has changed from `created` to `installed`.

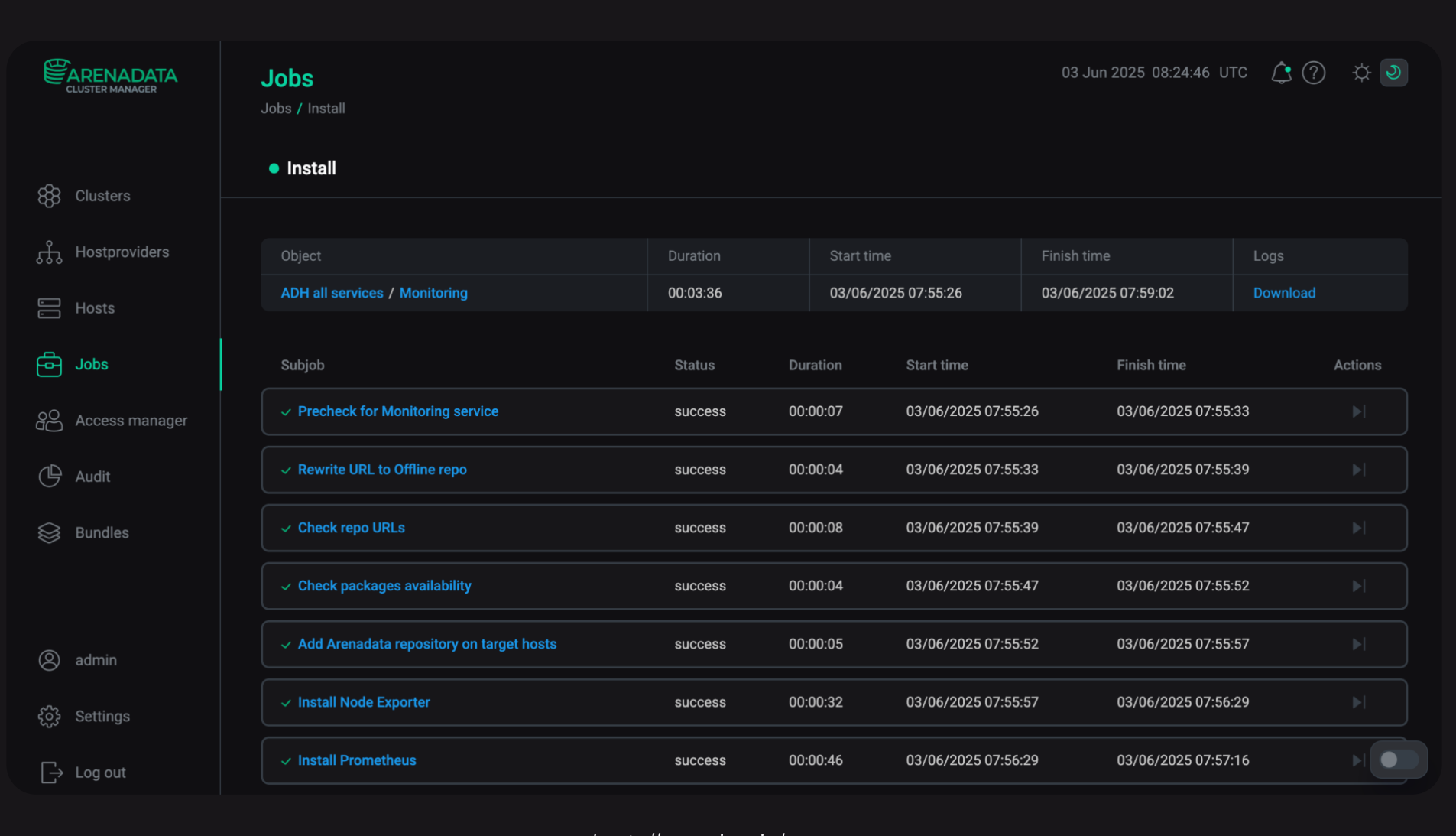


ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH all services / Services
03 Jun 2025 08:24:26 UTC
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	installed		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

Installation is complete

To view the service installation process and analyze errors if they occur, select ***Jobs*** in the left navigation menu and click the ***Install*** job name in the ***Jobs*** list.



ARENADATA CLUSTER MANAGER
Jobs
Jobs / Install
03 Jun 2025 08:24:46 UTC
Install

Object	Duration	Start time	Finish time	Logs
ADH all services / Monitoring	00:03:36	03/06/2025 07:55:26	03/06/2025 07:59:02	Download

Subjob	Status	Duration	Start time	Finish time	Actions
Precheck for Monitoring service	success	00:00:07	03/06/2025 07:55:26	03/06/2025 07:55:33	
Rewrite URL to Offline repo	success	00:00:04	03/06/2025 07:55:33	03/06/2025 07:55:39	
Check repo URLs	success	00:00:08	03/06/2025 07:55:39	03/06/2025 07:55:47	
Check packages availability	success	00:00:04	03/06/2025 07:55:47	03/06/2025 07:55:52	
Add Arenadata repository on target hosts	success	00:00:05	03/06/2025 07:55:52	03/06/2025 07:55:57	
Install Node Exporter	success	00:00:32	03/06/2025 07:55:57	03/06/2025 07:56:29	
Install Prometheus	success	00:00:46	03/06/2025 07:56:29	03/06/2025 07:57:16	

Install service job page

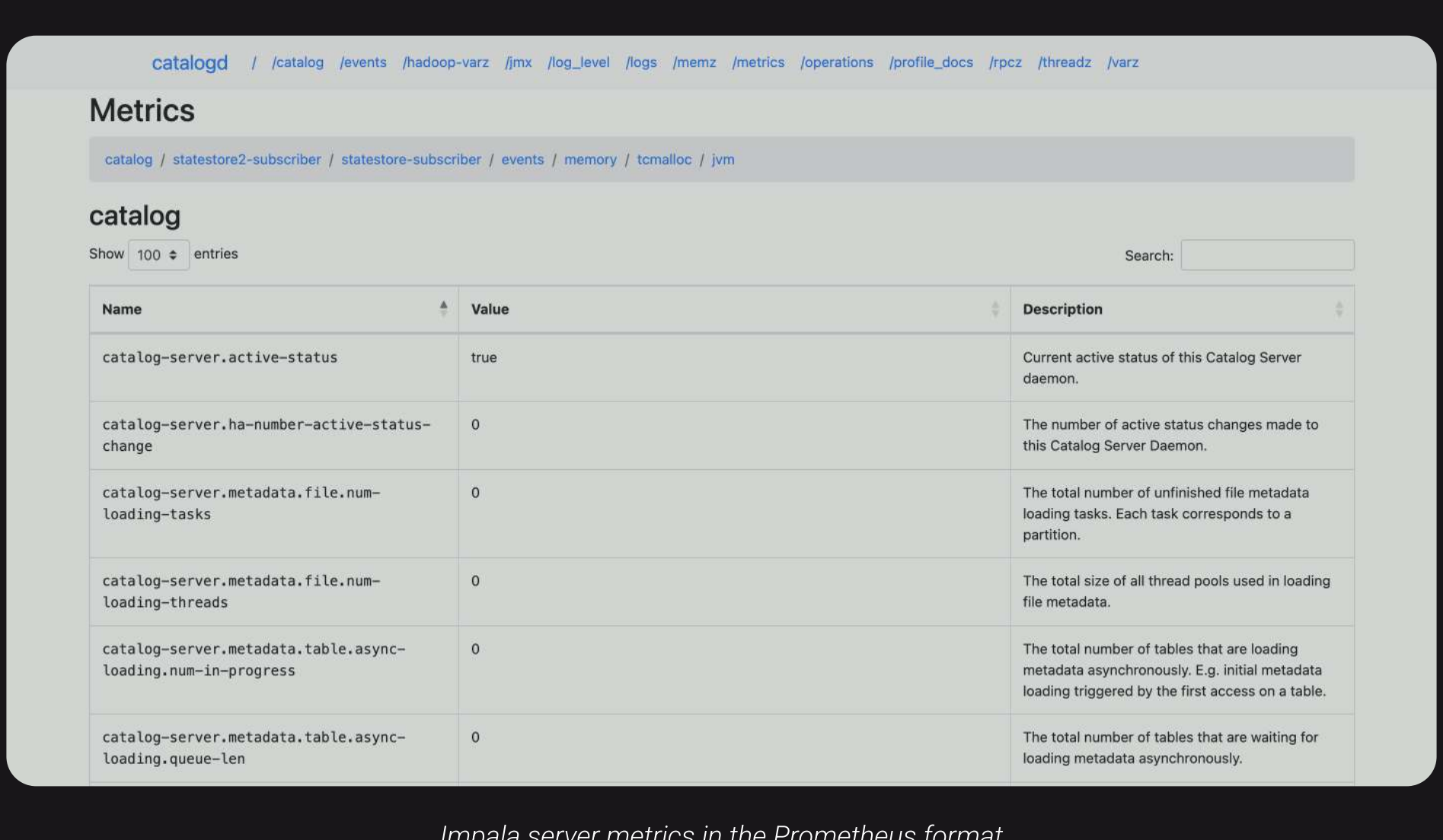
Step 5. View results

The Monitoring service starts automatically after the installation. To ensure that monitoring works correctly, check that both system and ADH service metrics are collected from the cluster hosts. To do this, view metrics in the Prometheus format in the browser or use the Prometheus or Grafana web interface.

Metrics in the Prometheus format

1. In the address bar of your browser, enter an address of an ADH host with the specified port and endpoint to listen for service or system metrics. Port numbers and endpoints are defined by the parameters located in:
   - the configuration pages of the ADH services — settings for accessing monitoring metrics of these services;
   - the ***Node Exporter settings*** section on the configuration page of the Monitoring service — settings for accessing system metrics of the ADH cluster hosts.

   For example, *http://10.92.42.57:25020/metrics* is an address to view metrics of the Impala server on a host with IP `10.92.42.57`.
2. The page that opens will display monitoring metrics from the specified host of the ADH cluster in the Prometheus format.



catalogd
/ /catalog /events /hadoop-varz /jmx /log_level /logs /memz /metrics /operations /profile_docs /rpcz /threadz /varz
catalog / statestore2-subscriber / statestore-subscriber / events / memory / tcmalloc / jvm
Show 100 entries
Search:

Metrics

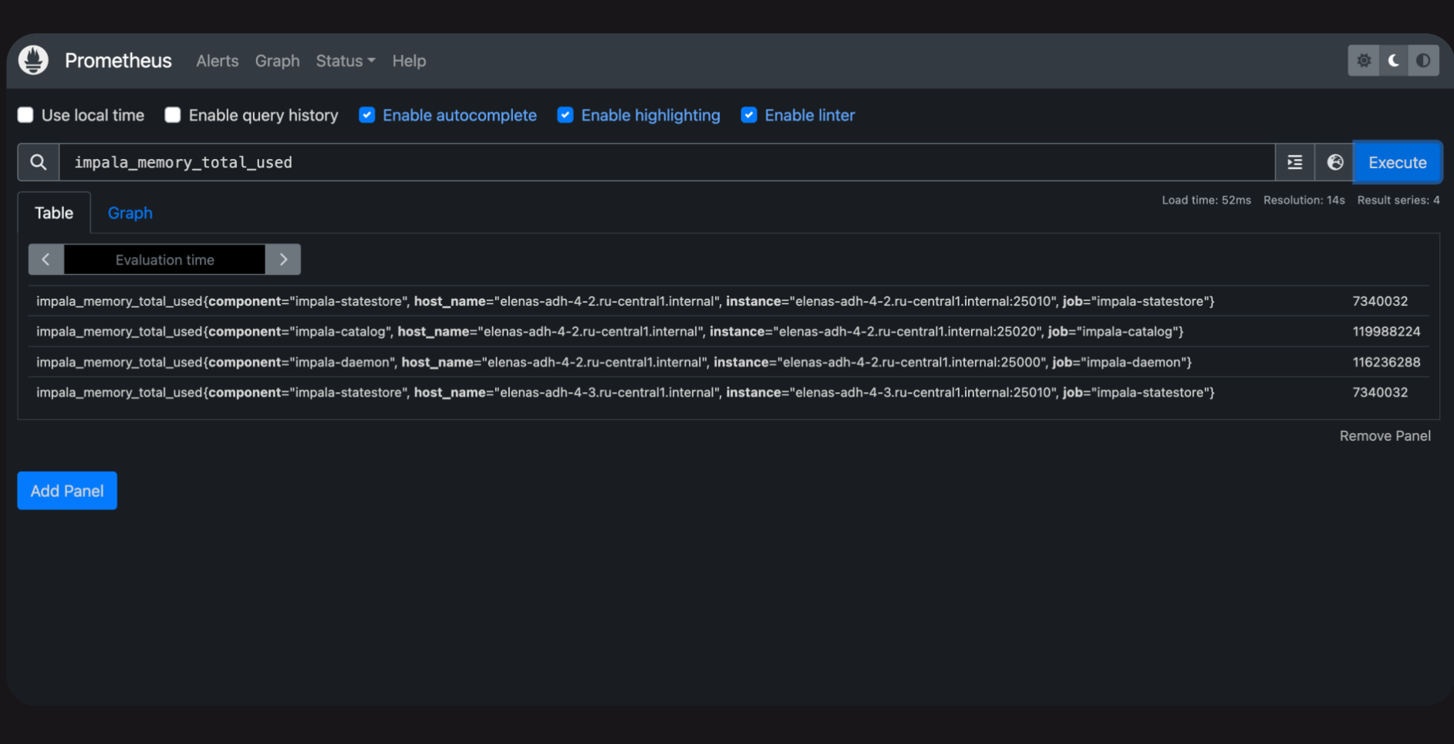
catalog

Name	Value	Description
catalog-server.active-status	true	Current active status of this Catalog Server daemon.
catalog-server.ha-number-active-status-change	0	The number of active status changes made to this Catalog Server Daemon.
catalog-server.metadata.file.num-loading-tasks	0	The total number of unfinished file metadata loading tasks. Each task corresponds to a partition.
catalog-server.metadata.file.num-loading-threads	0	The total size of all thread pools used in loading file metadata.
catalog-server.metadata.table.async-loading.num-in-progress	0	The total number of tables that are loading metadata asynchronously. E.g. initial metadata loading triggered by the first access on a table.
catalog-server.metadata.table.async-loading.queue-len	0	The total number of tables that are waiting for loading metadata asynchronously.

Impala server metrics in the Prometheus format

Prometheus web interface

1. In the address bar of your browser, enter an IP address of a host where the Prometheus Server component is installed. Specify also a port number from the ***web.listen-address*** parameter in the ***Service parameters*** group within the ***Prometheus settings*** section of the Monitoring service configuration (the default port is `11200`). For example: *http://10.92.40.107:11200*. To log into the Prometheus interface, use a username and password that are also specified in the ***Prometheus settings*** section — the ***Prometheus users to login/logout to Prometheus*** setting.
2. In the ***Expression*** field, you can enter a metric and click ***Execute*** — values of this metric on all hosts of the ADH cluster will be shown in the interface.



Prometheus
Alerts
Graph
Status
Help
Execute
Table
Graph
Evaluation time
Load time: 52ms Resolution: 14s Result series: 4
Remove Panel
Add Panel

☐ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

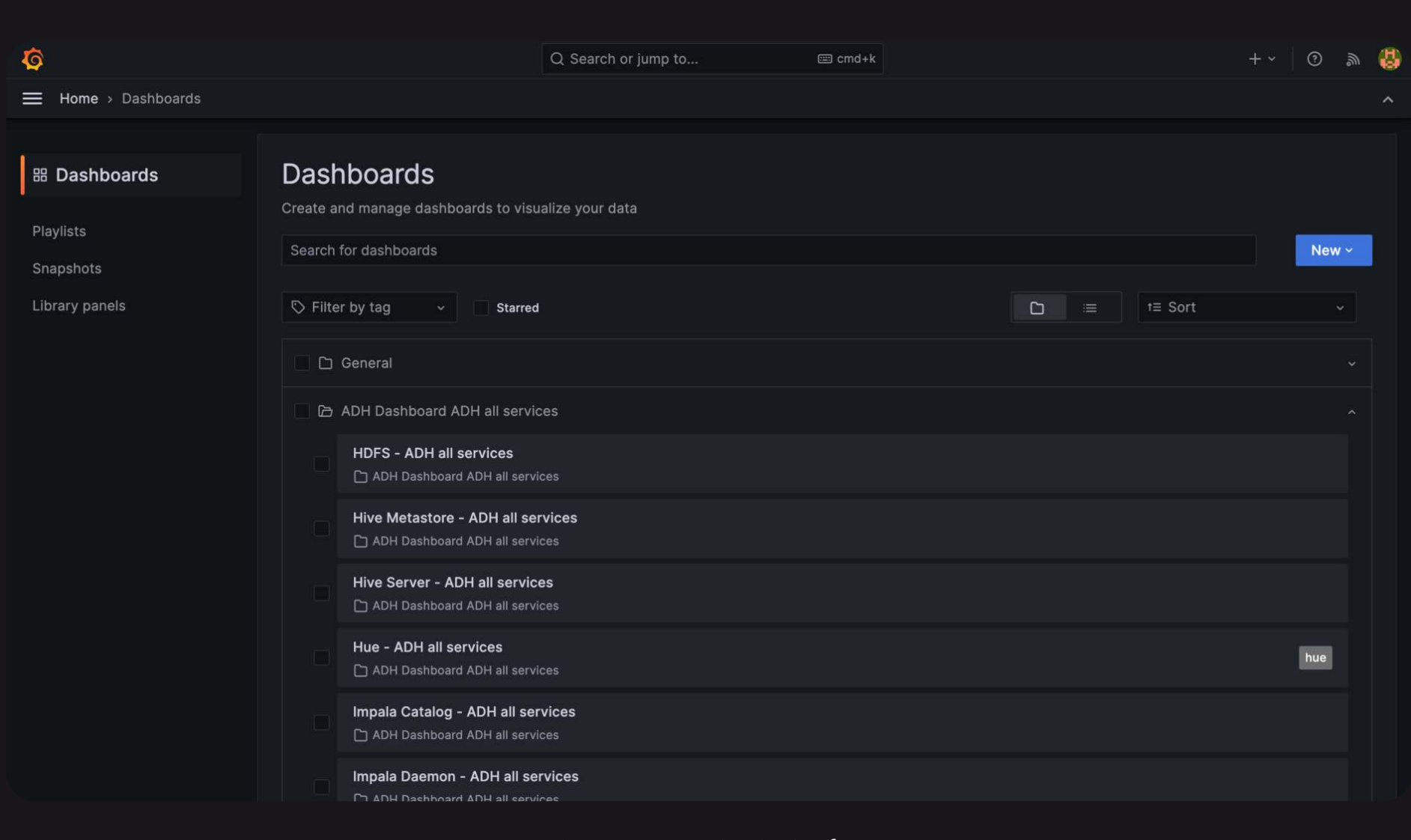
`impala_memory_total_used`

```
impala_memory_total_used{component="impala-statestore", host_name="elenas-adh-4-2.ru-central1.internal", instance="elenas-adh-4-2.ru-central1.internal:25010", job="impala-statestore"}   7340032
impala_memory_total_used{component="impala-catalog", host_name="elenas-adh-4-2.ru-central1.internal", instance="elenas-adh-4-2.ru-central1.internal:25020", job="impala-catalog"}   119988224
impala_memory_total_used{component="impala-daemon", host_name="elenas-adh-4-2.ru-central1.internal", instance="elenas-adh-4-2.ru-central1.internal:25000", job="impala-daemon"}   116236288
impala_memory_total_used{component="impala-statestore", host_name="elenas-adh-4-3.ru-central1.internal", instance="elenas-adh-4-3.ru-central1.internal:25010", job="impala-statestore"}   7340032
```

Prometheus web interface

Grafana web interface

1. In the address bar of your browser, enter an address of a host on which Grafana is deployed and add a port number — a value of the ***Grafana listen port*** parameter located in the ***Grafana settings*** section of the Monitoring service configuration (the default value is `11210`). For example, *http://10.92.40.107:11210*. To log in, use `admin` as a username, and the ***Grafana administrator's password*** parameter value (also found in the ***Grafana settings*** section of service configuration parameters) as a password.
2. In the window that opens, navigate to ***Home*** → ***Dashboards*** and expand the ***ADH Dashboard <ADH_cluster_name>*** section. In this section, you can select one of available dashboards to view service or system metrics coming from your ADH cluster.



Search or jump to...
cmd+k
Home › Dashboards
Dashboards
Playlists
Snapshots
Library panels
Create and manage dashboards to visualize your data
Search for dashboards
New
Filter by tag
Starred
Sort
General
hue

ADH Dashboard ADH all services
- HDFS - ADH all services
- Hive Metastore - ADH all services
- Hive Server - ADH all services
- Hue - ADH all services
- Impala Catalog - ADH all services
- Impala Daemon - ADH all services

View ADH metrics in Grafana

Monitoring cluster

Elena Kostyuchenko

To Table of Contents

To set up monitoring for ADH, you can create a monitoring cluster with the Diamond, Graphite and Grafana services installed, and integrate it with the ADH cluster.

For **online** installation, you need a separate monitoring cluster. The steps for installing monitoring in this case are listed below. You can perform them both before and after the ADH cluster installation, except the last step Integrate with ADH cluster, that can be executed only when the ADH cluster already exists.

- Create a cluster
- Add services
- Add hosts to a cluster
- Add components
- Configure services
- Install a cluster
- Integrate with ADH cluster

Create a cluster

Daria Barysheva

Contents

To Table of Contents

Step 1. Download a cluster bundle
Step 2. Upload a cluster bundle to ADCM
Step 3. Create a cluster based on the uploaded bundle

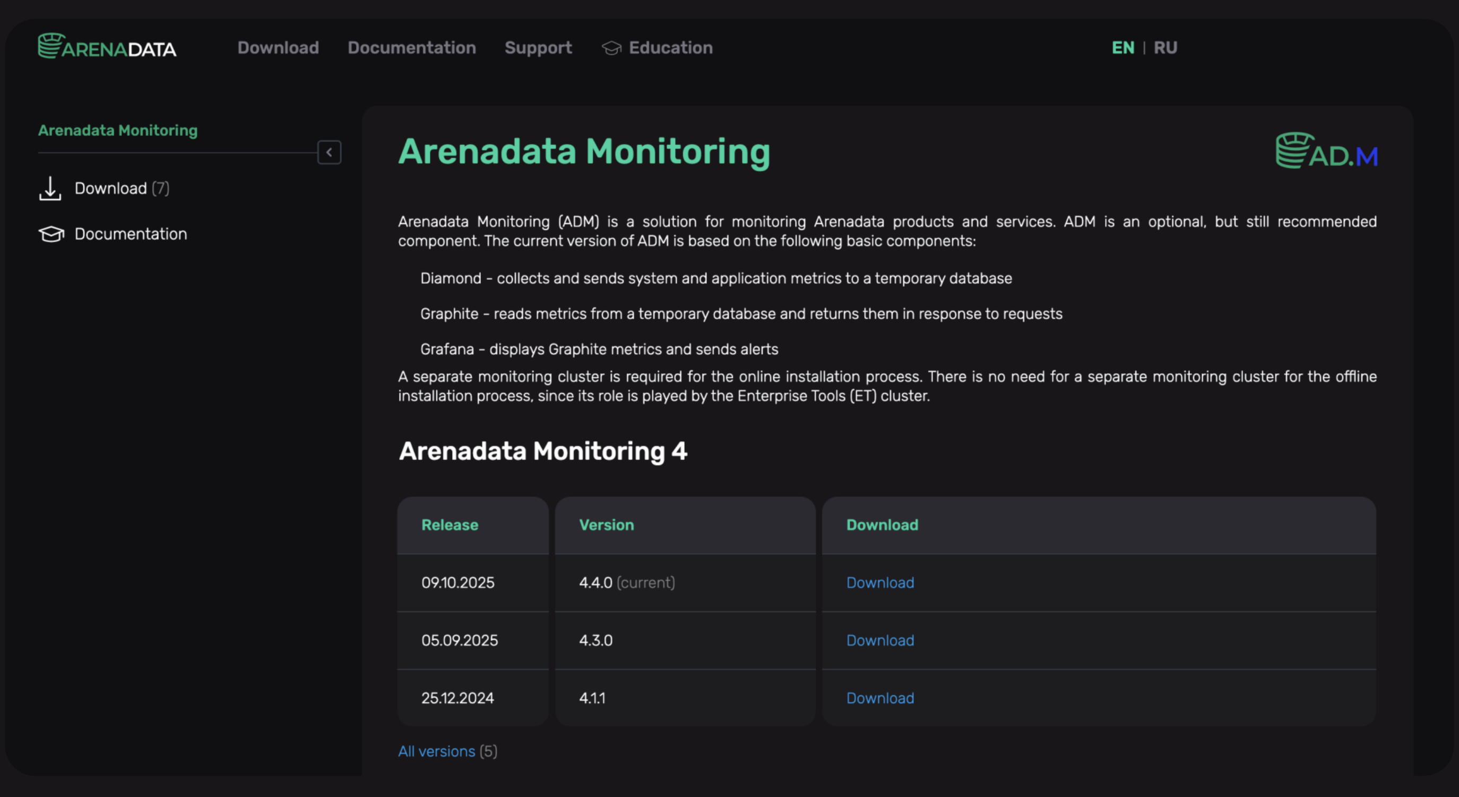
In ADCM a **cluster** means a set of services running on a set of hosts.

Clusters distributions for ADCM are made in the form of so-called bundles. Generally a bundle is a regular archive that includes a description and software logic of cluster deployment and exploitation.

The steps for adding a cluster to ADCM are listed below.

Step 1. Download a cluster bundle

1. Go to https://network.arenadata.io/ and select ***Arenadata Monitoring***.
2. Download a bundle of the desired version.



ARENADATA
Download
Documentation
Support
Education
EN | RU
Arenadata Monitoring
Download (7)
Documentation
AD.M

Arenadata Monitoring

Arenadata Monitoring (ADM) is a solution for monitoring Arenadata products and services. ADM is an optional, but still recommended component. The current version of ADM is based on the following basic components:

Diamond - collects and sends system and application metrics to a temporary database

Graphite - reads metrics from a temporary database and returns them in response to requests

Grafana - displays Graphite metrics and sends alerts

A separate monitoring cluster is required for the online installation process. There is no need for a separate monitoring cluster for the offline installation process, since its role is played by the Enterprise Tools (ET) cluster.

Arenadata Monitoring 4

Release	Version	Download
09.10.2025	4.4.0 (current)	Download
05.09.2025	4.3.0	Download
25.12.2024	4.1.1	Download

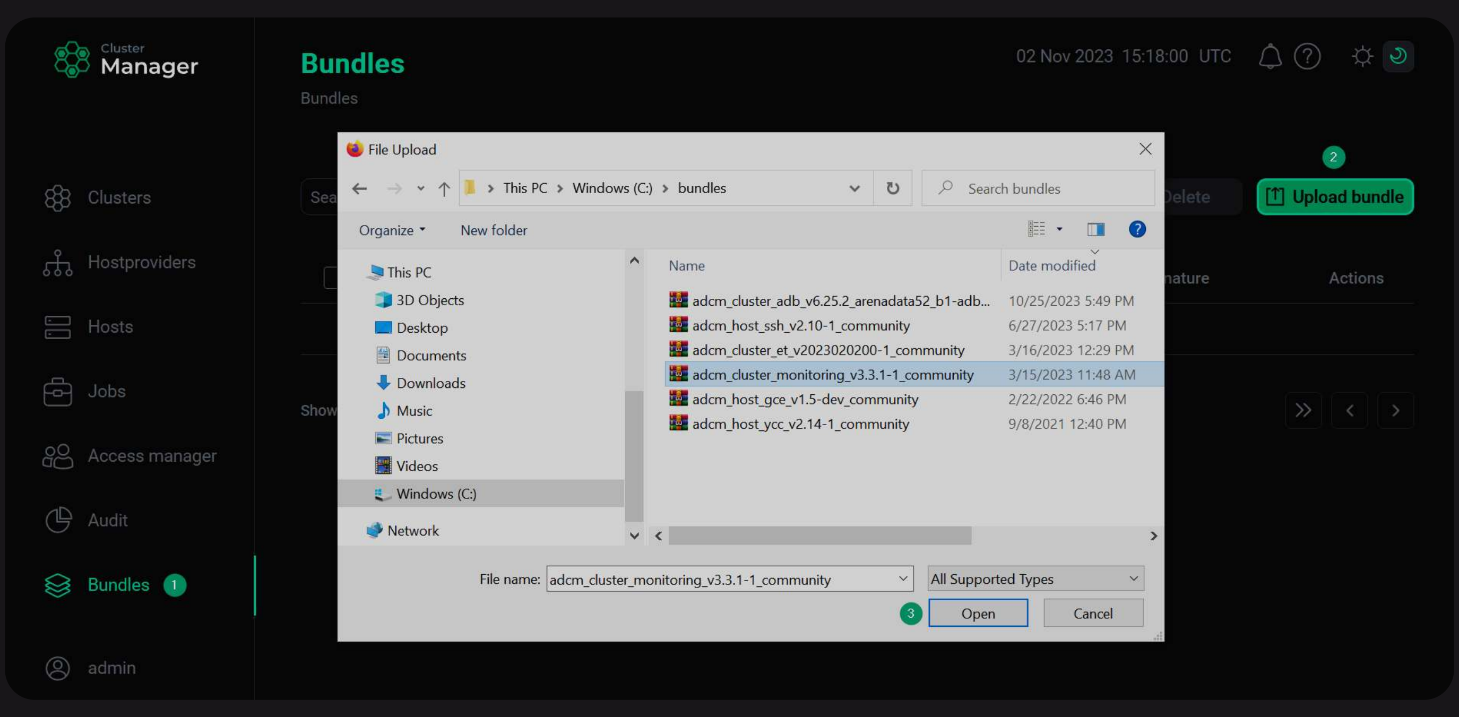
All versions (5)

Arenadata Monitoring download page

Step 2. Upload a cluster bundle to ADCM

To upload a bundle to ADCM, follow the steps:

1. Select the ***Bundles*** item in the left navigation menu and click ***Upload bundle***.
2. Select a bundle in the Open File Dialog.



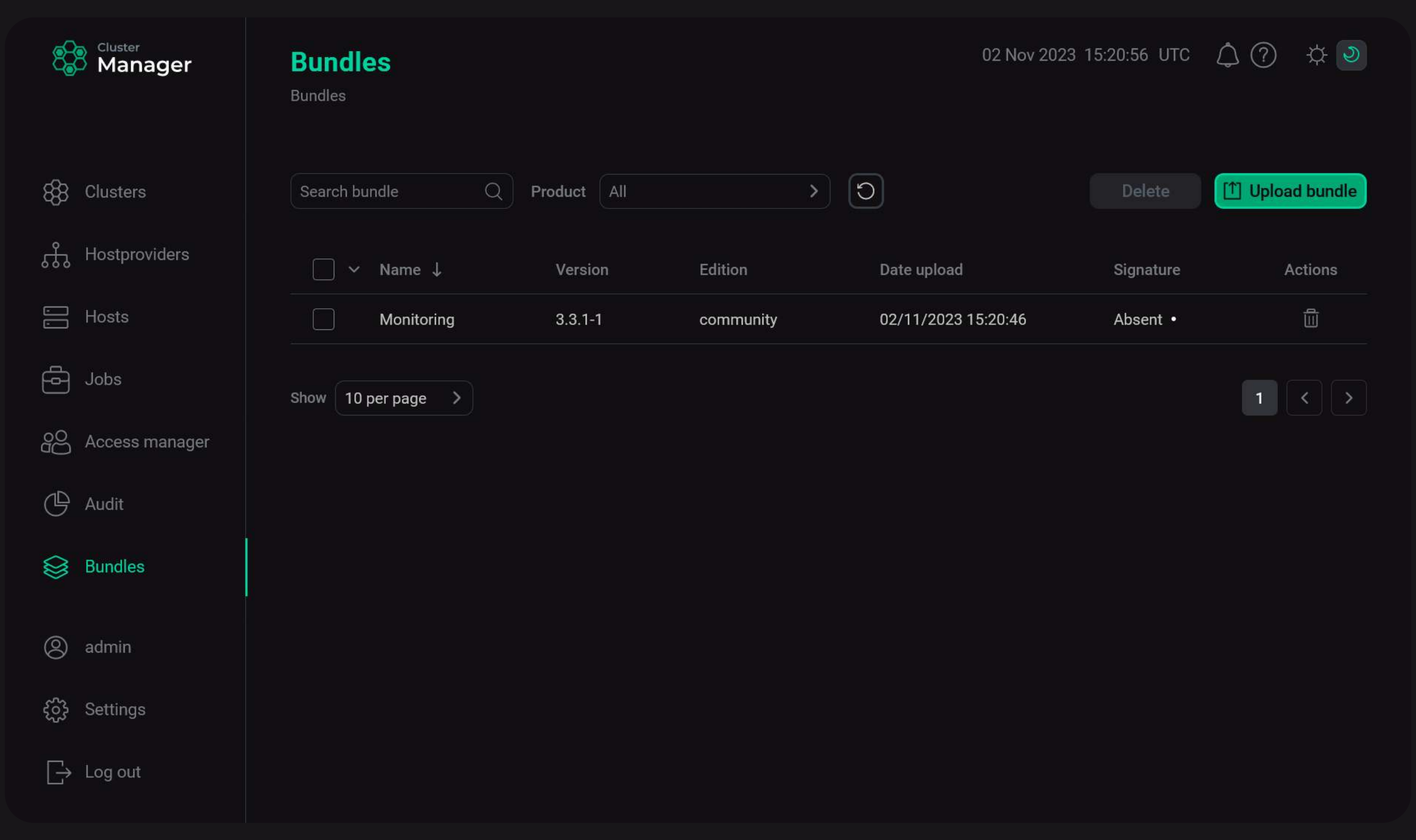
Cluster
Manager
Bundles
Bundles
02 Nov 2023 15:18:00 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
1
admin
File Upload
This PC > Windows (C:) > bundles
Search bundles
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Network
2
Upload bundle
Delete
Actions
File name:
All Supported Types
3
Open
Cancel

Name	Date modified
adcm_cluster_adb_v6.25.2_arenadata52_b1-adb...	10/25/2023 5:49 PM
adcm_host_ssh_v2.10-1_community	6/27/2023 5:17 PM
adcm_cluster_et_v2023020200-1_community	3/16/2023 12:29 PM
adcm_cluster_monitoring_v3.3.1-1_community	3/15/2023 11:48 AM
adcm_host_gce_v1.5-dev_community	2/22/2022 6:46 PM
adcm_host_ycc_v2.14-1_community	9/8/2021 12:40 PM

File name: adcm_cluster_monitoring_v3.3.1-1_community

Upload a bundle

3. As a result of the performed actions, a bundle is displayed on the ***Bundles*** page.



Cluster
Manager
Bundles
Bundles
02 Nov 2023 15:20:56 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search bundle
Product
All
Delete
Upload bundle
Show
10 per page
1

Name	Version	Edition	Date upload	Signature	Actions
Monitoring	3.3.1-1	community	02/11/2023 15:20:46	Absent	

The result of successful uploading a bundle

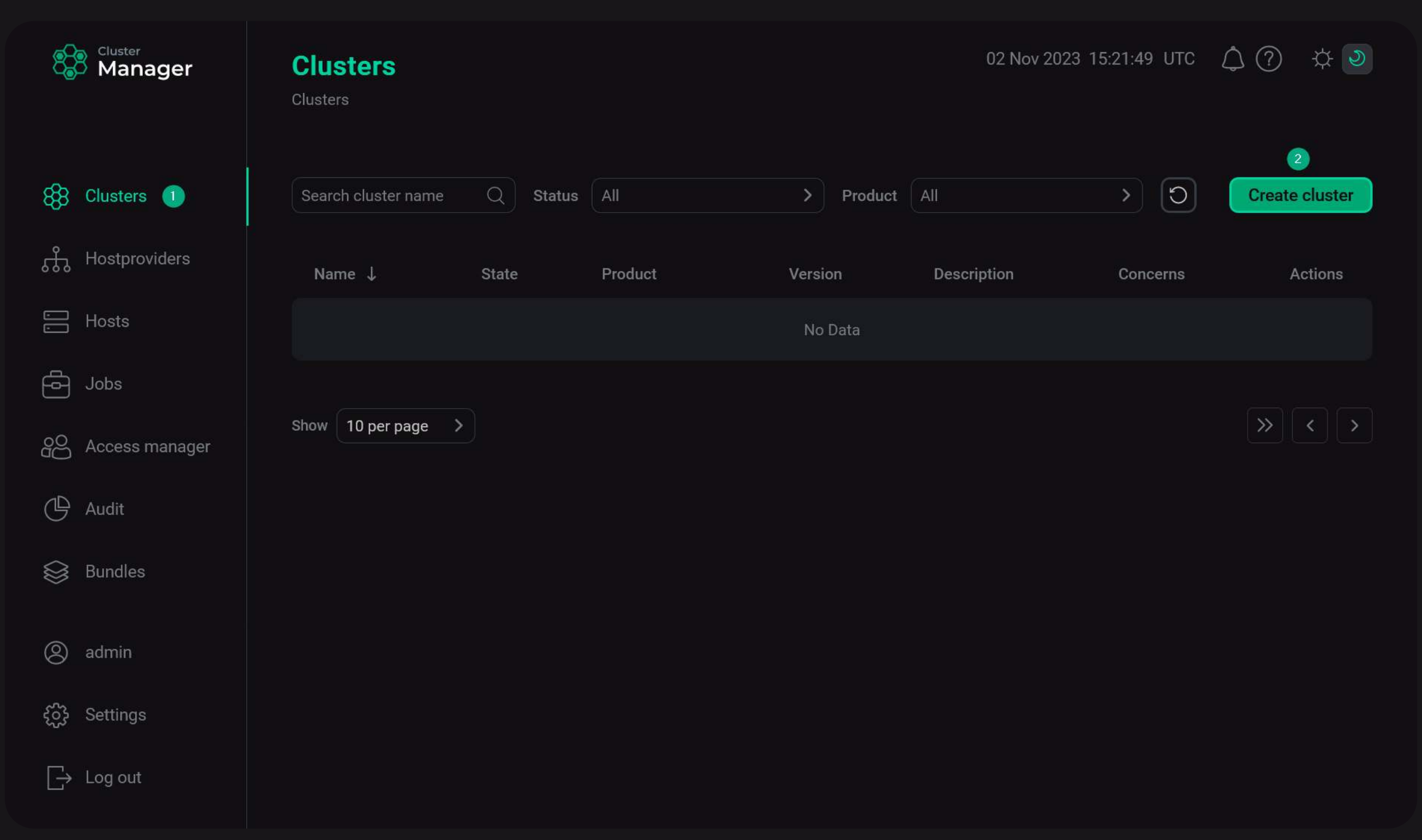
Step 3. Create a cluster based on the uploaded bundle

IMPORTANT

When creating a cluster via the ADCM web interface, you generate a new ADCM cluster instance. It means only adding the cluster to ADCM — at this stage no services are deployed.

The steps for cluster creation are given below:

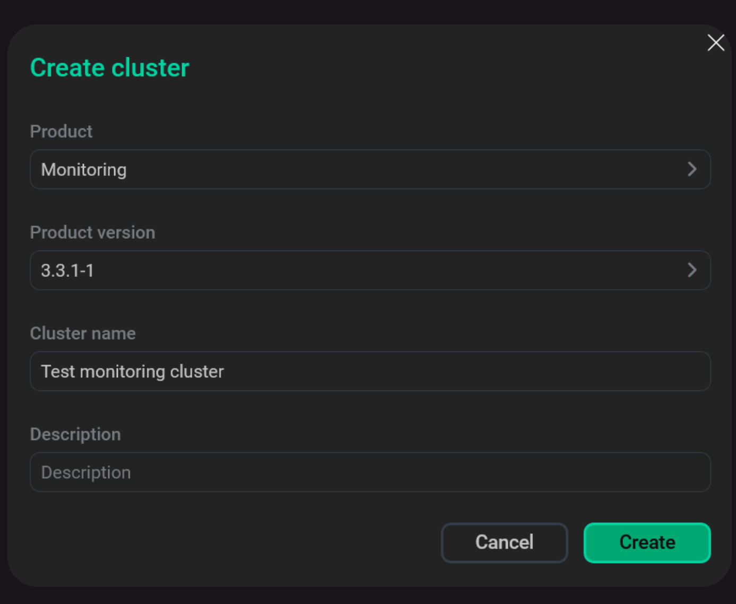
1. Select the ***Clusters*** item in the left navigation menu and click ***Create cluster***.



Cluster
Manager
Clusters
Clusters
02 Nov 2023 15:21:49 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
2
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
No Data
Show
10 per page

Switch to cluster creation

2. In the opened dialog, describe a new cluster instance:
   - Select an uploaded cluster bundle in the ***Product*** field.
   - Select a bundle version in the ***Product version*** field. Several versions become available in the case of different versions of the same bundle being uploaded.
   - Enter a cluster name in the ***Cluster name*** field.
   - Enter a cluster description in the ***Description*** field if necessary.
   - Set the ***I accept Terms of Agreement*** flag to sign the license agreement. This flag is displayed only for Enterprise bundles. To read the agreement text, refer to the ***Terms of Agreement*** link. You can also sign the license agreement on the ***Bundles*** page by clicking the desired product bundle.
   - Click ***Create***.



Create cluster
Description
Cancel
Create

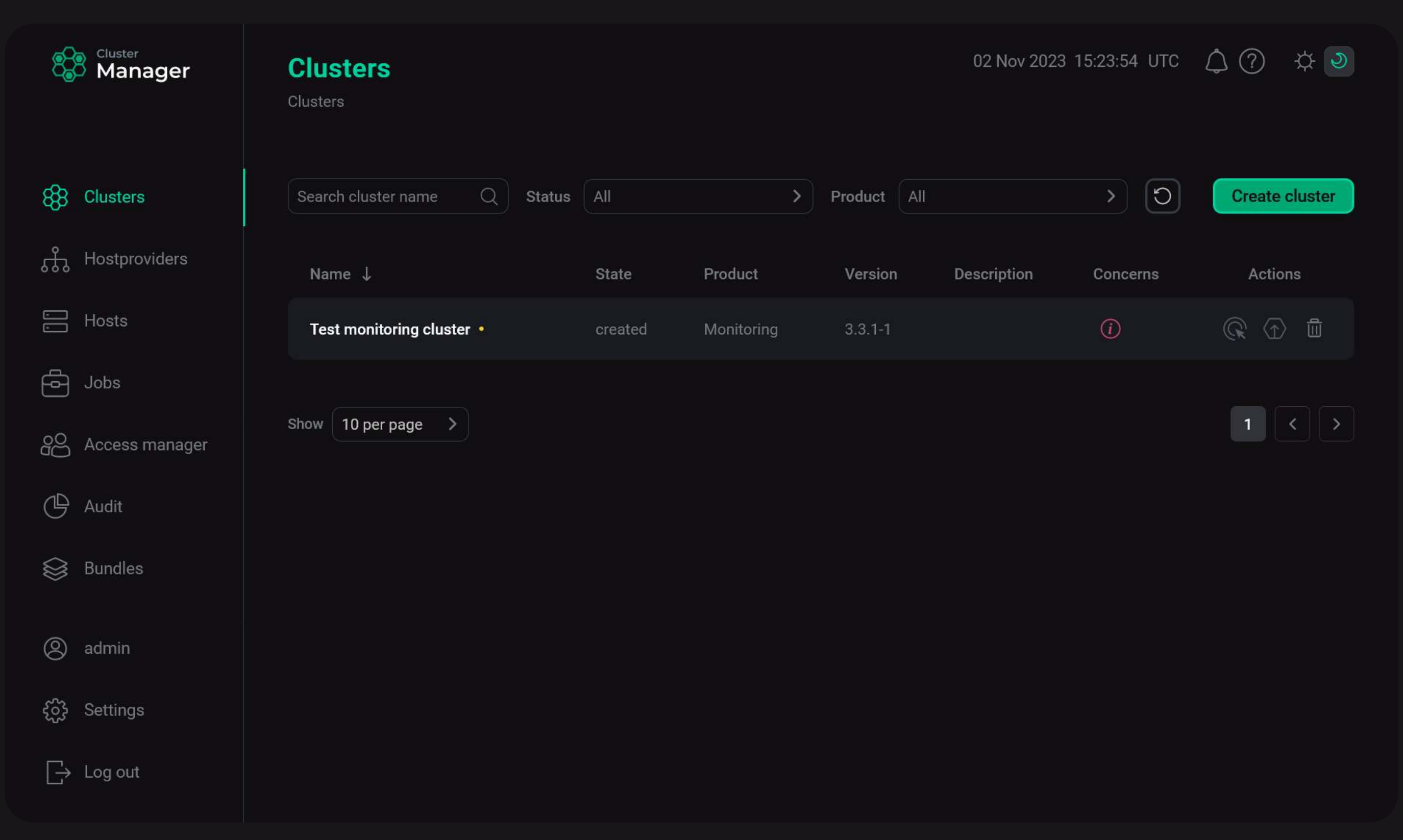
Product: Monitoring

Product version: 3.3.1-1

Cluster name: Test monitoring cluster

Create a cluster

3. As a result, the created cluster instance is displayed on the ***Clusters*** page.



Cluster
Manager
Clusters
Clusters
02 Nov 2023 15:23:54 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test monitoring cluster	created	Monitoring	3.3.1-1			

The result of successful cluster creation

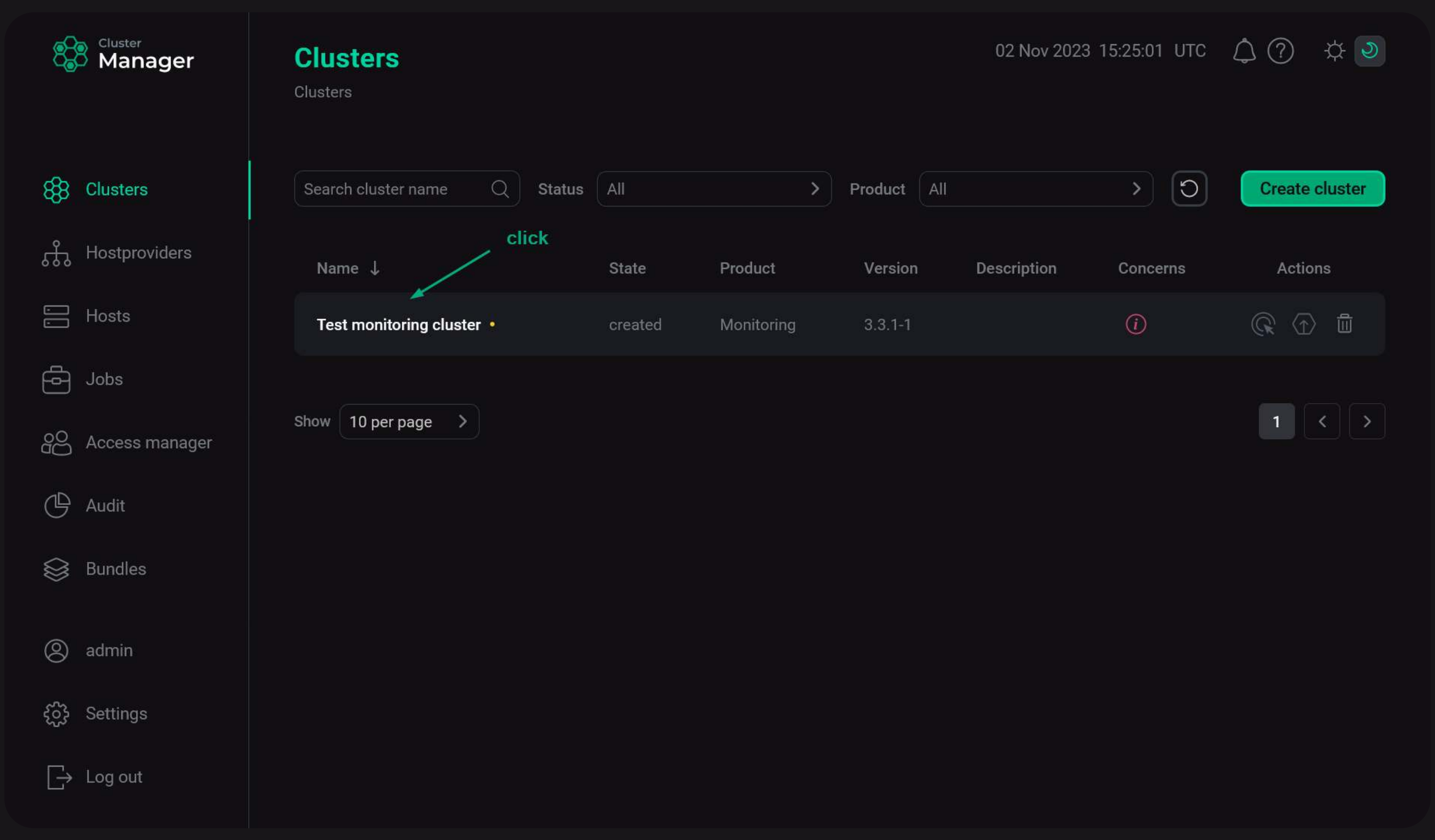
Add services

Daria Barysheva

To Table of Contents

In ADCM a **service** means a software that performs some function. The steps for adding services to a cluster are listed below:

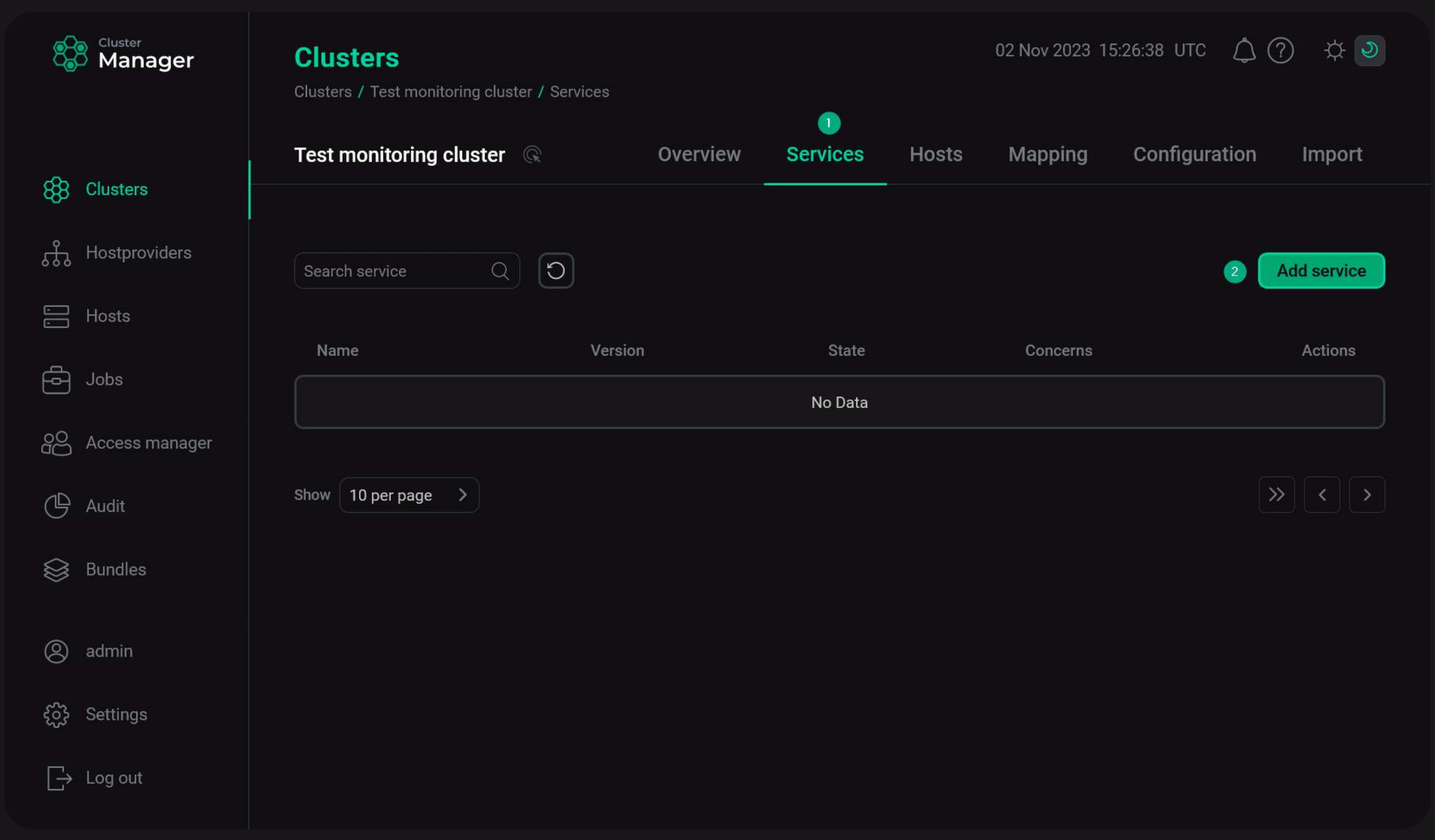
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Nov 2023 15:25:01 UTC
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test monitoring cluster	created	Monitoring	3.3.1-1			

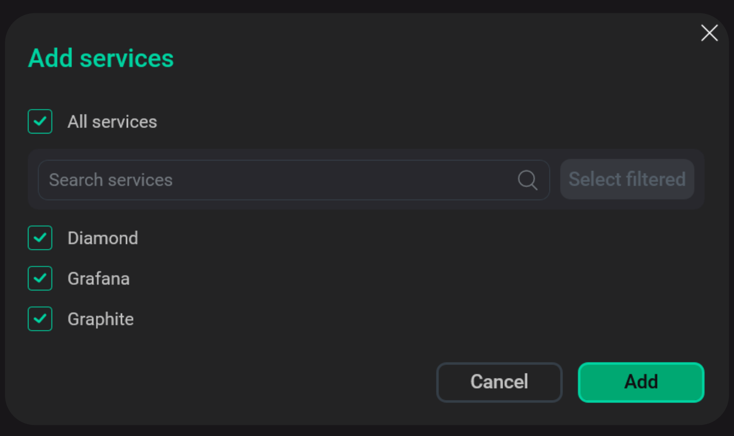
Select a cluster

2. Open the ***Services*** tab on the cluster page and click ***Add service***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test monitoring cluster / Services
02 Nov 2023 15:26:38 UTC
Test monitoring cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Switch to adding services

3. In the opened dialog, select services that should be added to the cluster and click ***Add***.


Add services
☑ All services
Search services
Select filtered
☑ Diamond
☑ Grafana
☑ Graphite
Cancel
Add

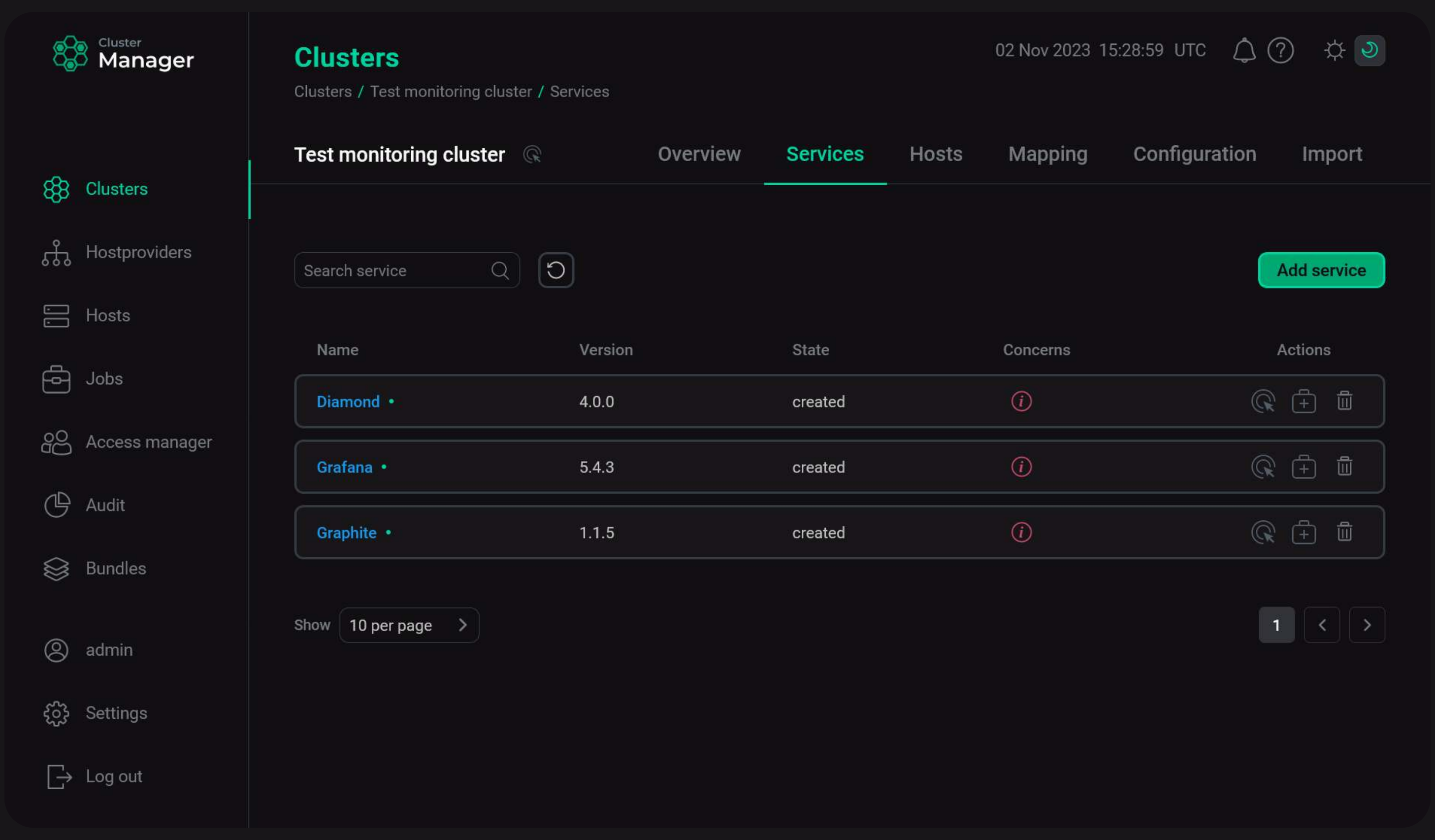
Select services

The brief description of available services is listed below.

Services that should be added to the monitoring cluster

Service	Purpose
Diamond	A Python service used for collecting system metrics from the hosts that are connected to the monitoring system. The collected metrics are processed by Carbon (the backend services of Graphite)
Graphite	Includes several components: ▪ **Carbon** — a set of services that are used to process metrics collected by Diamond. ▪ **Whisper DB** — a temporary file database where collected metrics are stored. ▪ **Web service** — allows to get and visualize metrics from Whisper DB.
Grafana	A web service that allows users to create convenient sets of graphs, alerts, and other visualization tools for metrics received from Graphite. By default, uses the port `3000`

4. As a result, the added services are displayed on the ***Services*** tab.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test monitoring cluster / Services
02 Nov 2023 15:28:59 UTC
Test monitoring cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
Diamond	4.0.0	created		
Grafana	5.4.3	created		
Graphite	1.1.5	created		

The result of successful adding services to a cluster

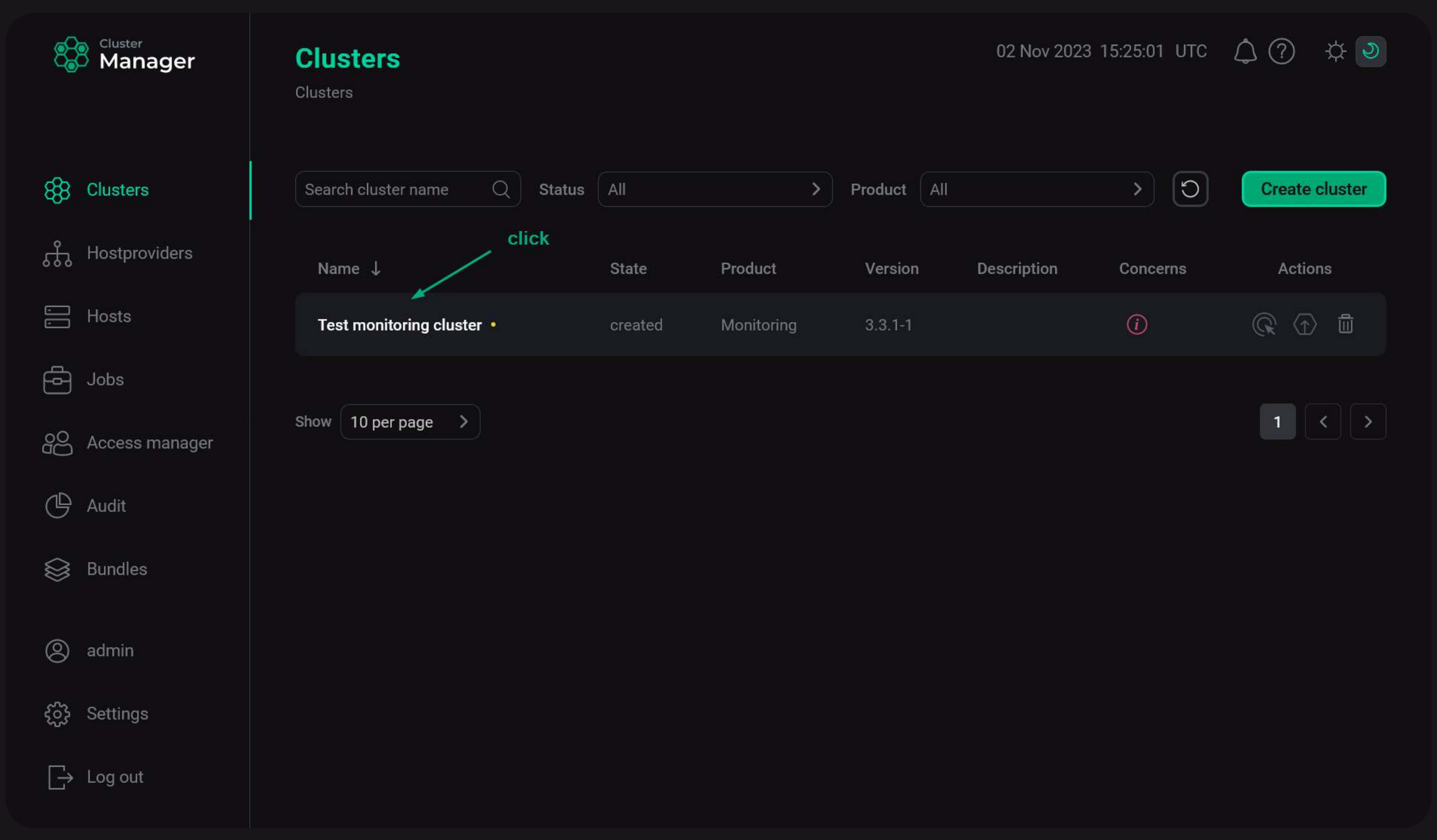
Add hosts to a cluster

Daria Barysheva

To Table of Contents

The steps for adding hosts to a cluster are listed below:

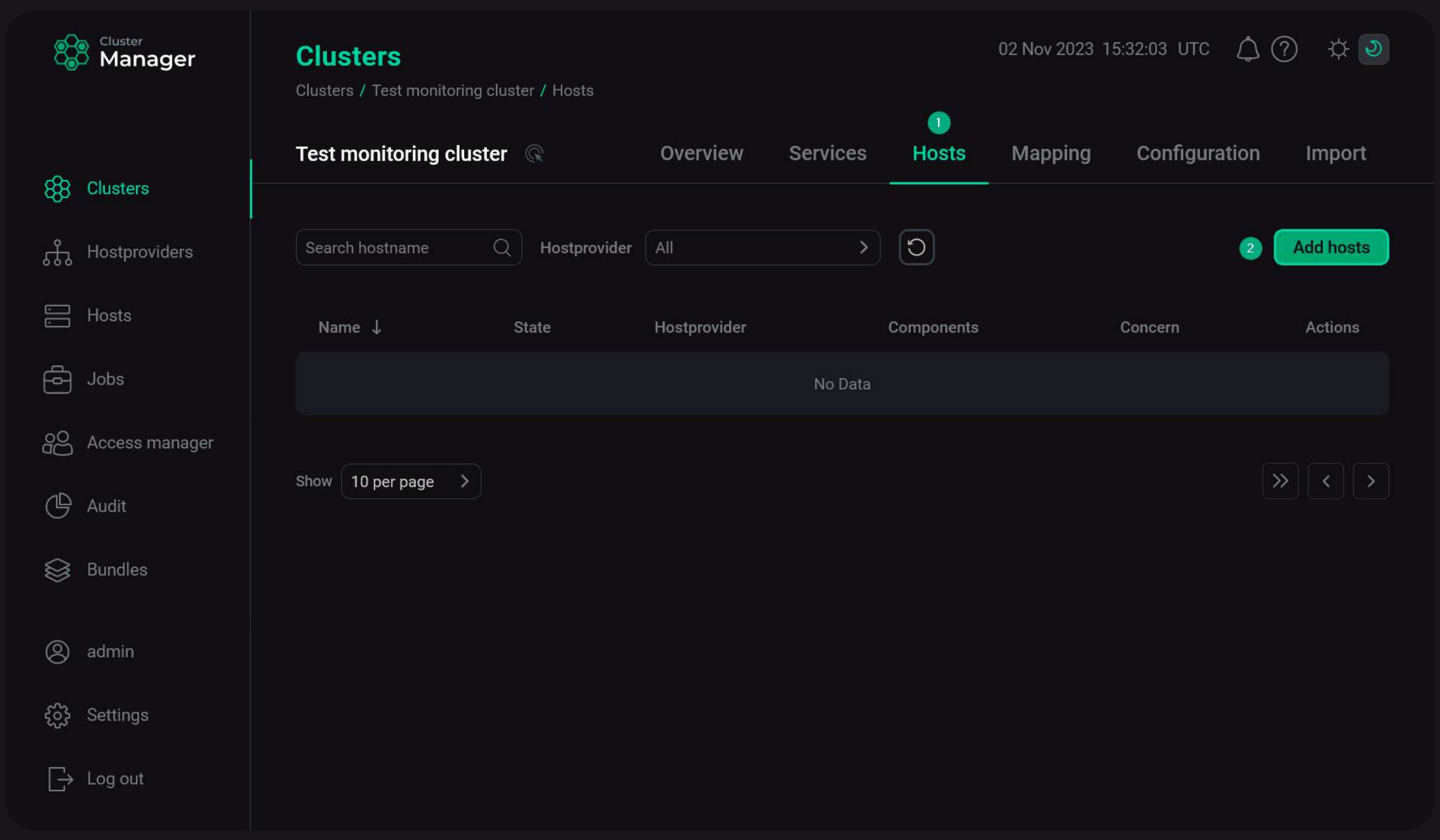
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Clusters
02 Nov 2023 15:25:01 UTC
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test monitoring cluster	created	Monitoring	3.3.1-1			

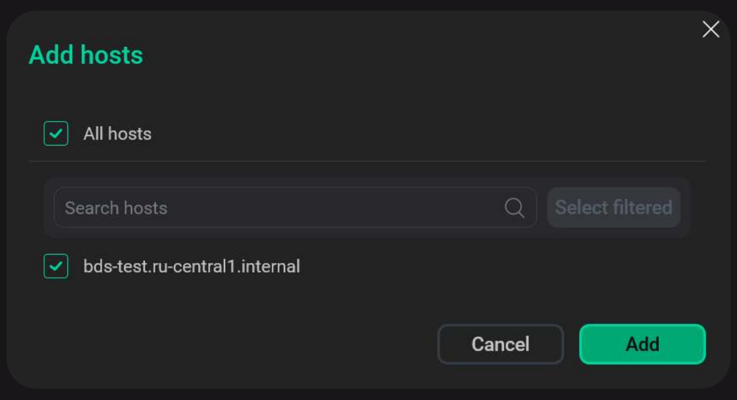
Select a cluster

2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.


Cluster Manager
Clusters
Clusters / Test monitoring cluster / Hosts
02 Nov 2023 15:32:03 UTC
Test monitoring cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
2
Add hosts
Name
State
Hostprovider
Components
Concern
Actions
No Data
Show
10 per page

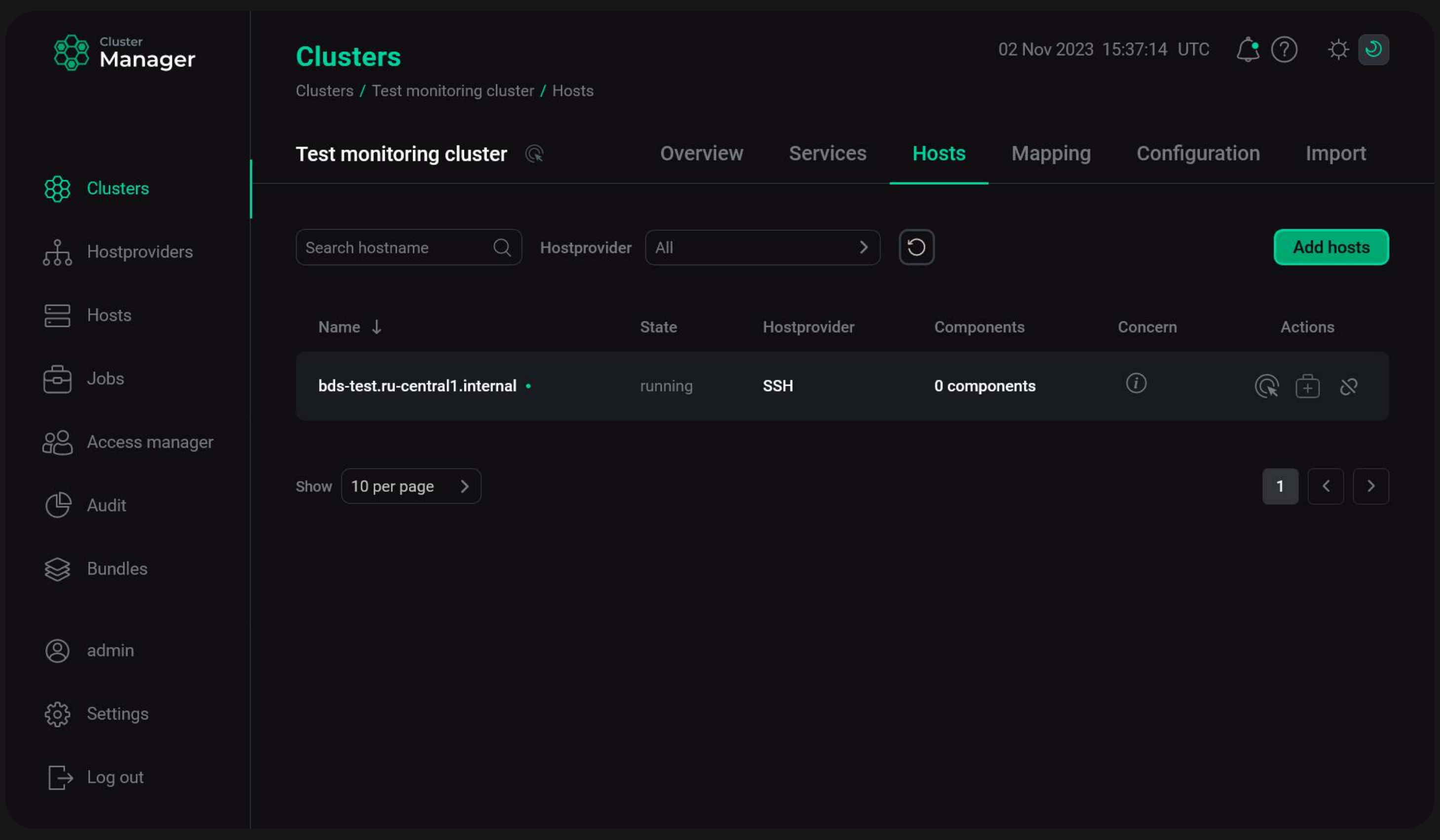
Switch to adding hosts

3. In the opened dialog, select hosts that should be added to the cluster and click ***Add***.


Add hosts
☑ All hosts
Search hosts
Select filtered
☑ bds-test.ru-central1.internal
Cancel
Add

Select hosts

4. As a result, the added hosts are displayed on the ***Hosts*** tab.


Cluster Manager
Clusters
Clusters / Test monitoring cluster / Hosts
02 Nov 2023 15:37:14 UTC
Test monitoring cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Add hosts
Show
10 per page
1

Name	State	Hostprovider	Components	Concern	Actions
bds-test.ru-central1.internal	running	SSH	0 components		

The result of successful adding hosts to a cluster

Add components

Daria Barysheva

To Table of Contents

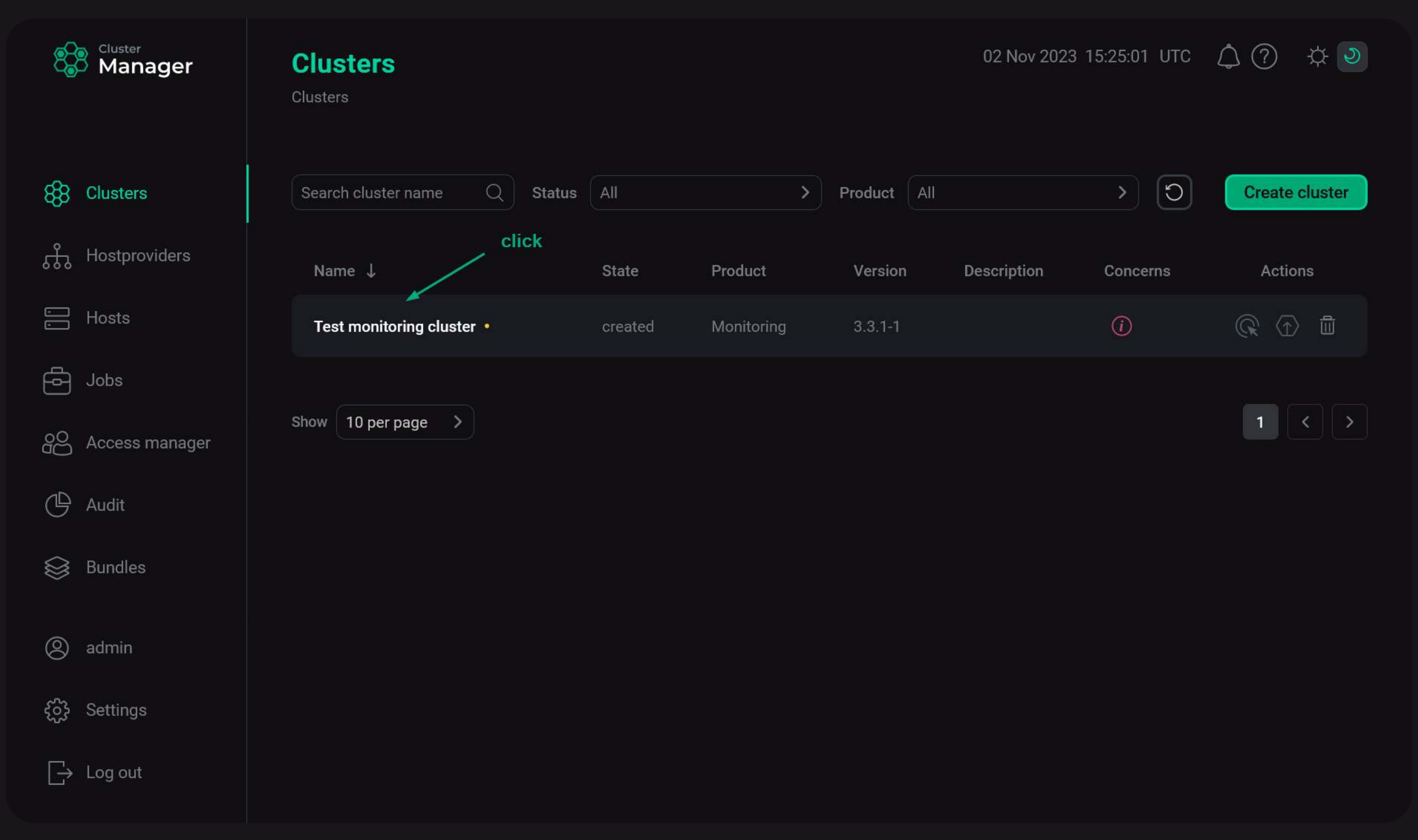
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts.

IMPORTANT

Initially, there are no components on any of cluster hosts. You should map components to hosts manually.

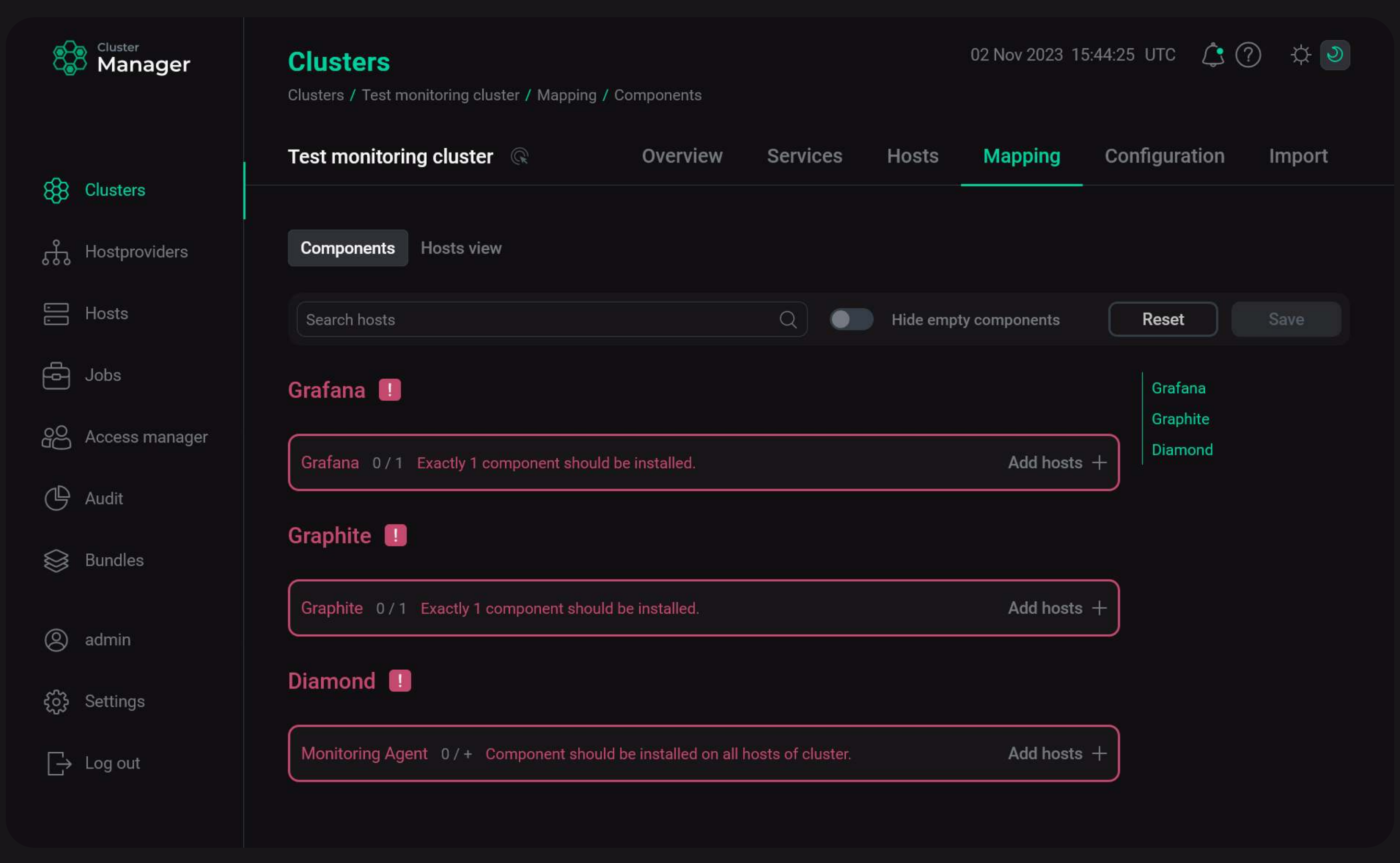
The steps for adding components to hosts are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Clusters
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
02 Nov 2023 15:25:01 UTC
Search cluster name
Status All
Product All
Create cluster
click
Name
State
Product
Version
Description
Concerns
Actions
Test monitoring cluster
created
Monitoring
3.3.1-1
Show 10 per page
1

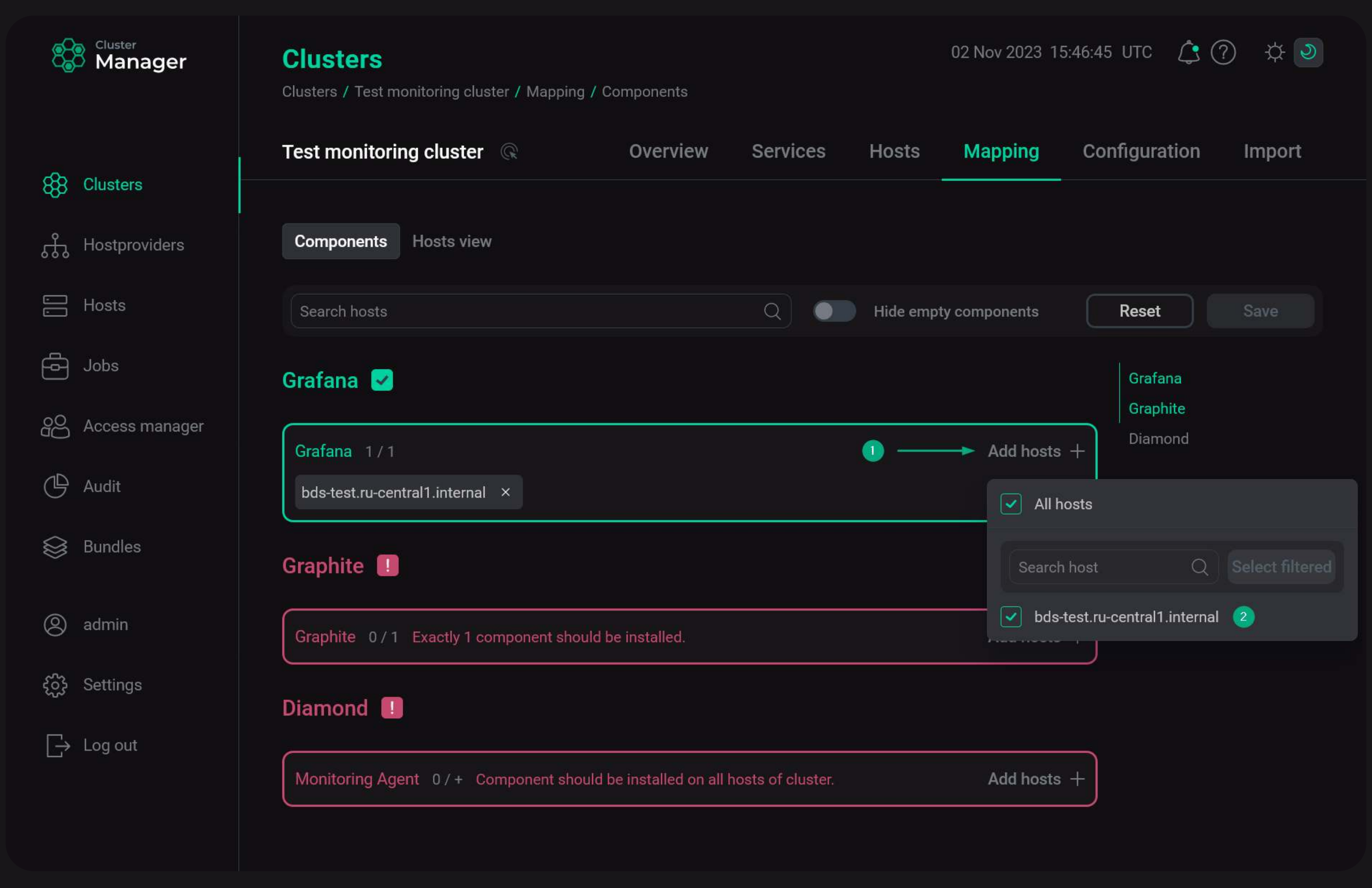
Select a cluster

2. Open the ***Mapping*** tab on the cluster page.


Clusters
Clusters / Test monitoring cluster / Mapping / Components
02 Nov 2023 15:44:25 UTC
Test monitoring cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
Grafana
Grafana 0 / 1 Exactly 1 component should be installed
Add hosts
Graphite
Graphite 0 / 1 Exactly 1 component should be installed
Add hosts
Diamond
Monitoring Agent 0 / + Component should be installed on all hosts of cluster
Add hosts
Grafana
Graphite
Diamond

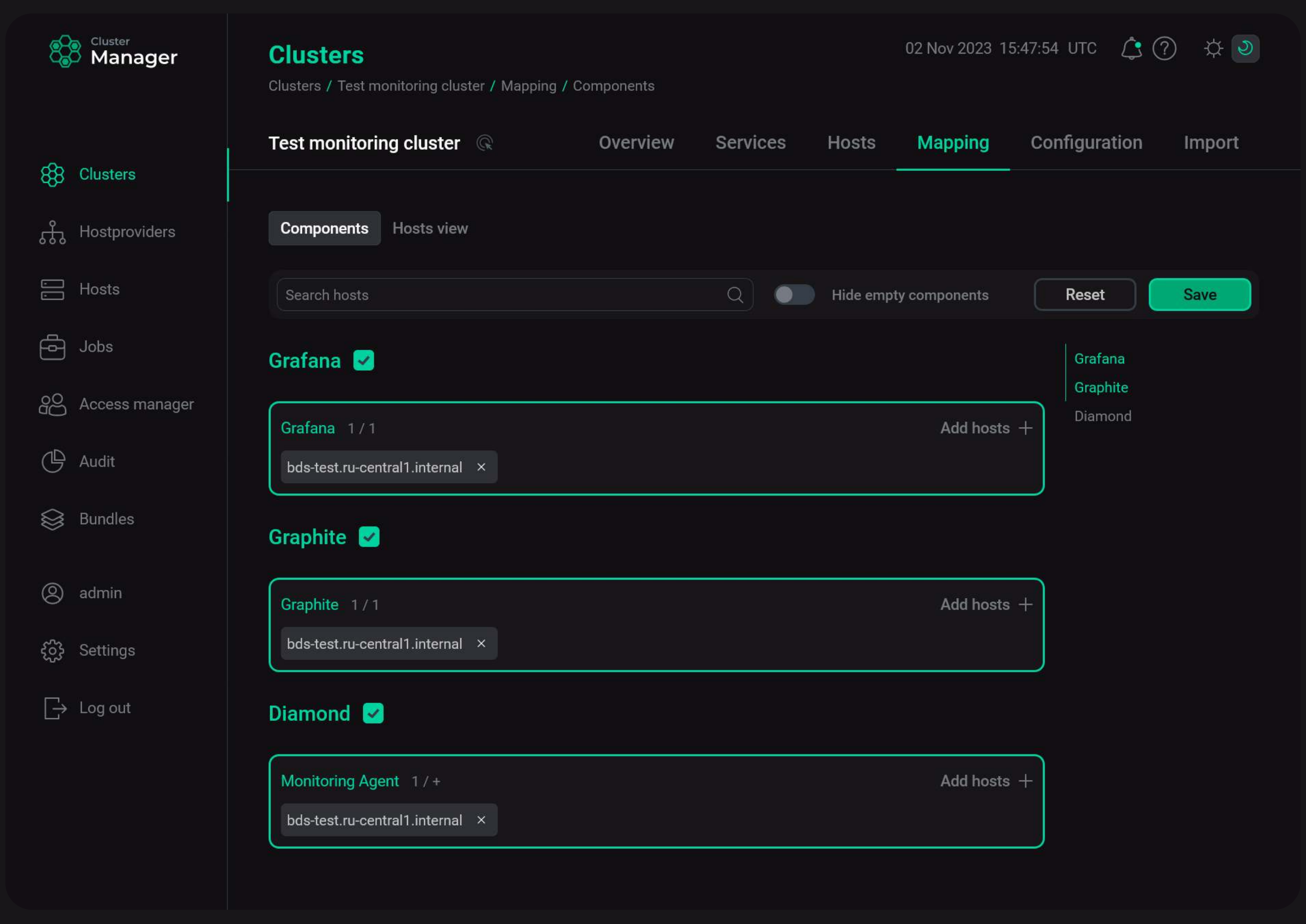
Switch to mapping service components

3. In the window that opens, map components to hosts. To assign hosts for each component, click ***Add hosts*** and select hosts in the pop-up window.


02 Nov 2023 15:46:45 UTC
Grafana
Grafana 1 / 1
bds-test.ru-central1.internal
Add hosts
All hosts
Search host
Select filtered
bds-test.ru-central1.internal
Graphite
Graphite 0 / 1 Exactly 1 component should be installed
Diamond
Monitoring Agent 0 / + Component should be installed on all hosts of cluster
Add hosts

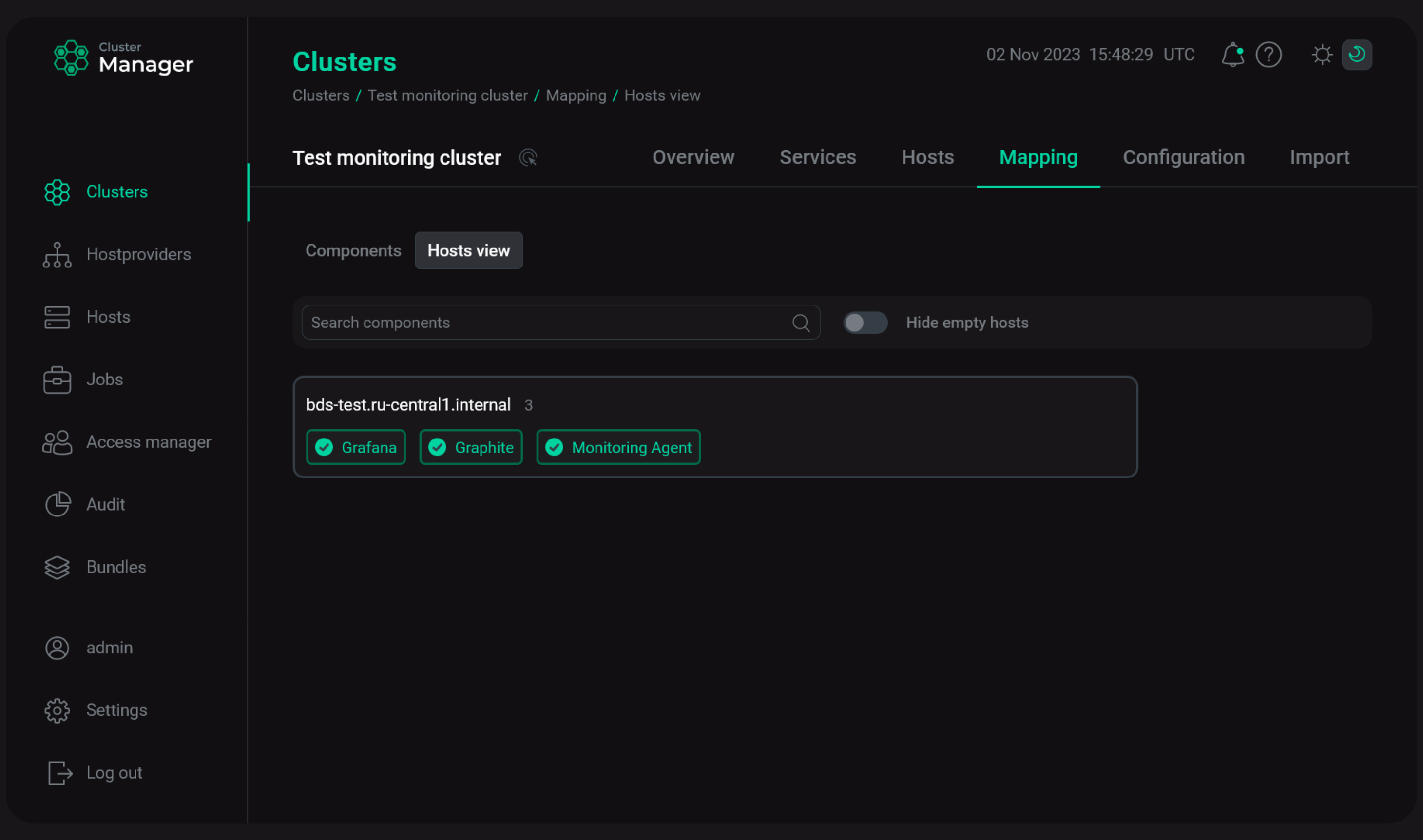
Select hosts for a component

4. After component distribution is completed, click ***Save***.


02 Nov 2023 15:47:54 UTC
Reset
Save
Grafana
Grafana 1 / 1
bds-test.ru-central1.internal
Graphite
Graphite 1 / 1
bds-test.ru-central1.internal
Diamond
Monitoring Agent 1 / +
bds-test.ru-central1.internal
Add hosts

Save mapping of components

5. To open an alternative mapping view (by hosts), select the ***Hosts view*** tab.


Clusters / Test monitoring cluster / Mapping / Hosts view
02 Nov 2023 15:48:29 UTC
Components
Hosts view
Search components
Hide empty hosts
bds-test.ru-central1.internal 3
Grafana
Graphite
Monitoring Agent

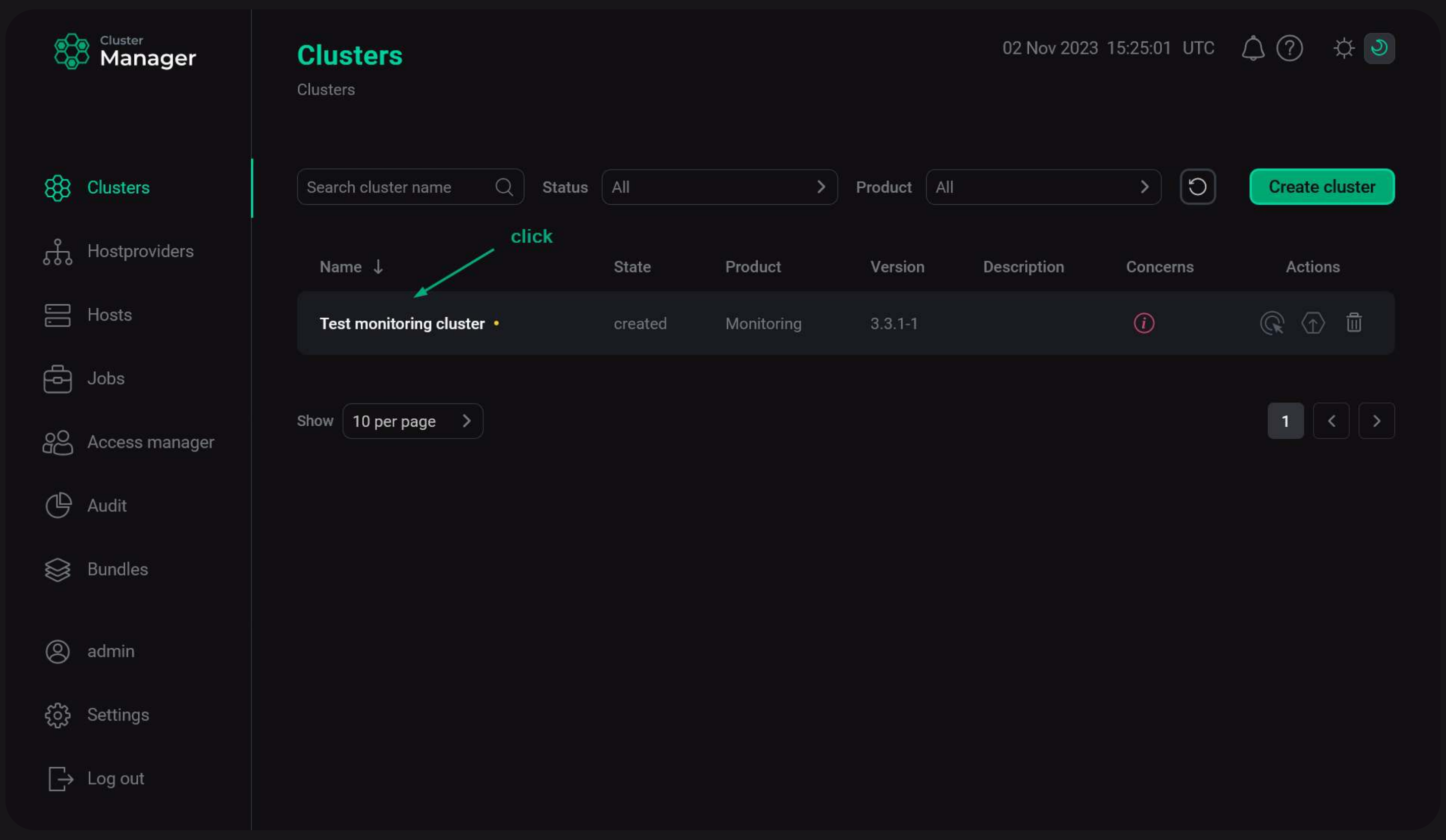
View mapping by hosts

Configure services

Daria Barysheva

To Table of Contents

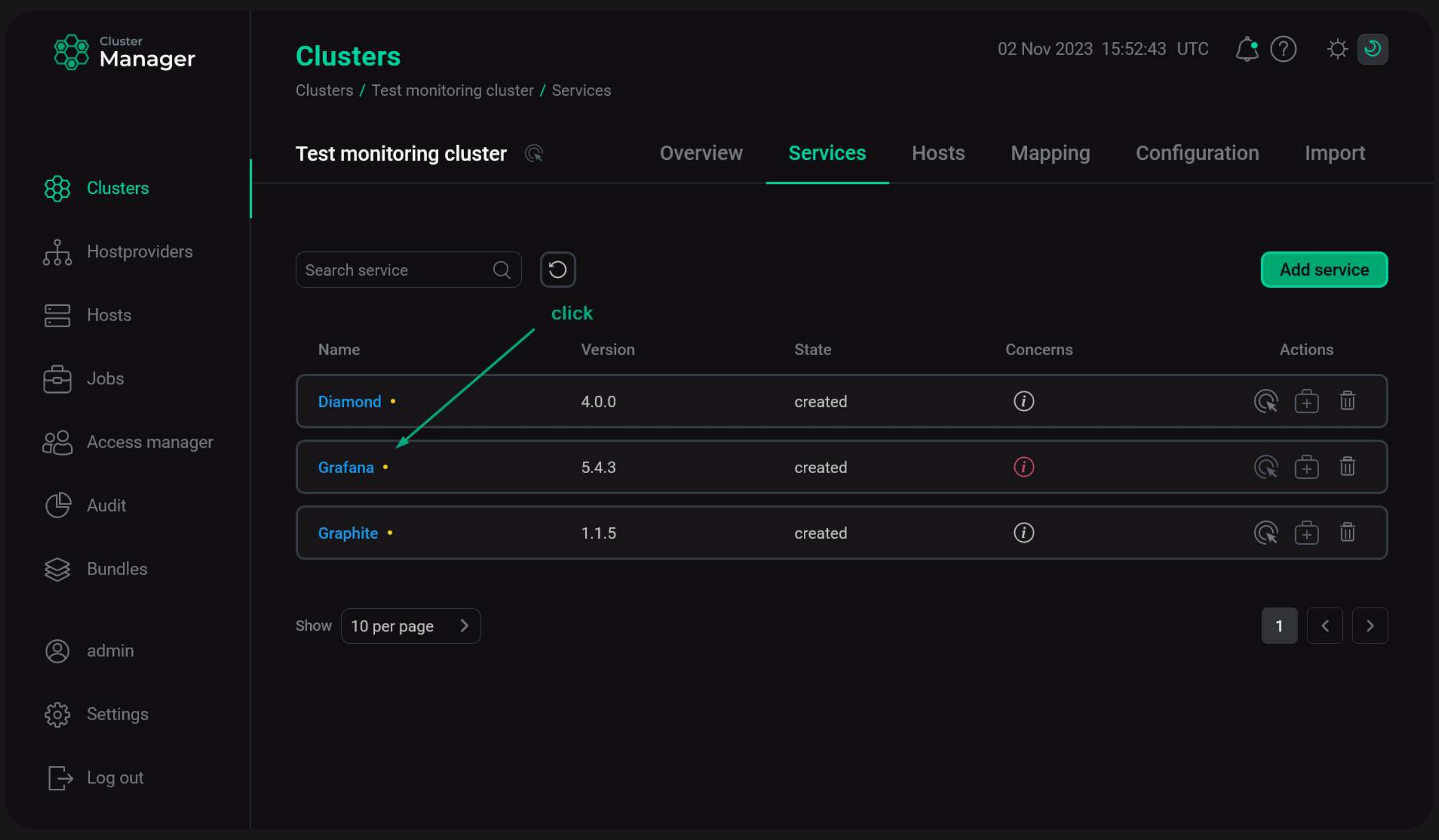
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Nov 2023 15:25:01 UTC
Search cluster name
Status All
Product All
Create cluster
click
Show 10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test monitoring cluster	created	Monitoring	3.3.1-1			

Select a cluster

2. Open the ***Services*** tab on the cluster page, find the row that contains the desired service, and click its name in the ***Name*** column.

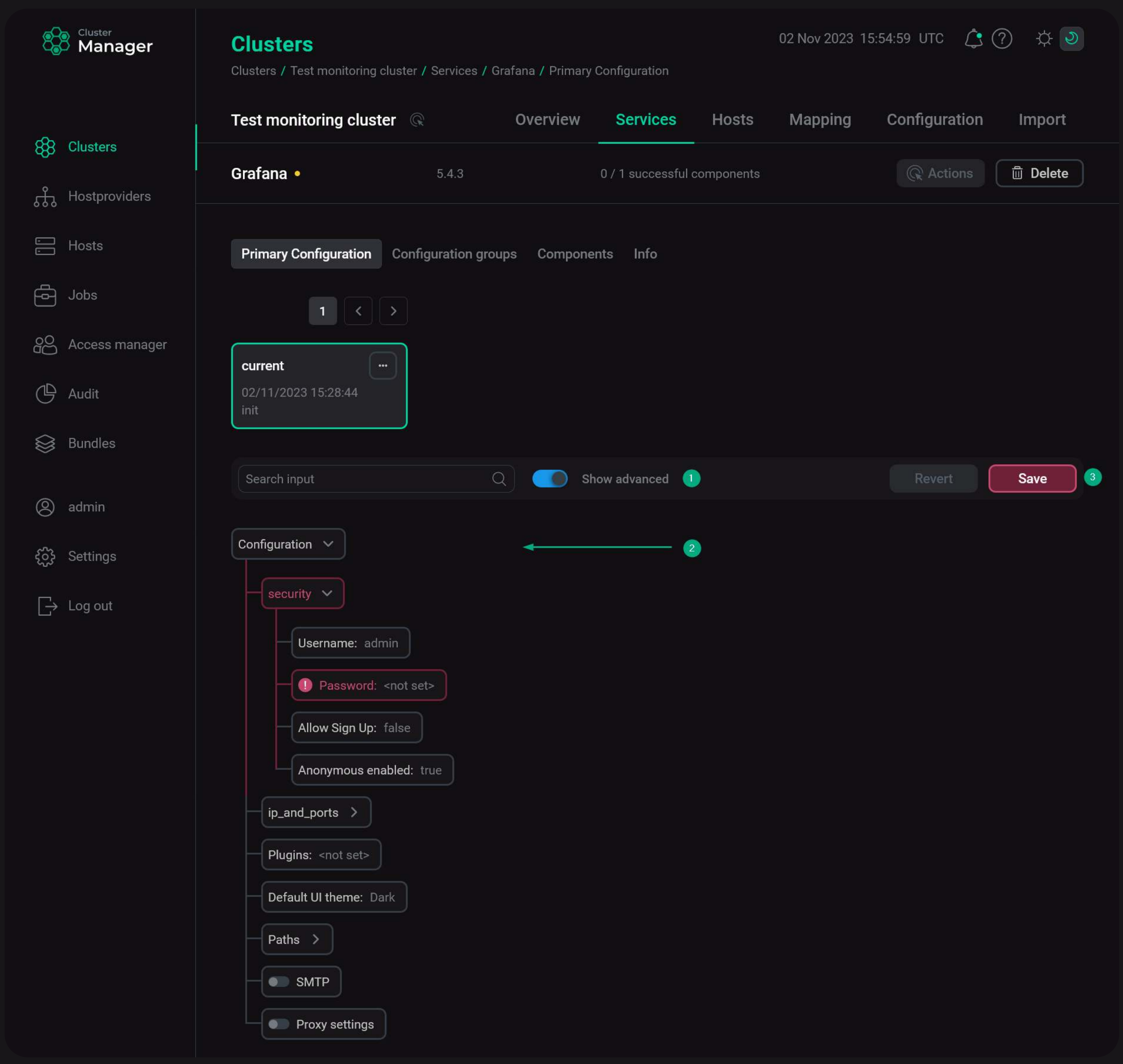

Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test monitoring cluster / Services
02 Nov 2023 15:52:43 UTC
Test monitoring cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
click
Show 10 per page
1

Name	Version	State	Concerns	Actions
Diamond	4.0.0	created		
Grafana	5.4.3	created		
Graphite	1.1.5	created		

Switch to configuring a service

3. In the window that opens:

   - Open the ***Primary Configuration*** tab.
   - Switch on the ***Show advanced*** toggle.
   - Fill in configuration parameters of the selected service. The fields highlighted in red are mandatory.
   - Click ***Save***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test monitoring cluster / Services / Grafana / Primary Configuration
02 Nov 2023 15:54:59 UTC
Test monitoring cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Grafana
5.4.3
0 / 1 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1
current
02/11/2023 15:28:44
init
Search input
Show advanced
1
Revert
Save
3
2
Configuration
security
ip_and_ports
Paths
SMTP
Proxy settings

Username: admin
Password: <not set>
Allow Sign Up: false
Anonymous enabled: true
Plugins: <not set>
Default UI theme: Dark

Configure a service

Install a cluster

Daria Barysheva

Contents

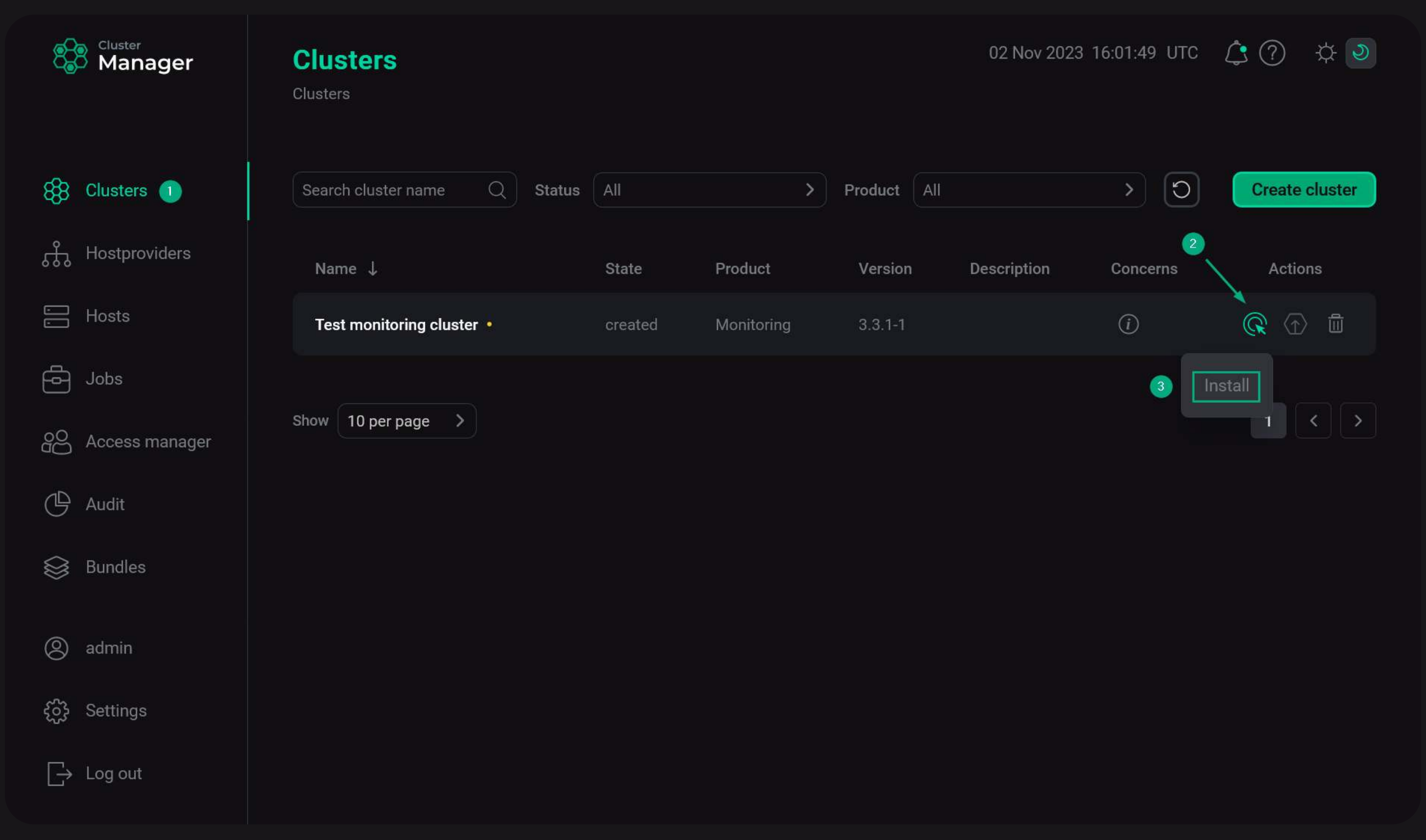
To Table of Contents

Step 1. Run installation
Step 2. View results
- Graphite
- Grafana

Step 1. Run installation

The last step of adding a new monitoring cluster is to install its services. To do this:

1. Apply the ***Install*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column. Verify the requested action in the opened window. All services are installed consecutively.



Cluster Manager
Clusters
Clusters
02 Nov 2023 16:01:49 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test monitoring cluster
created
Monitoring
3.3.1-1
Install
Show 10 per page
1
2

Switch to monitoring cluster installation

2. Wait until the job is completed.

NOTE

In case of errors, you can find logs on the ***Jobs*** page.

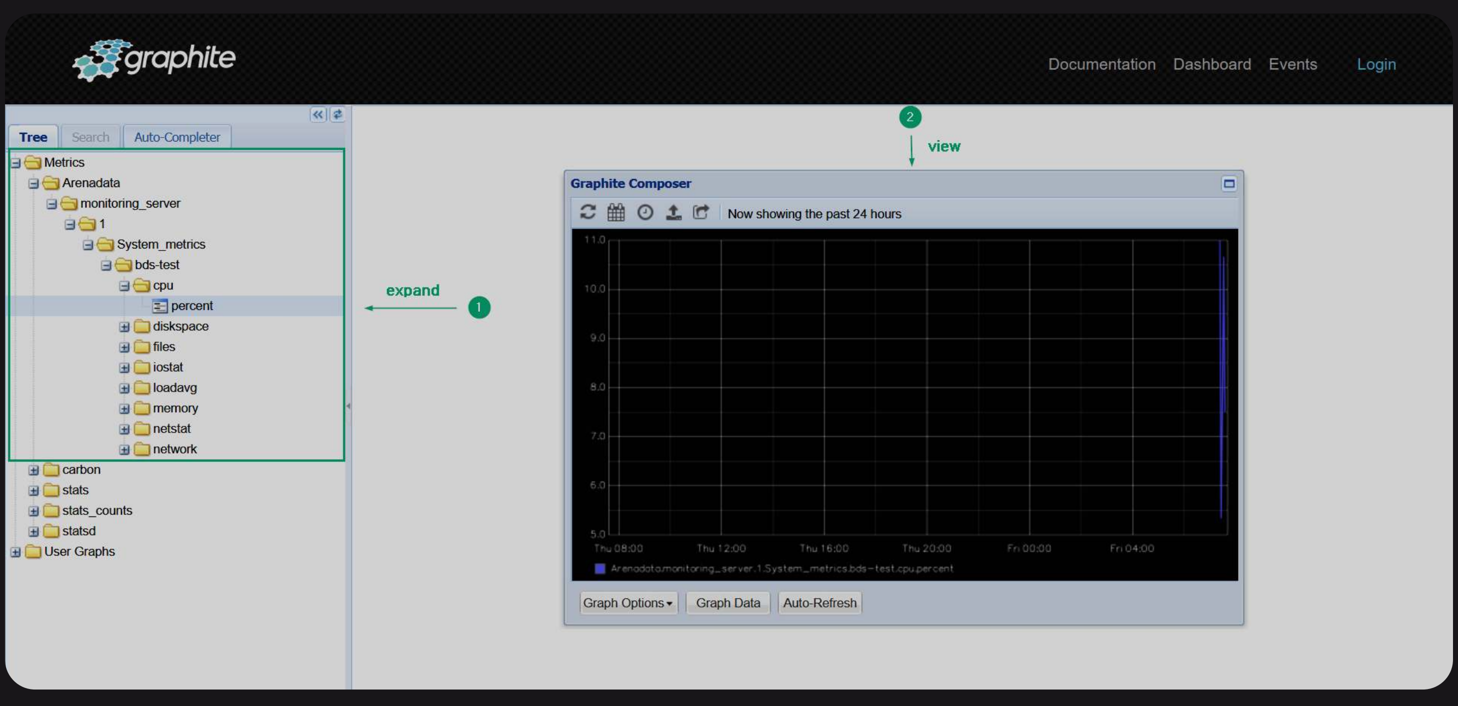
Step 2. View results

Graphite

To check availability of the Graphite web-interface, you can do:

1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Web-interface TCP port*** parameter from the ***ip_and_ports*** section defined when configuring the Graphite service during the monitoring cluster installation. The value `80` (used by default) can be omitted.

2. Expand a tree at the left side of the opened form. System metrics are displayed there, grouped by hosts. You can select some of these metrics to view their change dynamics in the form of a graph at the right side of the screen.



graphite
Documentation
Dashboard
Events
Login
Tree
Search
Auto-Completer
Metrics
Arenadata
monitoring_server
1
System_metrics
bds-test
cpu
percent
diskspace
files
iostat
loadavg
memory
netstat
network
carbon
stats
stats_counts
statsd
User Graphs
expand
1
2
view
Graphite Composer
Now showing the past 24 hours
Graph Options
Graph Data
Auto-Refresh

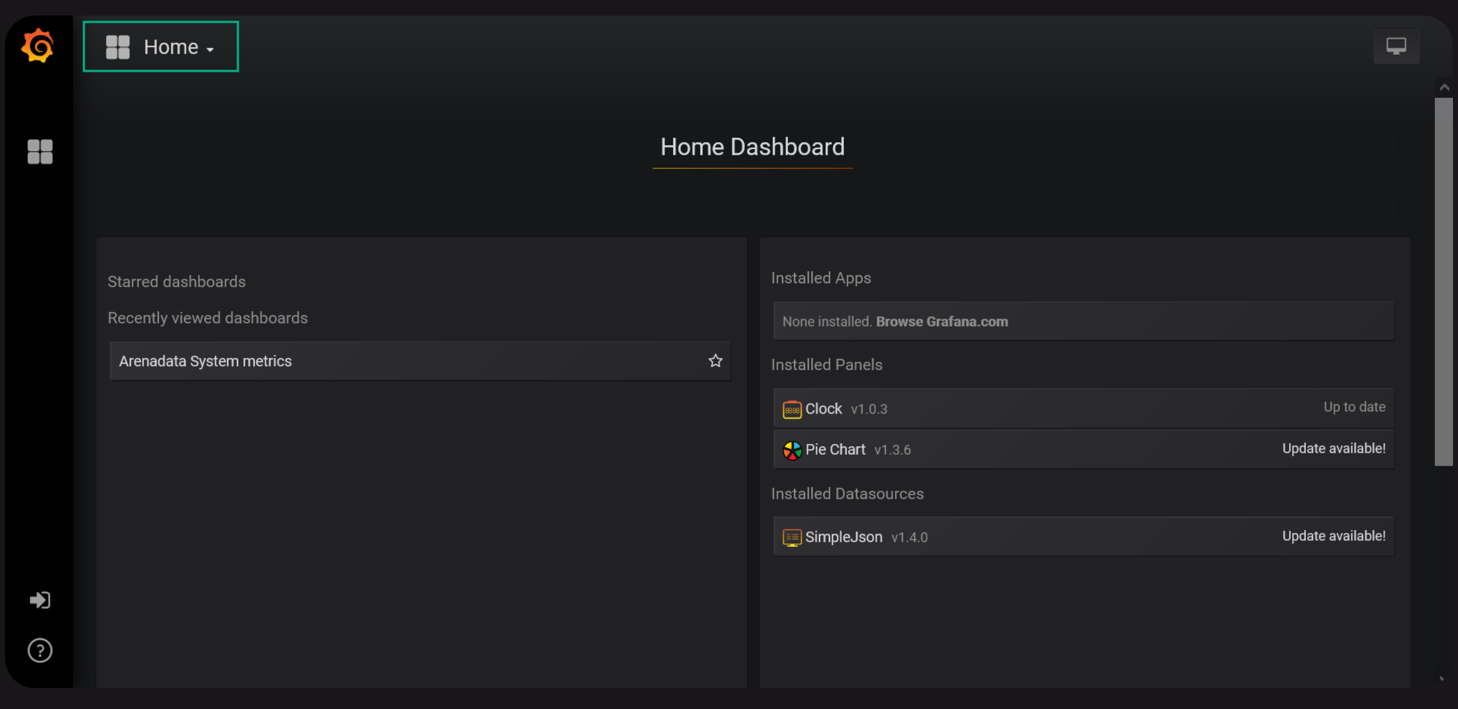
Graphite web-interface

Grafana

To check availability of the Grafana web-interface, you can do:

1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Port*** parameter from the ***ip_and_ports*** section defined when configuring the Grafana service during the monitoring cluster installation (by default, `3000`).

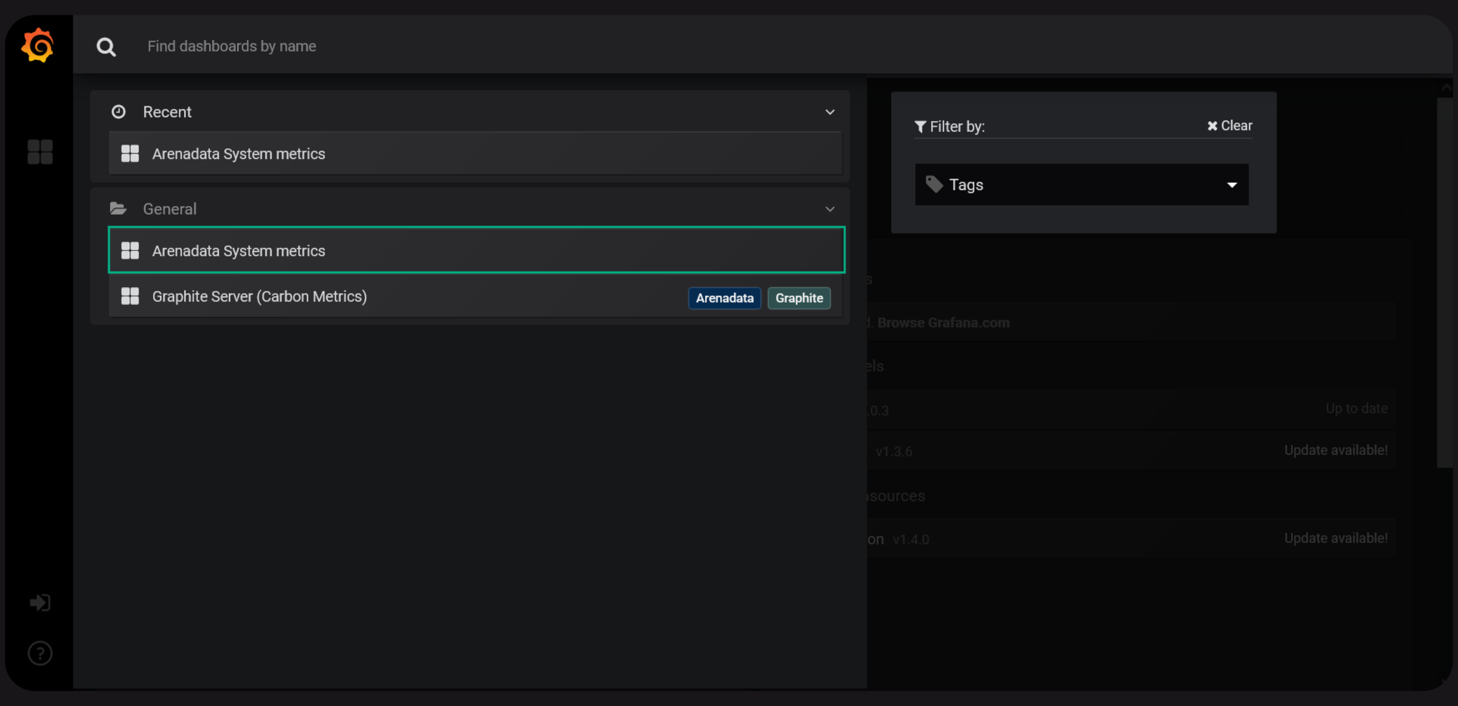
2. Select the ***Home*** menu item in the opened form.



Home
Home Dashboard
Starred dashboards
Recently viewed dashboards
Arenadata System metrics
Installed Apps
None installed. Browse Grafana.com
Installed Panels
Clock v1.0.2
Up to date
Pie Chart v1.3.3
Update available!
Installed Datasources
SimpleJson v1.4.0
Update available!

Grafana web-interface

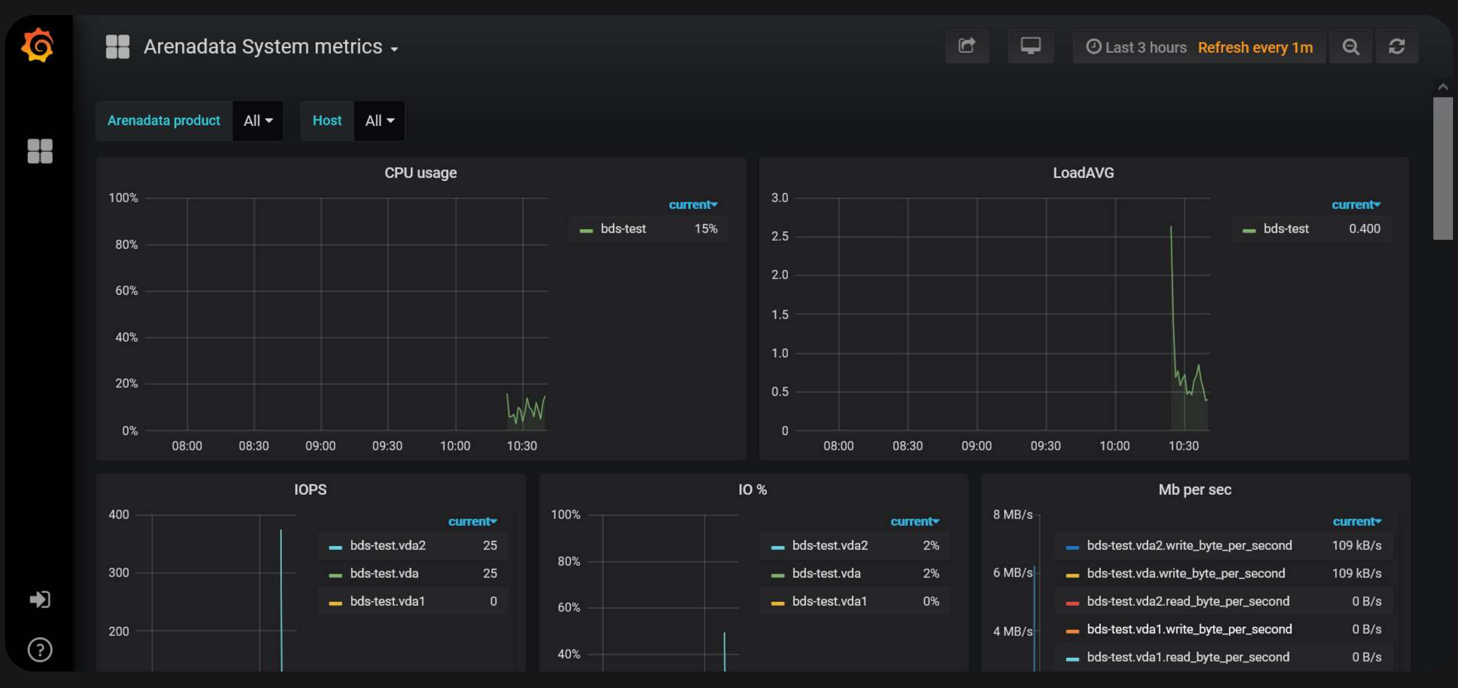
3. Open one of available dashboards. For example, ***Arenadata System metrics***.



Find dashboards by name
Recent
Arenadata System metrics
General
Arenadata System metrics
Graphite Server (Carbon Metrics)
Arenadata
Graphite
Filter by:
Clear
Tags

Switch to Arenadata System metrics

4. View available graphs.



Arenadata System metrics
Last 3 hours
Refresh every 1m
Arenadata product
All
Host
All
CPU usage
LoadAVG
IOPS
IO %
Mb per sec
current
bds-test

View Arenadata System metrics

Integrate with ADH cluster

Daria Barysheva

Contents

To Table of Contents

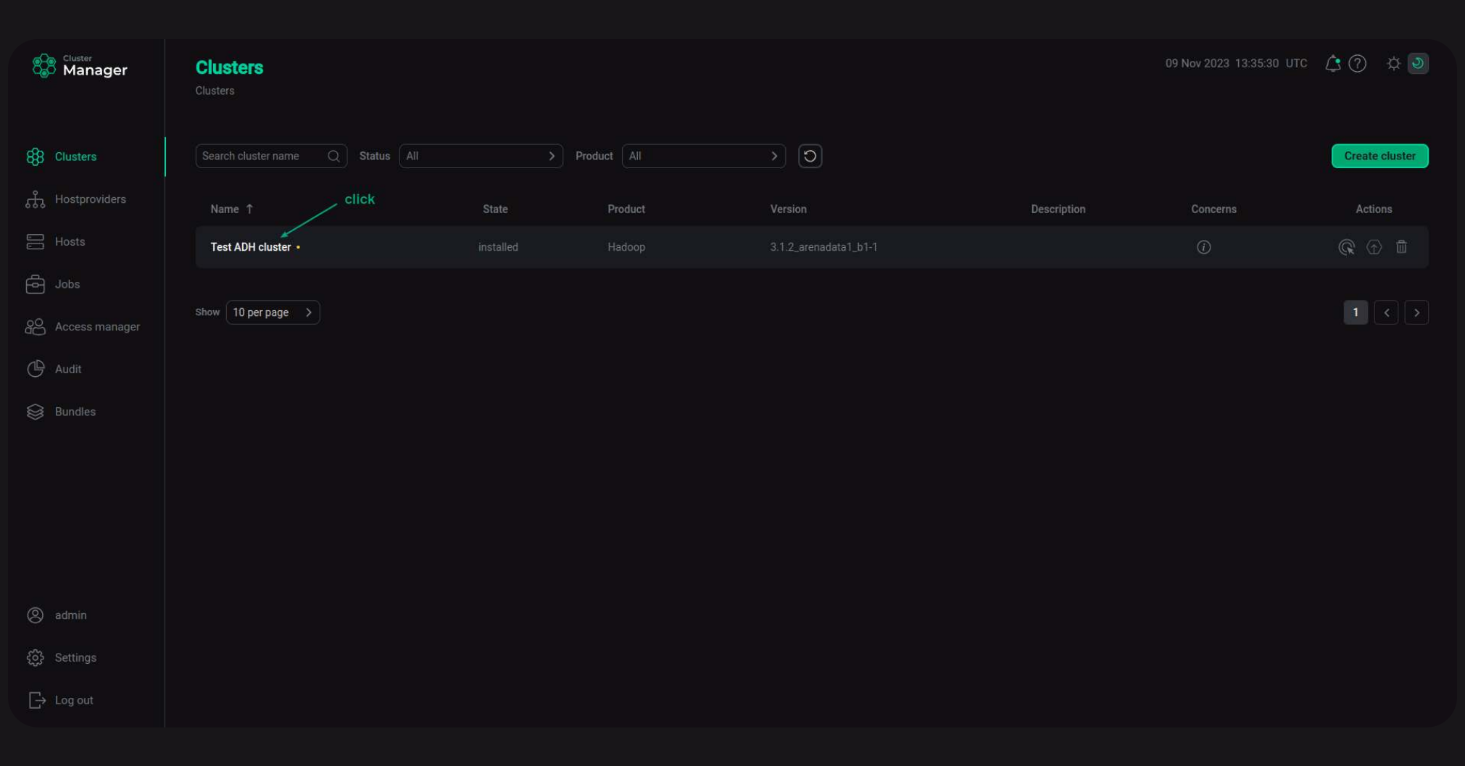
Step 1. Import monitoring settings
Step 2. Add a monitoring service
Step 3. Map monitoring components
Step 4. Install a monitoring service
Step 5. Verify results
  Graphite
  Grafana

In order to determine that a new cluster should be monitored, it is necessary to integrate it with the previously created monitoring cluster and to add a separate Monitoring service into it.

The steps for building integration between a monitoring cluster and ADH cluster are described below. All actions should be performed on the base of ADH cluster.

Step 1. Import monitoring settings

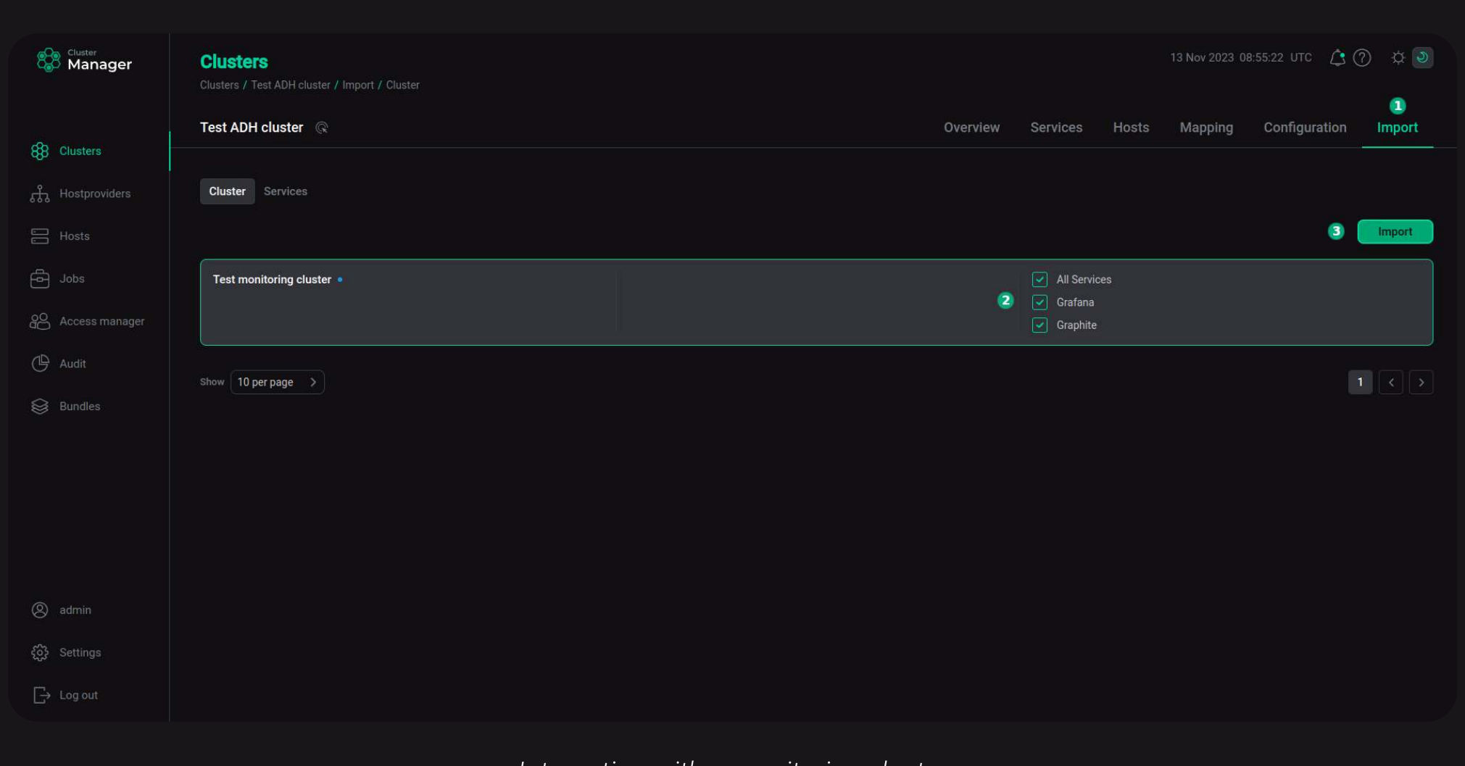
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.



Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Create cluster
click
Test ADH cluster
installed
Hadoop
3.1.2_arenadata1_b1-1

Select a cluster

2. Open the ***Import*** tab on the cluster page.
3. Select ***Graphite*** and ***Grafana*** services of the previously created monitoring cluster.
4. Click ***Import***.



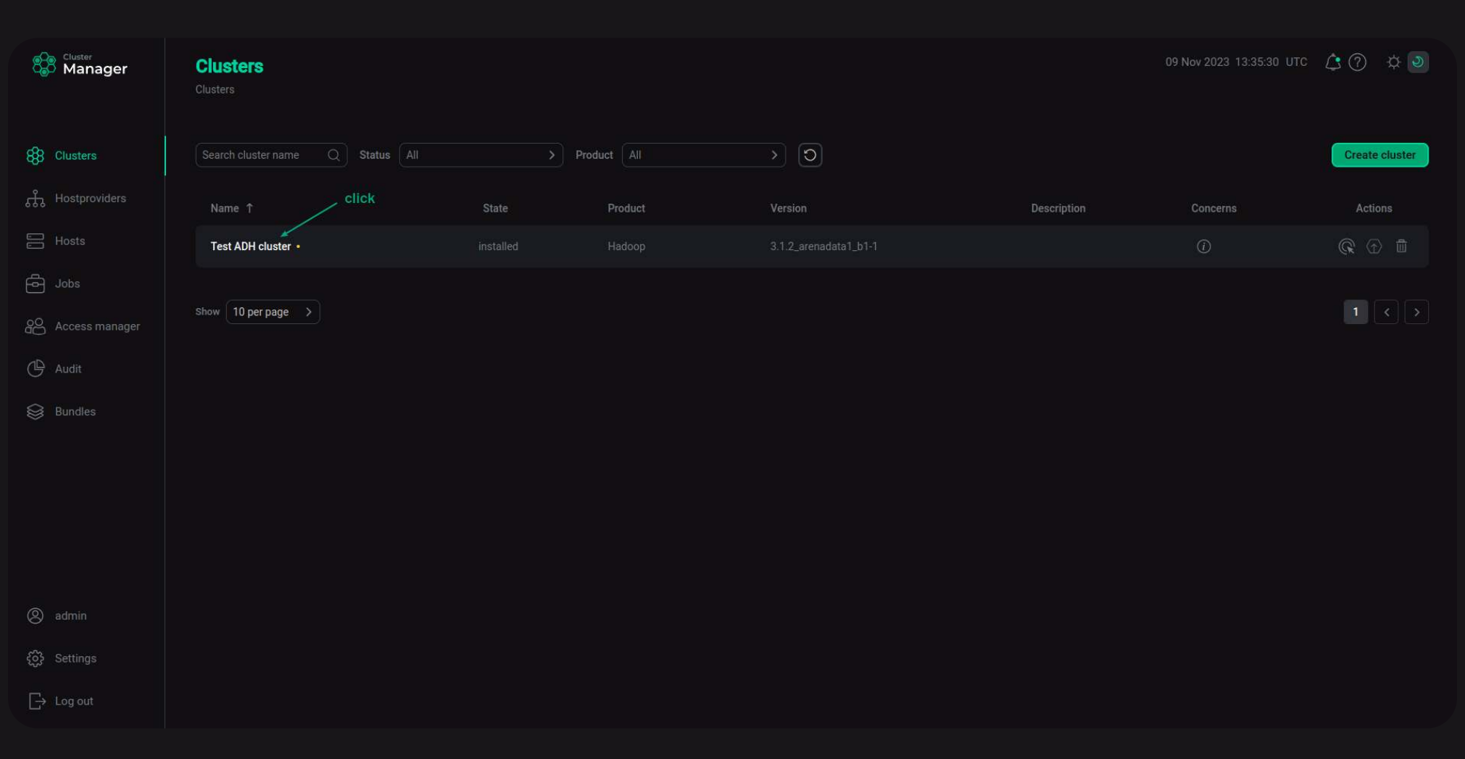
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Cluster
Services
Import
Test monitoring cluster
All Services
Grafana
Graphite
1
2
3

Integration with a monitoring cluster

Step 2. Add a monitoring service

The steps for adding services to a cluster are listed below:

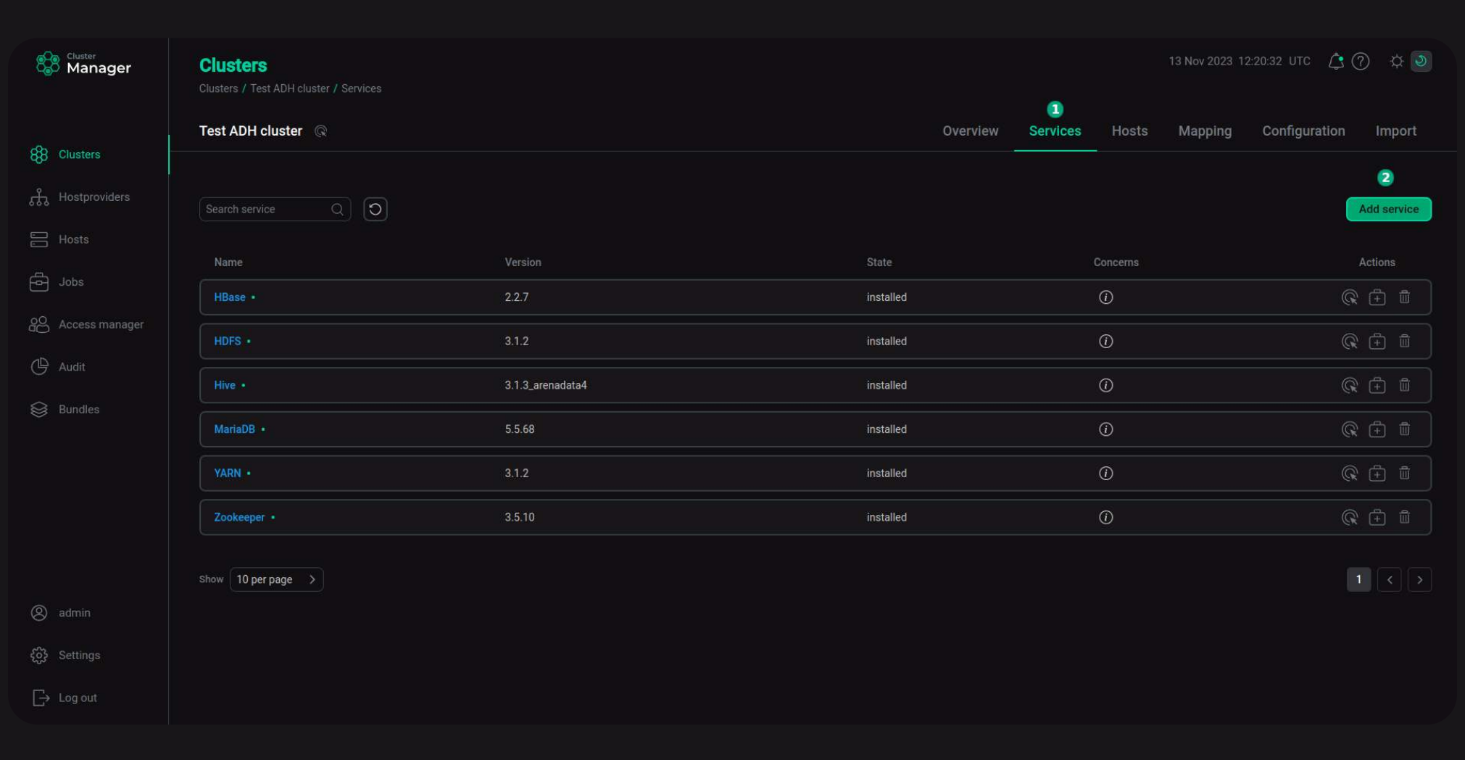
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.



Clusters
Create cluster
click
Test ADH cluster

Select a cluster

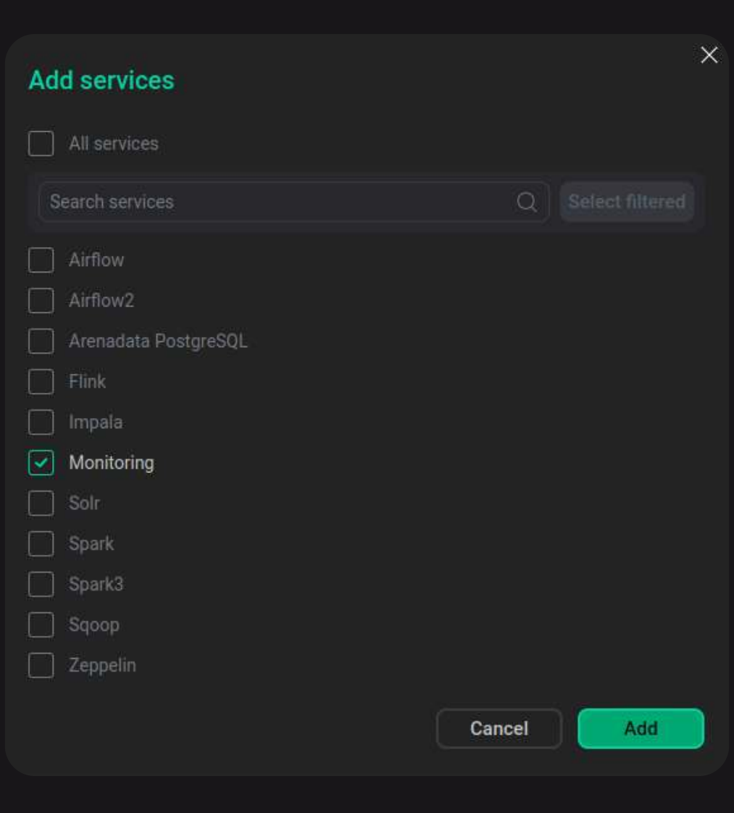
2. Open the ***Services*** tab on the cluster page and click ***Add service***.



Test ADH cluster
Services
Add service
HBase
HDFS
Hive
MariaDB
YARN
Zookeeper
1
2

Switch to adding services

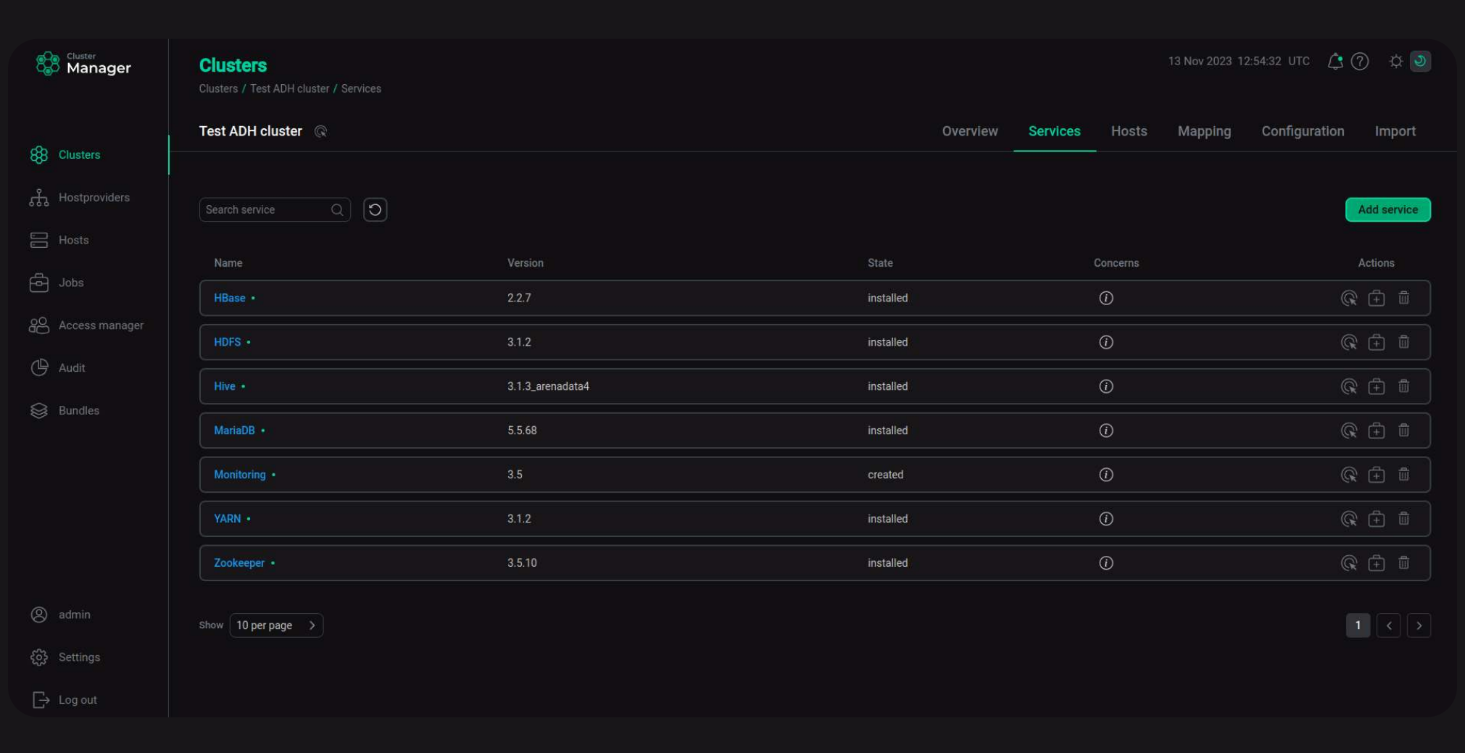
3. Select the service Monitoring in the opened dialog and click ***Add***.



Add services
☐ All services
Search services
Select filtered
☐ Airflow
☐ Airflow2
☐ Arenadata PostgreSQL
☐ Flink
☐ Impala
☑ Monitoring
☐ Solr
☐ Spark
☐ Spark3
☐ Sqoop
☐ Zeppelin
Cancel
Add

Select services

4. As a result, the added services are displayed on the ***Services*** tab.



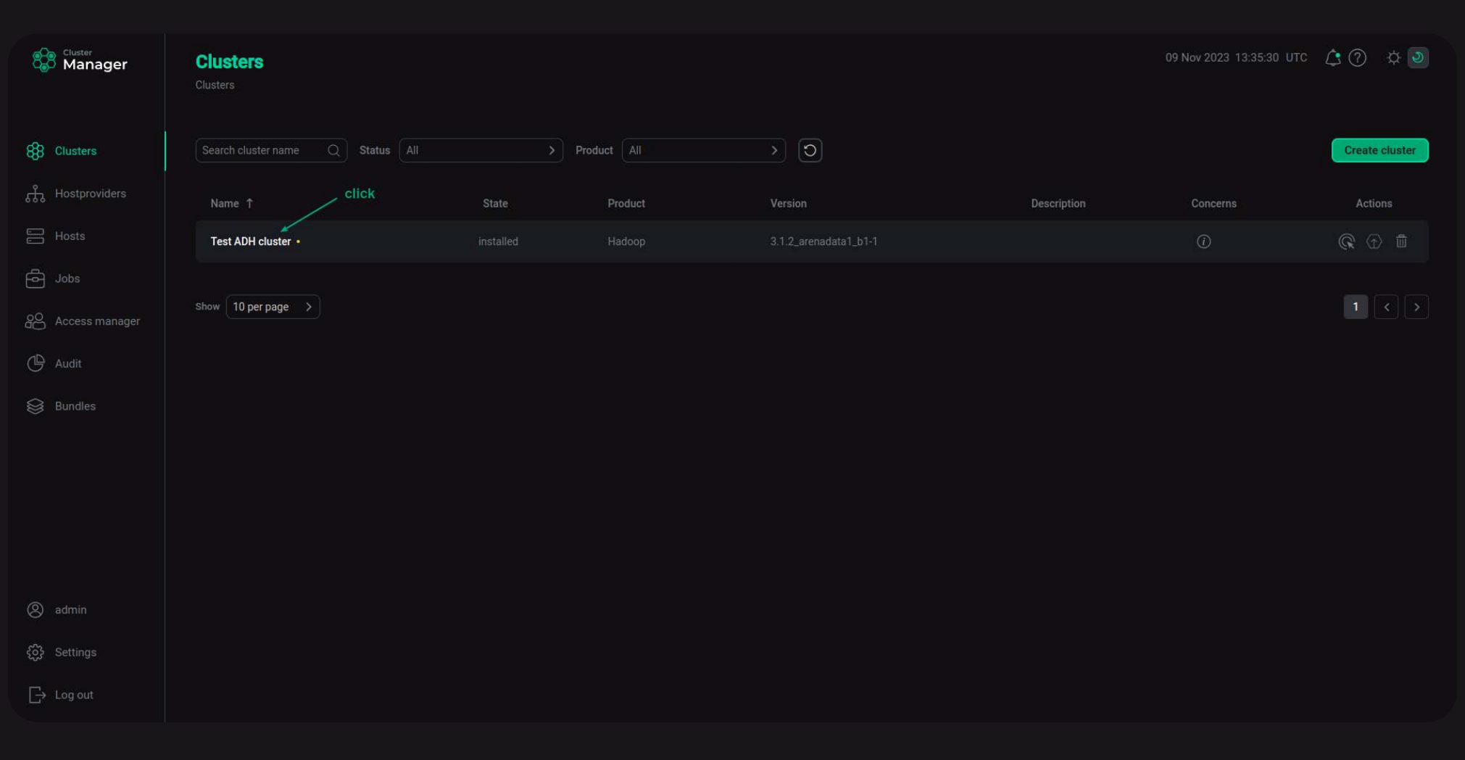
Test ADH cluster
Services
Add service
HBase
HDFS
Hive
MariaDB
Monitoring
YARN
Zookeeper
created
installed

The result of successful adding services to a cluster

Step 3. Map monitoring components

The steps for adding components to hosts are listed below:

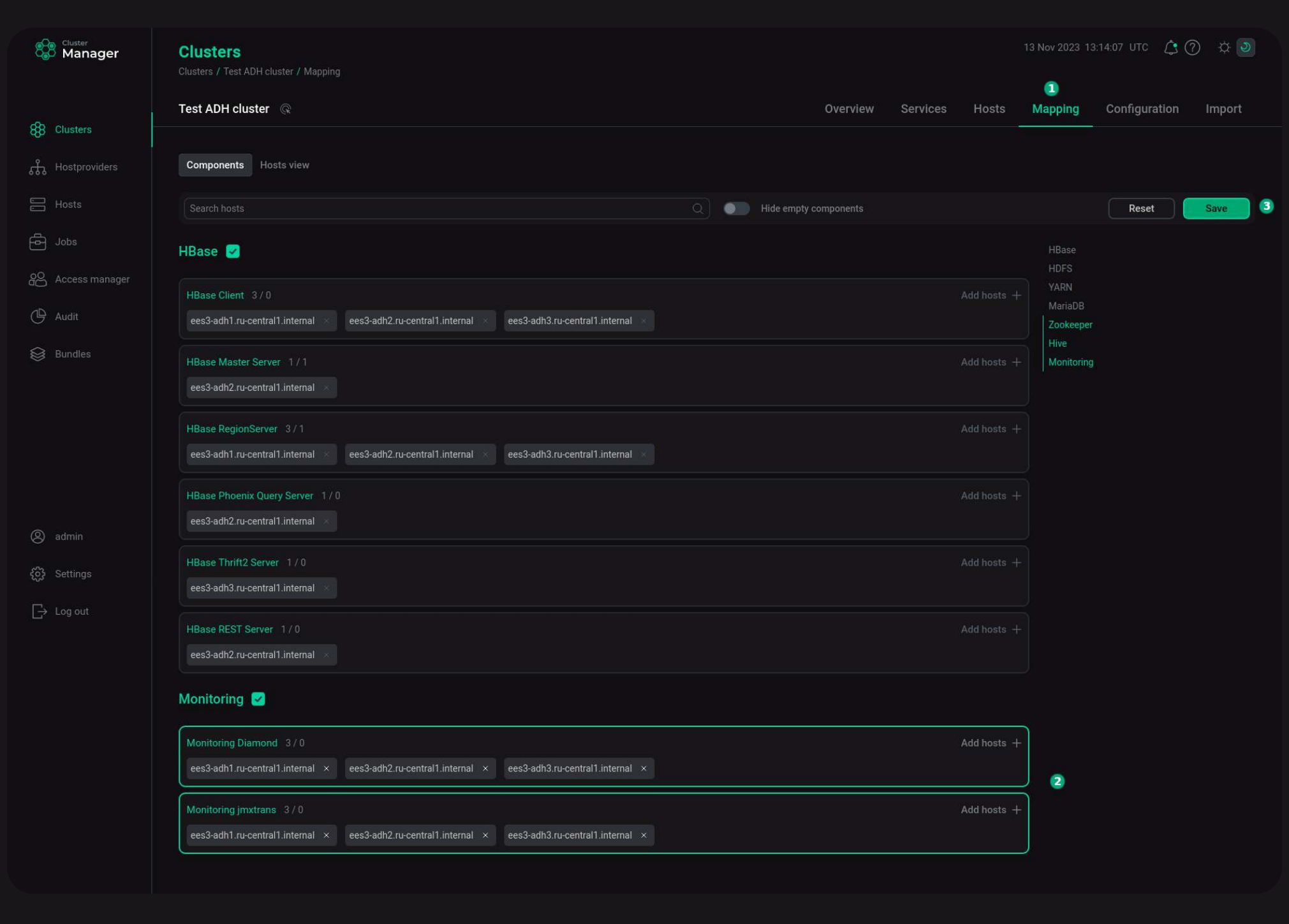
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.



Clusters
Create cluster
click
Test ADH cluster

Select a cluster

2. Open the ***Mapping*** tab on the cluster page, map monitoring components to all cluster hosts, and click ***Save***.

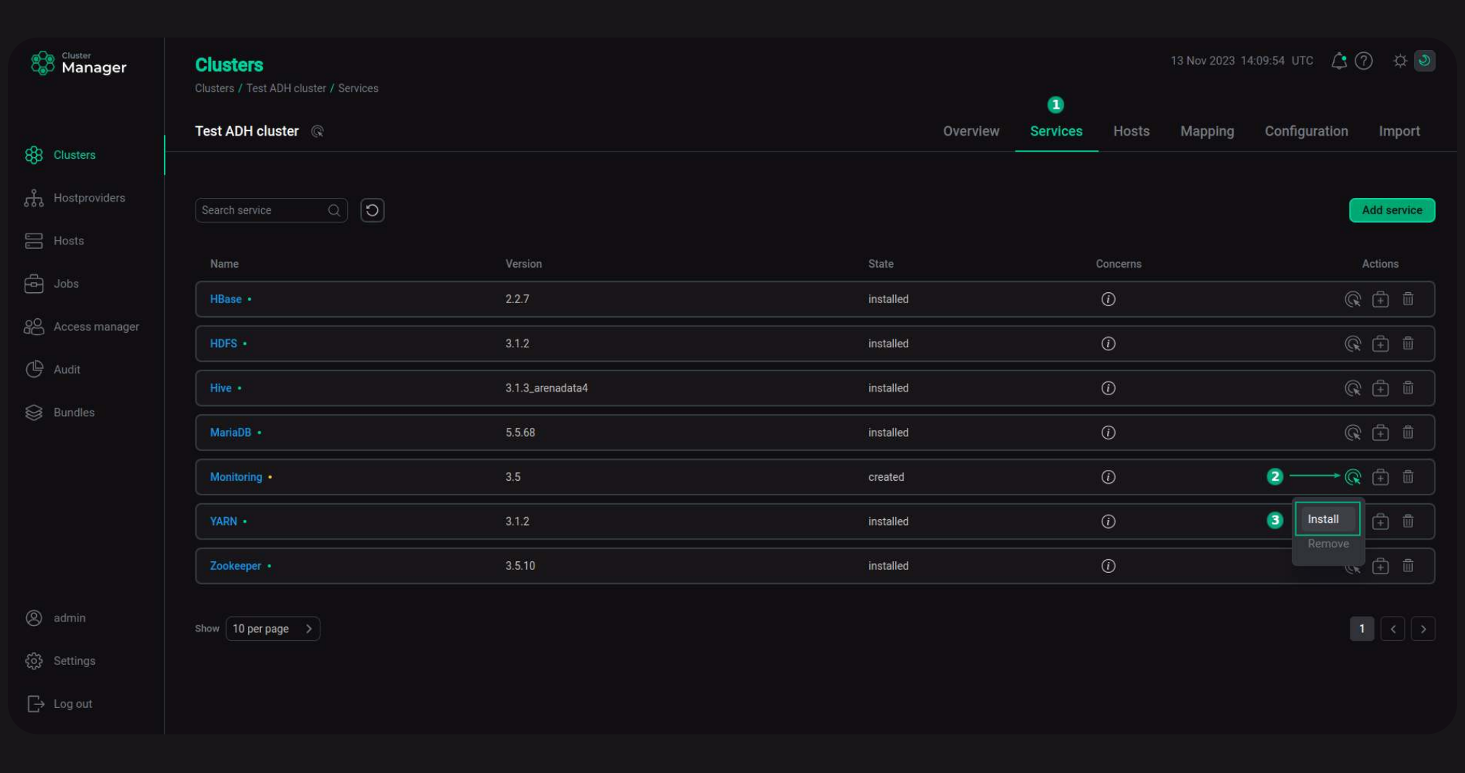


Test ADH cluster
Mapping
Components
Hosts view
Reset
Save
HBase
Monitoring
Monitoring Diamond
Monitoring jmxtrans
Add hosts
1
2
3

Map components to hosts

Step 4. Install a monitoring service

1. Open the ***Services*** tab in the cluster menu. Click the icon in the ***Actions*** column and select the ***Install*** action in the row that refers to the Monitoring service. Verify the action in the opened window.



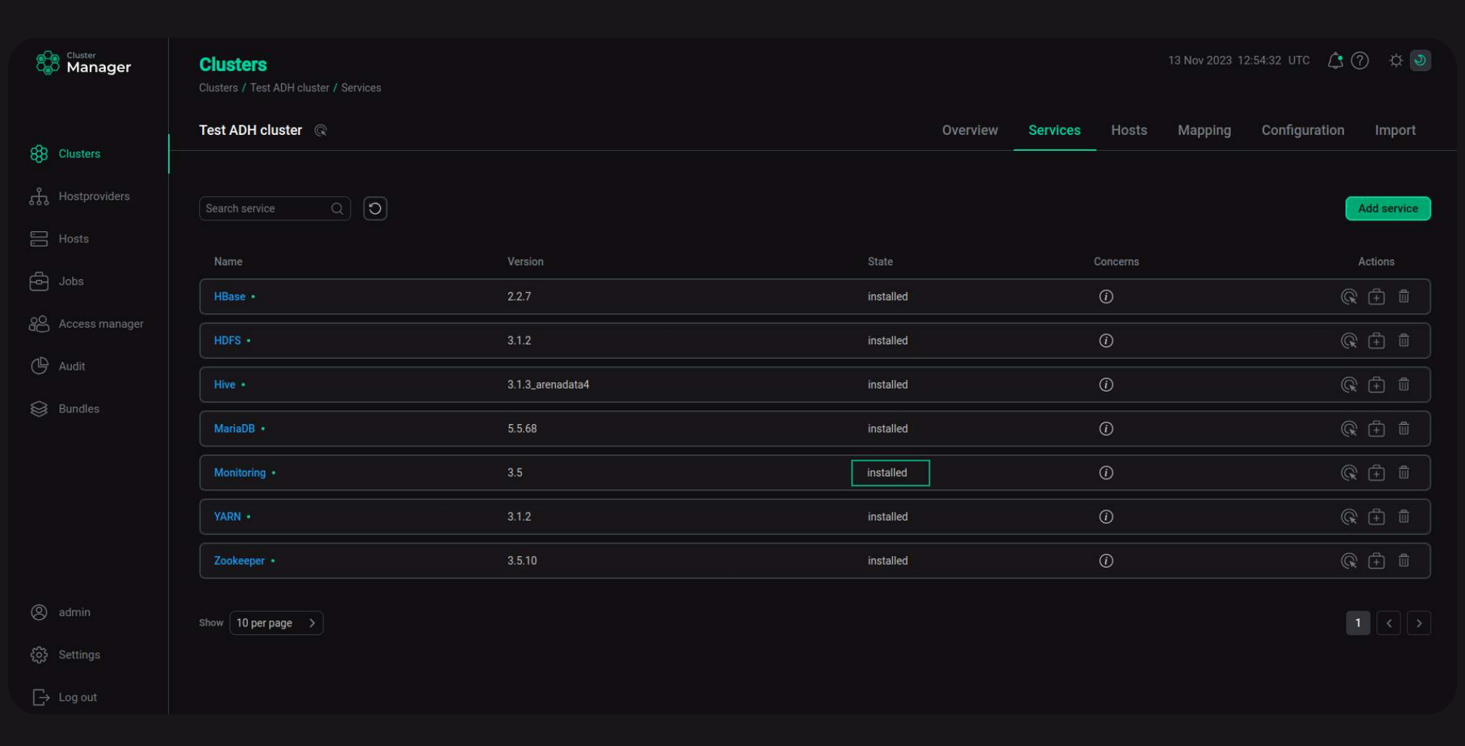
Test ADH cluster
Services
Monitoring
created
Install
Remove
1
2
3

Switch to monitoring service installation

2. Wait until the job is completed. After successful installation, the monitoring service changes its state from `created` to `installed`.

NOTE

In case of errors, you can find logs on the ***Jobs*** page.



Test ADH cluster
Services
Monitoring
installed

Monitoring installation is completed

CAUTION

After you install the monitoring cluster, you need to restart the ADH cluster to view metrics.

Step 5. Verify results

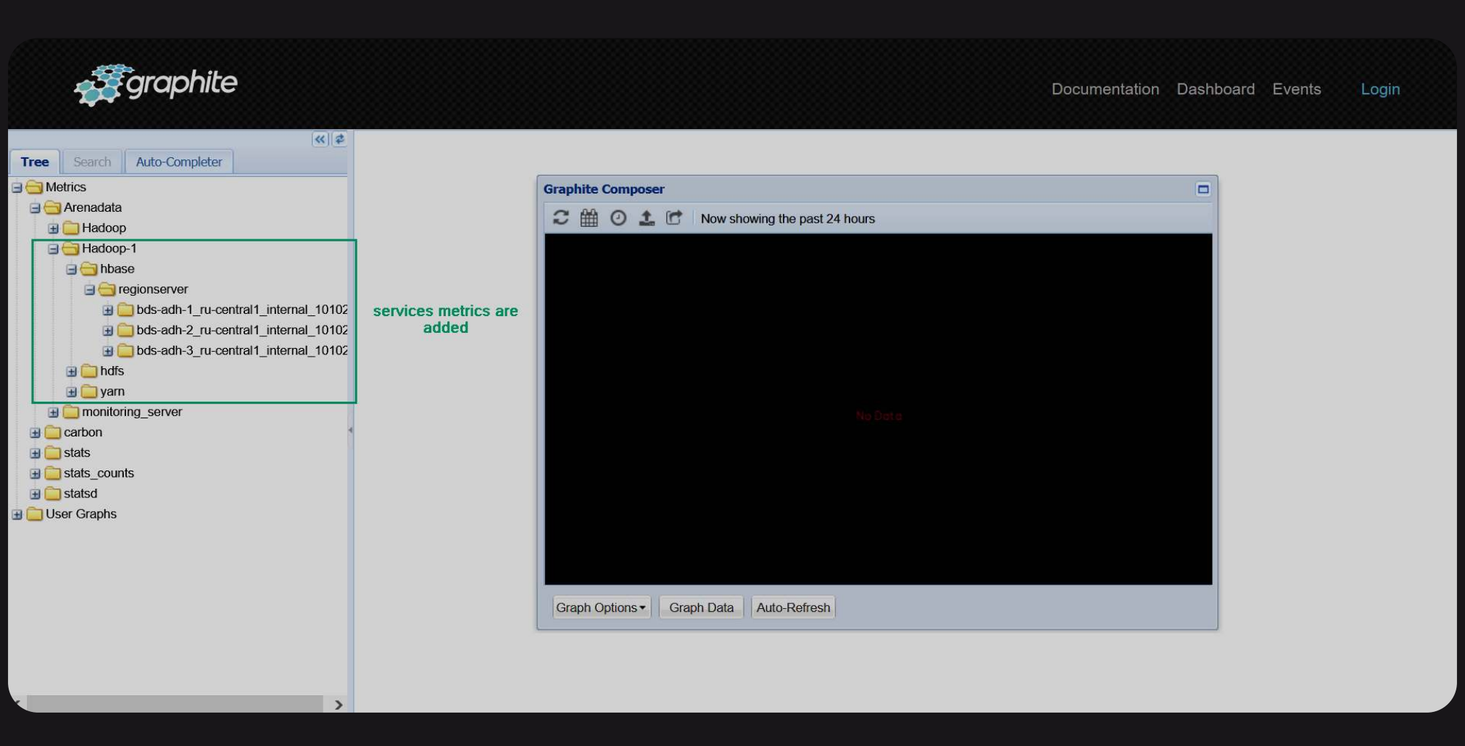
To ensure that installation process is completed successfully, check the following:

- System metrics are collected from the hosts of your cluster, not only from the host where the monitoring cluster has been deployed earlier.
- Along with system metrics, service metrics are collected from the hosts of your cluster. For example, ADH cluster provides metrics for such services as HDFS, YARN, HBase, Spark3.

You can check it both in the Graphite and Grafana web-interface.

Graphite

1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Web-interface TCP port*** parameter from the ***ip_and_ports*** section defined when configuring the Graphite service during the monitoring cluster installation. The value `80` (used by default) can be omitted.
2. Expand a tree at the left side of the opened form. System and service metrics are displayed there, grouped by hosts. You can select some of these metrics to view their change dynamics in the form of a graph at the right side of the screen.

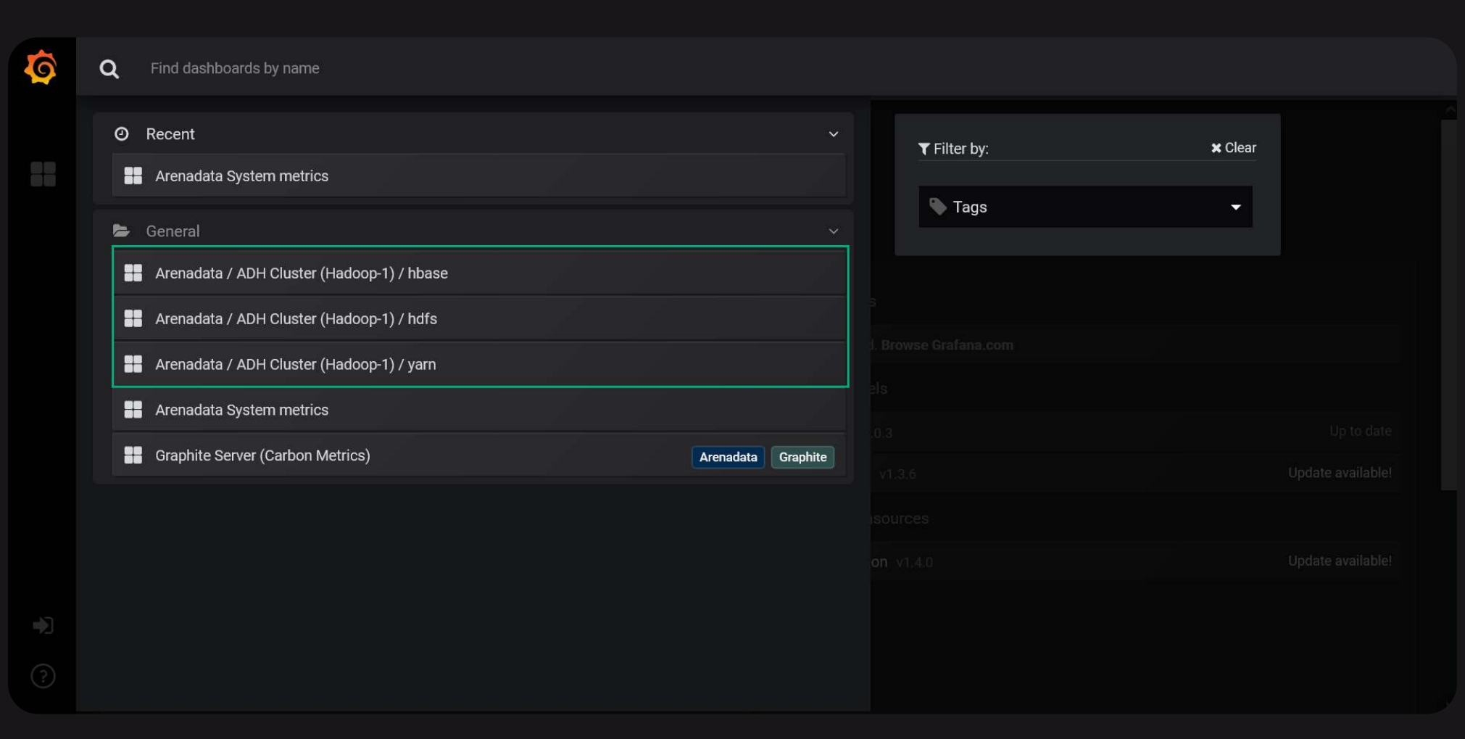


graphite
Documentation
Dashboard
Events
Login
Tree
Search
Auto-Completer
Metrics
Arenadata
Hadoop
Hadoop-1
hbase
regionserver
bds-adh-1_ru-central1_internal_10102
bds-adh-2_ru-central1_internal_10102
bds-adh-3_ru-central1_internal_10102
hdfs
yarn
monitoring_server
carbon
stats
stats_counts
statsd
User Graphs
services metrics are added
Graphite Composer
Now showing the past 24 hours
Graph Options
Graph Data
Auto-Refresh

New hosts and service metrics are available in Graphite

Grafana

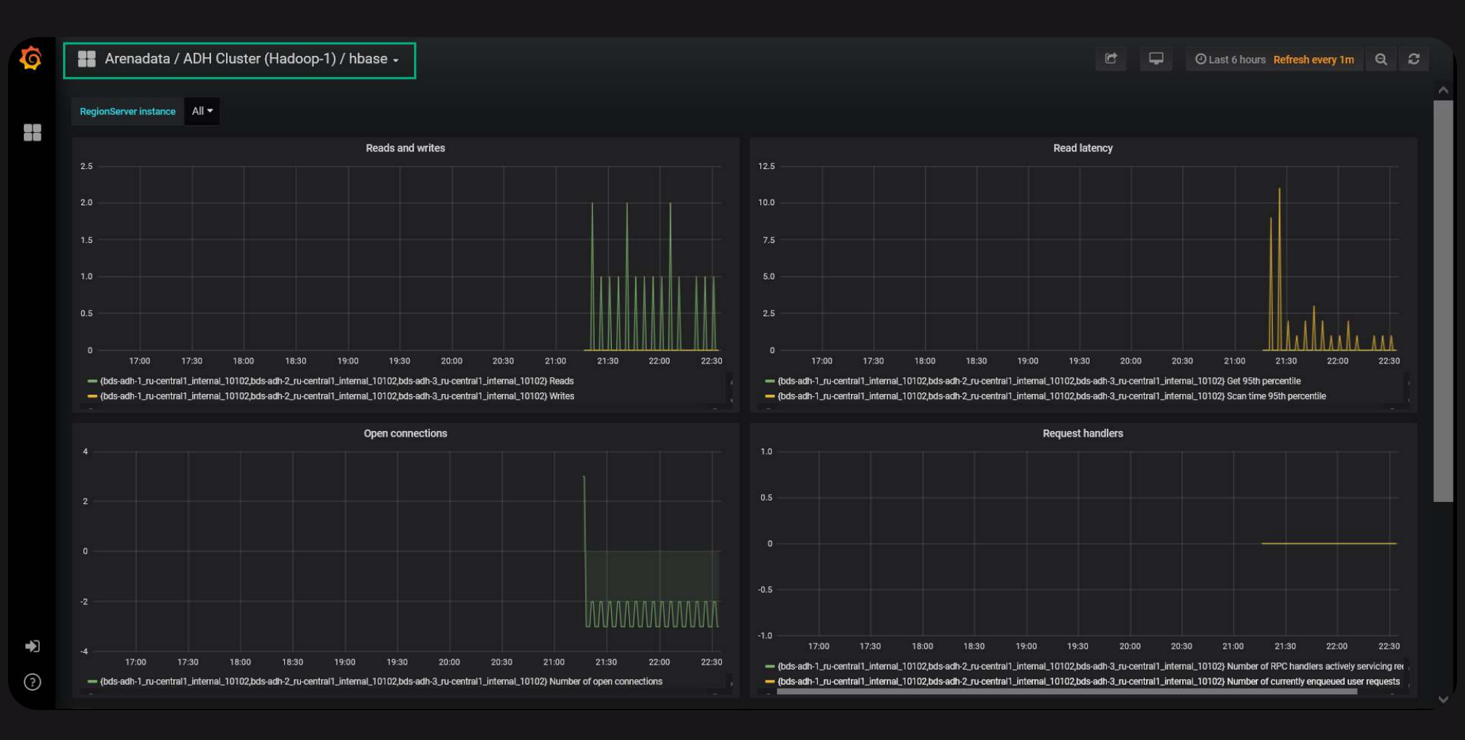
1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Port*** parameter from the ***ip_and_ports*** section defined when configuring the Grafana service during the monitoring cluster installation (by default, `3000`).
2. Select the ***Home*** menu item in the opened form.
3. Notice that new dashboards are available. Select one of them to view the service metrics that are collected from your cluster.



Find dashboards by name
Recent
Arenadata System metrics
General
Arenadata / ADH Cluster (Hadoop-1) / hbase
Arenadata / ADH Cluster (Hadoop-1) / hdfs
Arenadata / ADH Cluster (Hadoop-1) / yarn
Arenadata System metrics
Graphite Server (Carbon Metrics)
Arenadata
Graphite
Filter by:
Clear
Tags

Switch to service metrics

4. View available graphs.



Arenadata / ADH Cluster (Hadoop-1) / hbase
Last 6 hours
Refresh every 1m
RegionServer instance
All
Reads and writes
Read latency
Open connections
Request handlers

View service metrics

Offline installation

To Table of Contents

This topic describes **offline** installation of ADH. It can be applied to the hosts without access to the Internet.

For offline installation, you need to install and configure Arenadata Enterprise Tools (ET). ET is a solution that provides the necessary infrastructure for Arenadata product deployment in environments with limited internet access. The solution is a part of the Arenadata data platform. It provides the repositories necessary for installing and updating the platform components.

The process of offline installation via ADCM includes the following steps.

NOTE

This function is available only in the Enterprise Edition of ADH.

Step 1.

Install ADCM

Step 2.

Prepare hosts

Step 3.

Install ET cluster

Step 4.

Install ADH cluster

Step 5.

Install monitoring

Install ADCM

Daria Barysheva

Contents

To Table of Contents

- Step 1. Check prerequisites
  - Software and hardware requirements
  - Install Docker
  - Disable SELinux
  - Stop firewalld
  - Allow access to the Docker CLI without root privileges
- Step 2. Install ADCM
- Step 3. Start ADCM
- Step 4. Check installation
  - Check the Docker container status
  - Check the 8000 port availability
  - Check the ADCM web-interface
  - Set the ADCM URL
- Step 5. Update ADCM

ADCM requires a separate host. Before installing ADCM, we recommend to read its documentation.

The installation steps are listed below.

Step 1. Check prerequisites

Software and hardware requirements

ADCM is a software that is distributed in the form of a Docker image. To install it, you need:

- One of the following operating systems:
  - Ubuntu 22.04
  - CentOS 7
  - RHEL 7
  - Alt Linux SP 8
  - Alt Linux SP 10.2
  - Astra Linux SE 1.7 Orel
  - Astra Linux 1.7.6 Voronezh
  - RED OS 7.3 Certified edition
- A user account with `sudo` privileges.
- An access to the official repository of CentOS Extras and CentOS Base. This repository is enabled by default, but if you have disabled it — you need to re-enable it (e.g. via editing the repository configuration file in the */etc/yum.repos.d/* directory).
- The software package installer YUM/APT.
- The hardware that meets the following conditions:

> **IMPORTANT**
>
> The following system requirements are **minimal**. The target sizing should be calculated based on the customer requirements.

- **CPU**: 4 CPU cores;
- **RAM**: 16 GB;
- **HDD**: 50 GB.

> **IMPORTANT**
>
> Starting with ADCM 2.7.0, any Container Registry Client with downloaded images of ADPG, ADCM, and ADCM Installer will be required to install ADPG without the Internet connection and use it as the ADCM database. The installation is done via the ADCM Installer by specifying the URL of the local storage of the ADCM and ADPG images.

To grant the user the necessary rights to install ADCM, do the following:

1. Grant the `root` privileges for the user:

CentOS 7/RHEL 7/RED OS 7.3/Alt Linux SP 8

```
$ sudo usermod -aG wheel <user>
```

where `<user>` is a username.

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
$ sudo usermod -aG sudo <user>
```

where `<user>` is a username.

2. Make sure that the */etc/sudoers* file contains the record specified below depending on the operating system with the following command:

```
$ sudo vi /etc/sudoers
```

To return to the command line, press `Esc`, type `:q!`, and press `Enter`.

CentOS 7/RHEL 7/RED OS 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Install Docker

> **CAUTION**
>
> Do not install Docker from the official Docker site. Instead, use the official repositories of your Linux distribution. The repositories provided by operating system developers tend to be more stable and secure.

Below are the step-by-step procedures of Docker installation on the examples of several operating systems supported by ADCM. These steps may differ for operating system versions other than those specified on tabs.

CentOS 7/RHEL 7

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo yum update -y
```

2. Install the packages required for Docker installation and usage:

```
$ sudo yum install -y yum-utils docker device-mapper-persistent-data lvm2
```

where:

- `yum-utils` is the package that provides the `yum-config-manager` utility;
- `device-mapper-persistent-data` and `lvm2` are packages required by the devicemapper storage driver.

3. Start Docker:

```
$ sudo systemctl start docker
```

4. Enable Docker as a system service:

```
$ sudo systemctl enable docker
```

Ubuntu 22.04

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo apt-get update
```

2. Install the packages required for Docker installation and usage:

```
$ sudo apt-get install ca-certificates curl
```

where:

- `ca-certificates` is the package of Root CA certificates;
- `curl` is the package for working with web resources.

Confirm the use of additional disk space during the execution of the command.

3. Add the official GPG key for the Docker repository:

```
$ sudo install -m 0755 -d /etc/apt/keyrings
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

4. Add the Docker repository:

```
$ echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "jammy") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Update all packages, including packages for the Docker repository:

```
$ sudo apt-get update
```

6. Install Docker Engine:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

Confirm the use of additional disk space during the execution of the command.

7. Start Docker:

```
$ sudo systemctl start docker
```

8. Enable Docker as a system service:

```
$ sudo systemctl enable docker
```

Alt Linux SP 8

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo apt-get update
```

2. Install Docker Engine:

```
$ sudo apt-get install docker-engine
```

Confirm the use of additional disk space during the execution of the command.

3. Start and enable Docker as a system service:

```
$ sudo systemctl enable --now docker
```

Astra Linux SE 1.7 Orel

1. Using `root` privileges, update all packages that are currently installed in your system:

```
$ sudo apt-get update
```

2. Install the packages required for Docker installation and usage:

```
$ sudo apt-get install ca-certificates curl net-tools
```

where:

- `ca-certificates` is the package of Root CA certificates;
- `curl` is the package for working with web resources;
- `net-tools` is the package for controlling the network subsystem of the Linux kernel, in particular for executing the `netstat` command.

Confirm the use of additional disk space during the execution of the command.

3. Add the official GPG key for the Docker repository:

```
$ curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add -
```

4. Add the Docker repository:

```
$ echo "deb [arch=amd64] https://download.docker.com/linux/debian stretch stable" | sudo tee -a /etc/apt/sources.list
```

5. Update all packages, including packages for the Docker repository:

```
$ sudo apt-get update
```

6. Install Docker Engine:

```
$ sudo apt install docker-ce docker-ce-cli containerd.io
```

Confirm the use of additional disk space during the execution of the command.

7. Start Docker:

```
$ sudo systemctl start docker
```

8. Enable Docker as a system service:

```
$ sudo systemctl enable docker
```

RED OS 7.3

1. Install Docker Engine:

```
$ sudo dnf install docker-ce docker-ce-cli
```

Confirm the use of additional disk space during the execution of the command.

2. Start and enable Docker as a system service:

```
$ sudo systemctl enable docker --now
```

Disable SELinux

> **NOTE**
>
> It is recommended to disable SELinux only if CentOS 7 or RHEL 7 is used.

To disable SELinux permanently, set `SELINUX=disabled` in the */etc/selinux/config* file, and reboot your system.

You can edit the file via the `vi` or `vim` command:

```
$ sudo vi /etc/selinux/config
```

The content of the changed file should look like this:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted
```

You can also disable SELinux temporarily via the following command:

```
$ sudo setenforce 0
```

Still, it is strongly recommended to disable SELinux permanently, so that it does not restart on each system reboot.

> **CAUTION**
>
> Do not forget to reboot your host after SELinux is disabled.

Stop firewalld

If you use the firewalld service, stop it before installing ADCM.

Before stopping firewalld, make sure that it is installed, started, and enabled:

```
$ sudo systemctl status firewalld
```

The example of the command execution result:

```
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2024-12-02 10:02:57 UTC; 3min 25s ago
     Docs: man:firewalld(1)
 Main PID: 3023 (firewalld)
    Tasks: 2
   Memory: 22.8M
   CGroup: /system.slice/firewalld.service
           └─3023 /usr/bin/python2 -Es /usr/sbin/firewalld --nofork --nopid
```

Stop firewalld:

```
$ sudo systemctl stop firewalld
```

As an alternative, you can disable the firewalld service, so that it will not apply rules to network packets:

```
$ sudo systemctl disable firewalld
```

Allow access to the Docker CLI without root privileges

If in the future you want to access the Docker CLI without using `root` privileges, you should create a `docker` user group (if it does not exist) and add the current user to this group with the certain rights:

```
$ sudo groupadd docker
$ sudo usermod -a -G docker $USER
```

After running these commands, you should re-login under the current user.

Step 2. Install ADCM

Offline ADCM installation includes the following steps:

1. Get an archive with the *sh.xz* extension that contains a special installation shell script from the Arenadata support team.
2. Make a directory on the host where ADCM should be deployed, and put the received archive into it.
3. Unpack the archive into the created directory using any of the available methods.

> **NOTE**
>
> Running the command can take some time. Please wait until it is completed.

4. Load the Docker image to the local storage using the received script:

```
$ sudo bash <script_name> unpack_adcm
```

where `<script_name>` is the full script name with the extension (for example, *adcm_2.5.0.sh*).

The example of a successful command execution result:

```
Loaded image: hub.arenadata.io/adcm/adcm:2.5.0
```

5. Create a new container based on the downloaded image:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="
<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e
DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

Step 3. Start ADCM

ADCM is just a container operated by Docker. So you are able to use simple Docker commands to start and stop ADCM:

- To **start ADCM**, use the following command.

```
$ sudo docker start adcm
```

- To **stop ADCM**, use the following command.

```
$ sudo docker stop adcm
```

- In order to provide the **automatic start** of the Docker container after unexpected errors, use the following command.

```
$ sudo docker update --restart=on-failure adcm
```

> **NOTE**
>
> - For more information about Docker restart policies, refer to the Docker documentation.
> - If you allowed your user to access the Docker CLI without `root` privileges, then you can omit the `sudo` keyword from the commands listed above.

Step 4. Check installation

Check the Docker container status

To make sure that the Docker container `adcm` is successfully deployed, run the following command:

```
$ sudo docker container ls
```

The example of the command execution result is listed below. Make sure that the `STATUS` value is equal to `Up`.

```
CONTAINER ID        IMAGE                              COMMAND             CREATED             STATUS              PORTS
NAMES
74b5628146e5        hub.arenadata.io/adcm/adcm:2.0   "/etc/startup.sh"   4 days ago          Up 21 minutes       0.0.0.0:8000-
>8000/tcp   adcm
```

Check the 8000 port availability

After ADCM is installed and started, its web interface should become available on the `8000` port of the deployed container.

For Ubuntu 22.04, first install the packages required for executing the `netstat` command:

```
$ sudo apt-get install net-tools
```

You can check the 8000 port availability using these commands:

- The `netstat` command — to display information about port listening.

```
$ sudo netstat -ntpl | grep 8000
```

The example of the command execution result:

```
tcp6       0      0 :::8000                 :::*                    LISTEN
1514/docker-proxy-c
```

- The `curl` command — to check the URL connection.

```
$ curl http://localhost:8000
```

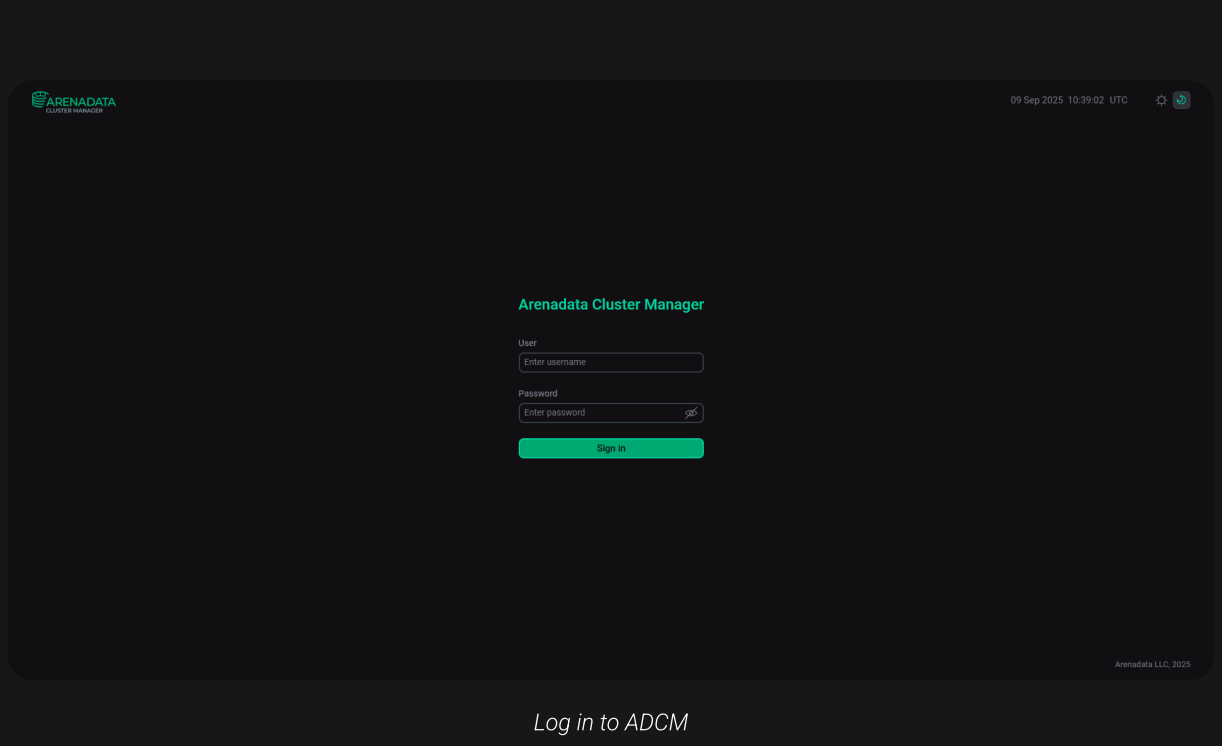
The example of the command execution result:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Arenadata Cluster Manager</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="assets/favicon.ico">
<link rel="stylesheet" href="styles.10db6328264e0907c52f.css"></head>
<body>
  <app-root></app-root>
<script src="runtime-es2015.7eb406ed18bf0258cd35.js" type="module"></script><script src="runtime-es5.7eb406ed18bf0258cd35.js"
nomodule defer></script><script src="polyfills-es5.2e224d70daec4412d3c2.js" nomodule defer></script><script src="polyfills-es2015.
37e0553ac06970d6a5b5.js" type="module"></script><script src="main-es2015.39851da0ebf9ed6fec45.js" type="module"></script><script
src="main-es5.39851da0ebf9ed6fec45.js" nomodule defer></script></body>
</html>
```

Check the ADCM web-interface

To log in to ADCM, use the user with administrator rights that is created by default:

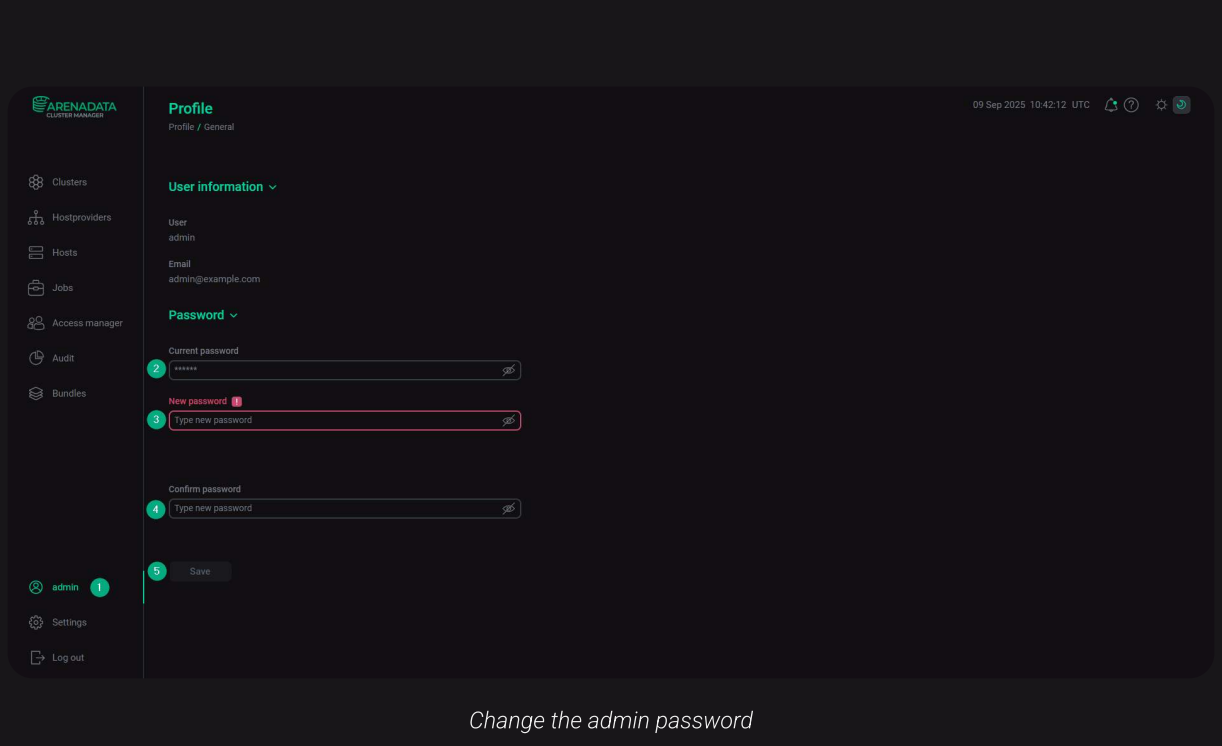
- ***User***: admin
- ***Password***: admin



Log in to ADCM

Later you can change the password if necessary. To do this:

1. Open the user profile in the left navigation menu.
2. Enter the current password in the ***Current password*** field.
3. Fill in the ***New password*** field.
4. Confirm a new password in the ***Confirm password*** field.
5. Click ***Save***.

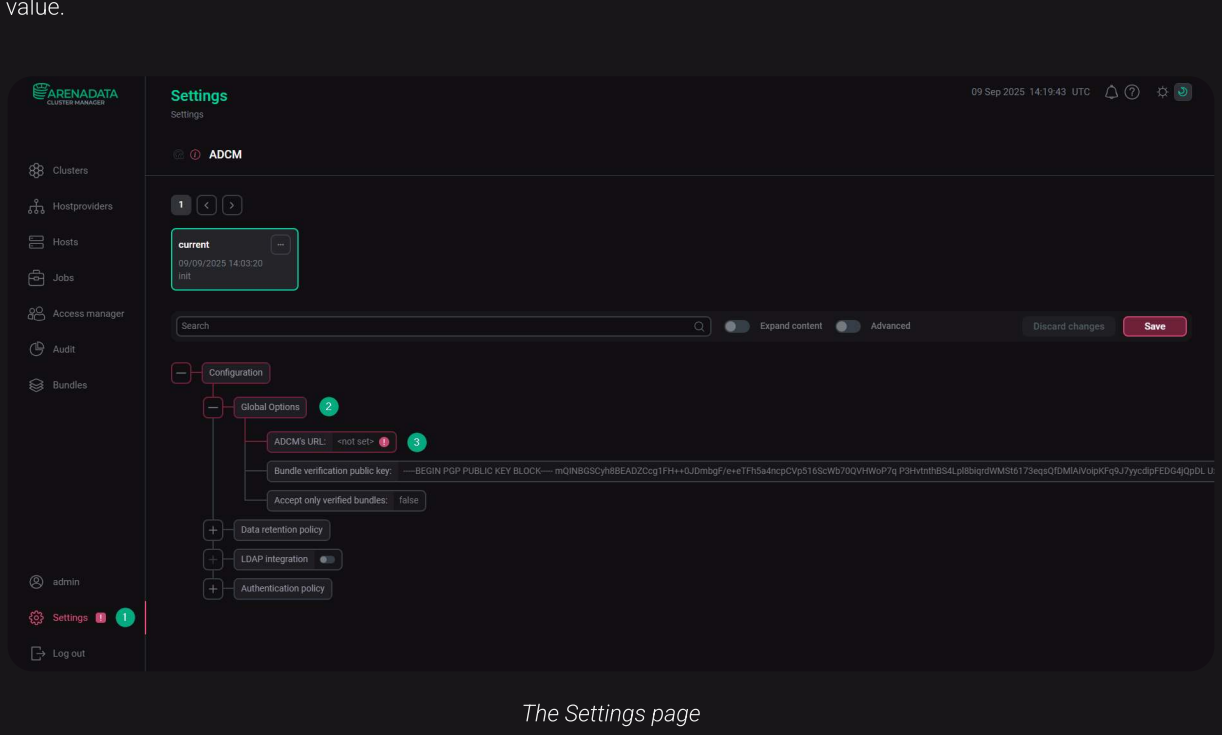


Change the admin password

Set the ADCM URL

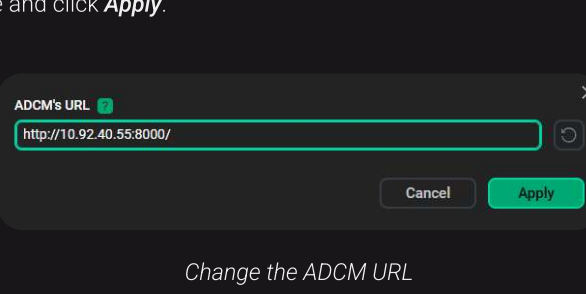
For successful communication between ADCM and product clusters deployed via ADCM, it is important to correctly define the external ADCM address. This address is used by cluster components for sending information about their state to ADCM. To specify the ADCM URL, use the environment variable `DEFAULT_ADCM_URL` when creating the ADCM container. If you did not specify that variable, you should manually install the URL in the ADCM web interface. To do this, follow the steps:

1. Select the ***Settings*** item in the left navigation menu.
2. Expand the ***Global Options*** node in the tree of configuration settings.
3. Click the ***ADCM's URL*** field value.



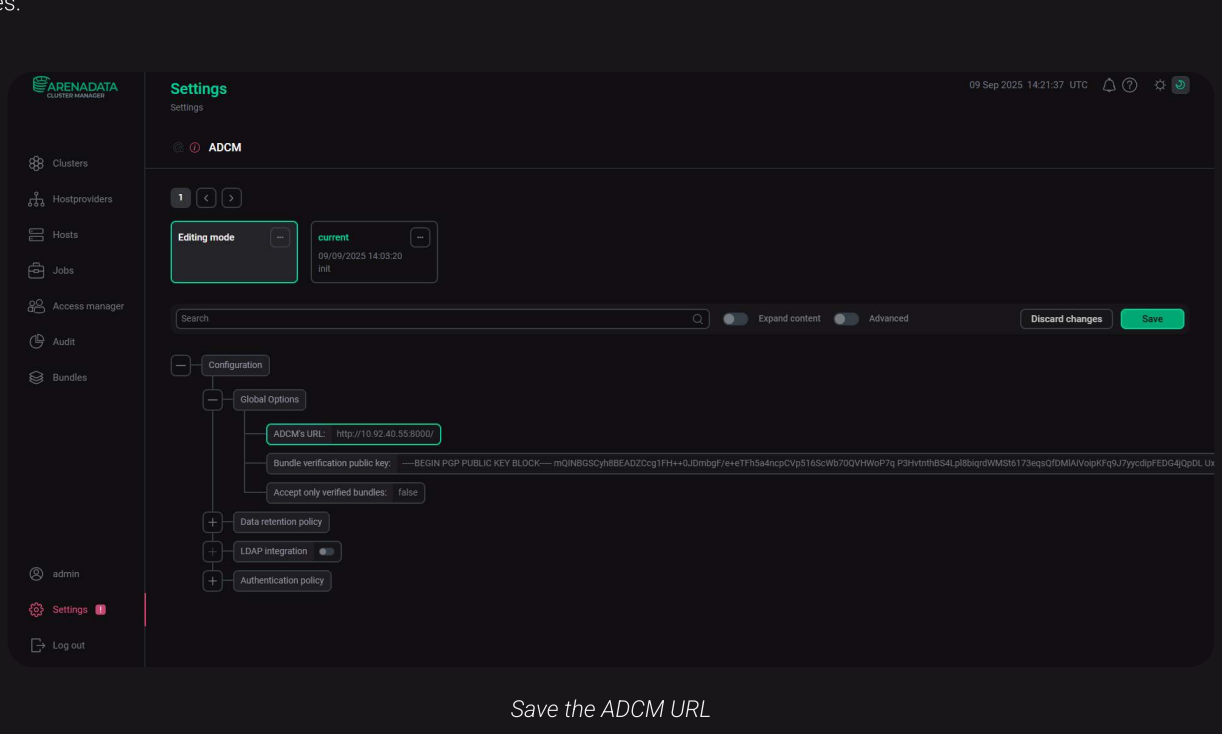
The Settings page

4. In the window that opens, edit the ***ADCM's URL*** field value and click ***Apply***.



Change the ADCM URL

5. Click ***Save*** to apply changes.



Save the ADCM URL

Step 5. Update ADCM

> **CAUTION**
>
> - Before upgrading, it is strongly recommended to back up ADCM data. For more information, see Backup and restore.
> - Downgrade of the ADCM version is not supported.
> - It is not recommended to update ADCM more than 5 minor versions ahead (see `<minor>` in the versioning format below).
> - Before upgrading ADCM to version 2.7.0, migrate data from SQLite to PostgreSQL. Migration can be performed in any ADCM version (up to 2.6.0).

To upgrade the previously installed ADCM version offline, follow these steps:

1. Stop the `adcm` container:

```
$ sudo docker stop adcm
```

2. Remove the `adcm` container:

```
$ sudo docker container rm adcm
```

> **NOTE**
>
> For customers who use Enterprise editions of products, it is possible to contact the Arenadata support team to get an archive with the *sh.xz* extension that contains a special installation shell script. After stopping the `adcm` container, the ADCM update process includes the installation according to the instruction.

3. Download the required Docker image from the Arenadata Docker Registry on a host that has Internet access:

```
$ sudo docker pull hub.arenadata.io/adcm/adcm:<version>
```

where `<version>` is a desired ADCM image version in one of the following formats:

- `<major>.<minor>.<patch>` — if you need a specific ADCM patch. For example, `2.0.0`.
- `<major>.<minor>` — if you need the last patch within the selected ADCM version. For example, `2.0`.

> **IMPORTANT**
>
> - `latest` was used for earlier versions of ADCM and is no longer supported starting from version `2.0.0`.
> - Before ADCM 2.0.0, the following format was used for versioning: `YYYY.MM.DD.HH`.

4. Save the Docker image to a *.tar* file:

```
$ sudo docker save -o <file_name> hub.arenadata.io/adcm/adcm:<version>
```

where `<file_name>` is the full name of the created file with the extension (for example, *adcm_2.5.0_arenadata.tar*).

> **NOTE**
>
> Running the command can take some time. Please wait until it is completed.

5. Copy the saved file from the current host to a host without Internet connection using any available methods.

> **IMPORTANT**
>
> Make sure you have configured access to hosts and necessary file permissions.

6. Load the Docker image from the *.tar* file:

```
$ sudo docker load -i <file_path>
```

where `<file_path>` is the absolute path to the *.tar* file with the Docker image.

The example of a successful command execution result:

```
Loaded image: hub.arenadata.io/adcm/adcm:2.5.0
```

7. Create a new container based on the downloaded image:

```
$ sudo docker create --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="<DATABASE_HOSTNAME_OR_IP_ADDRESS>" -e DB_PORT="
<DATABASE_TCP_PORT>" -e DB_USER="<DATABASE_USERNAME>" -e DB_NAME="<DATABASE_NAME>" -e DB_PASS="<DATABASE_USER_PASSWORD>" -e
DB_OPTIONS="<DATABASE_OPTIONS>" hub.arenadata.io/adcm/adcm:<version>
```

> **NOTE**
>
> For more information on the environment variables (`DB_HOST`, `DB_PORT`, `DB_USER`, `DB_NAME`, `DB_PASS`, and `DB_OPTIONS`), see the Installation article.

8. Start the new `adcm` container:

```
$ sudo docker start adcm
```

Prepare hosts

Contents

To Table of Contents

User permissions for ADH installation

To manage a cluster and its nodes via ADCM, it is necessary to prepare hosts for a cluster and link these hosts to ADCM. This is the function of ADCM **hostproviders** — special plugins for ADCM, which encapsulate the complex mechanism of interaction between ADCM and the hosts controlled by it.

ADCM supports several types of hostproviders. For more information on how to configure hostproviders and use them to create hosts, refer to the following documentation:

- SSH
- Yandex Cloud
- VCloud (VMware)
- Cloud.ru Advanced
- CROC Cloud

For offline installation, the SSH hostprovider is recommended.

CAUTION

When creating ADH hosts via ADCM, you should specify their FQDNs. The maximum FQDN length is 38 symbols.

User permissions for ADH installation

If you specify a custom UNIX user in the hostprovider settings, which will be used to run ADH installation commands, make sure that the specified user can run commands as a superuser.

To grant the user the necessary rights to install ADH, do the following:

1. Grant the `root` privileges for the user:

CentOS 7/RHEL 7/RED OS 7.3/Alt Linux SP 8

```
$ sudo usermod -aG wheel <user>
```

where `<user>` is a username.

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
$ sudo usermod -aG sudo <user>
```

where `<user>` is a username.

2. Make sure that the */etc/sudoers* file contains the record specified below depending on the operating system with the following command:

```
$ sudo vi /etc/sudoers
```

To return to the command line, press `Esc`, type `:q!`, and press `Enter`.

CentOS 7/RHEL 7/RED OS 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Create a cluster

Daria Barysheva

Contents

To Table of Contents

Step 1. Get the Enterprise version of a cluster bundle

Step 2. Upload a cluster bundle to ADCM

Step 3. Create a cluster based on the uploaded bundle

In ADCM a **cluster** means a set of services running on a set of hosts.

Clusters distributions for ADCM are made in the form of so-called bundles. Generally a bundle is a regular archive that includes a description and software logic of cluster deployment and exploitation.

The steps for adding a cluster to ADCM are listed below.

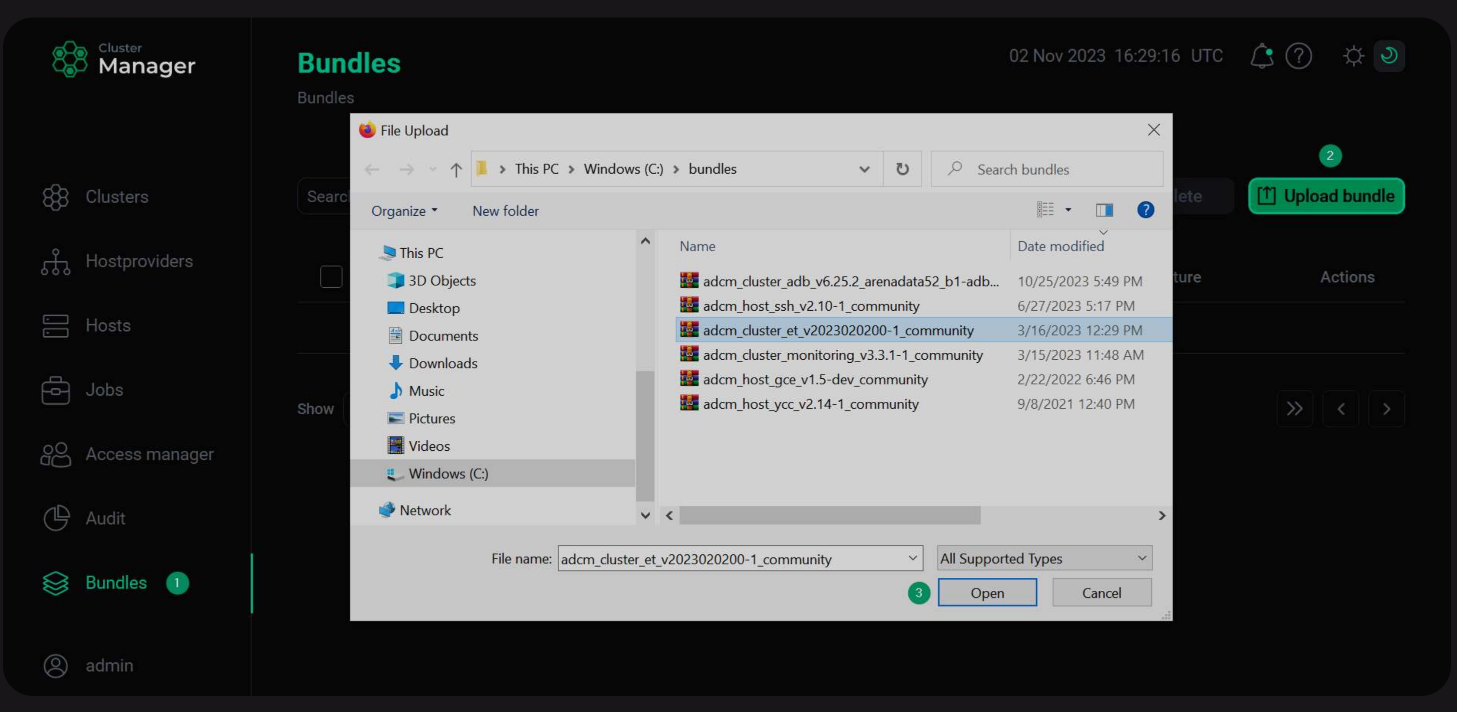
Step 1. Get the Enterprise version of a cluster bundle

Please get a cluster bundle from the Arenadata support team.

Step 2. Upload a cluster bundle to ADCM

To upload a bundle to ADCM, follow the steps:

1. Select the ***Bundles*** item in the left navigation menu and click ***Upload bundle***.
2. Select a bundle in the Open File Dialog.



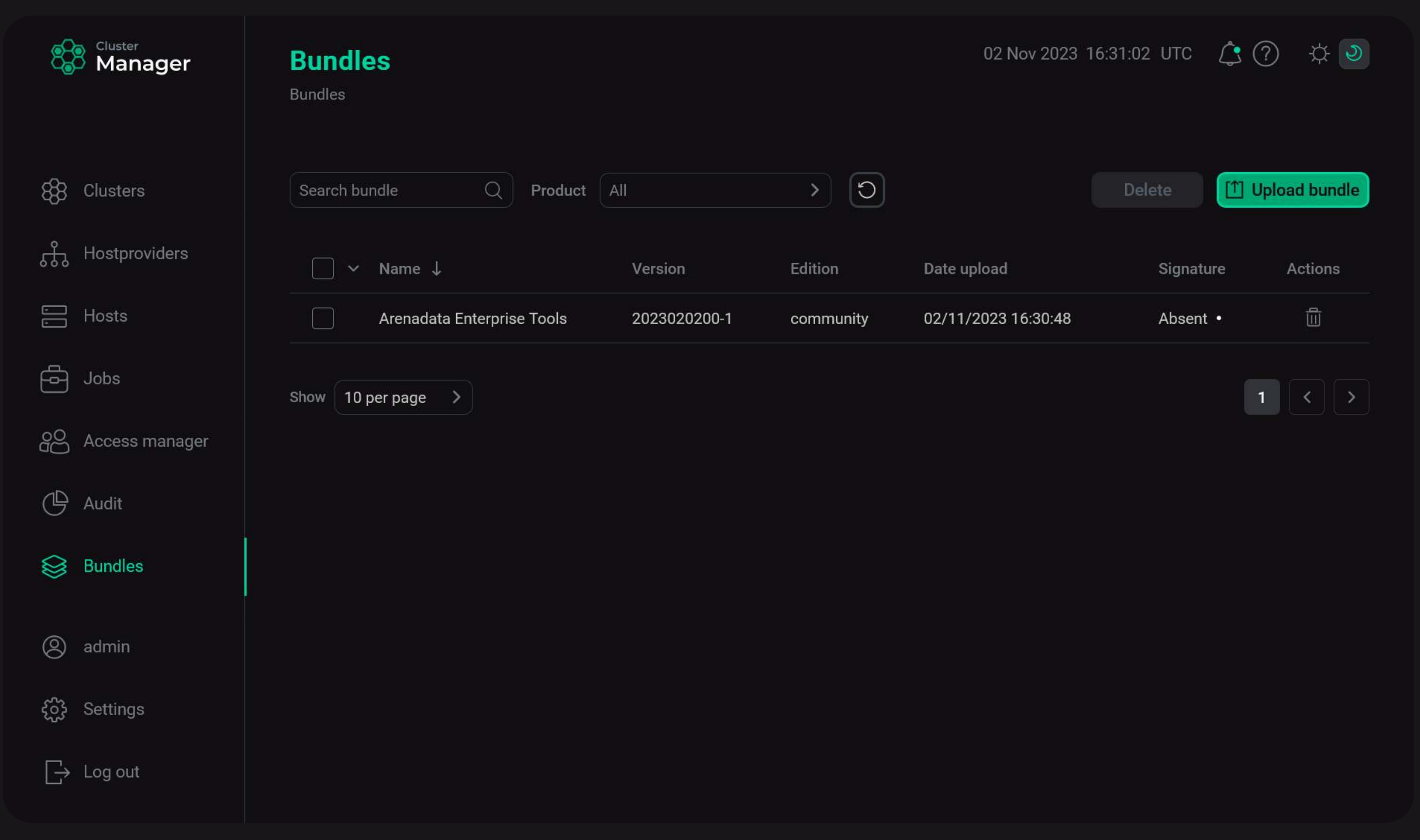
Cluster Manager
Bundles
Bundles
02 Nov 2023 16:29:16 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles 1
admin
Upload bundle 2
File Upload
This PC > Windows (C:) > bundles
Search bundles
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
Network
All Supported Types
3 Open
Cancel

Name	Date modified
adcm_cluster_adb_v6.25.2_arenadata52_b1-adb...	10/25/2023 5:49 PM
adcm_host_ssh_v2.10-1_community	6/27/2023 5:17 PM
adcm_cluster_et_v2023020200-1_community	3/16/2023 12:29 PM
adcm_cluster_monitoring_v3.3.1-1_community	3/15/2023 11:48 AM
adcm_host_gce_v1.5-dev_community	2/22/2022 6:46 PM
adcm_host_ycc_v2.14-1_community	9/8/2021 12:40 PM

File name: adcm_cluster_et_v2023020200-1_community

Upload a bundle

3. As a result of the performed actions, a bundle is displayed on the ***Bundles*** page.



Cluster Manager
Bundles
Bundles
02 Nov 2023 16:31:02 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search bundle
Product
All
Delete
Upload bundle
Show
10 per page
1

Name	Version	Edition	Date upload	Signature	Actions
Arenadata Enterprise Tools	2023020200-1	community	02/11/2023 16:30:48	Absent •	

The result of successful uploading a bundle

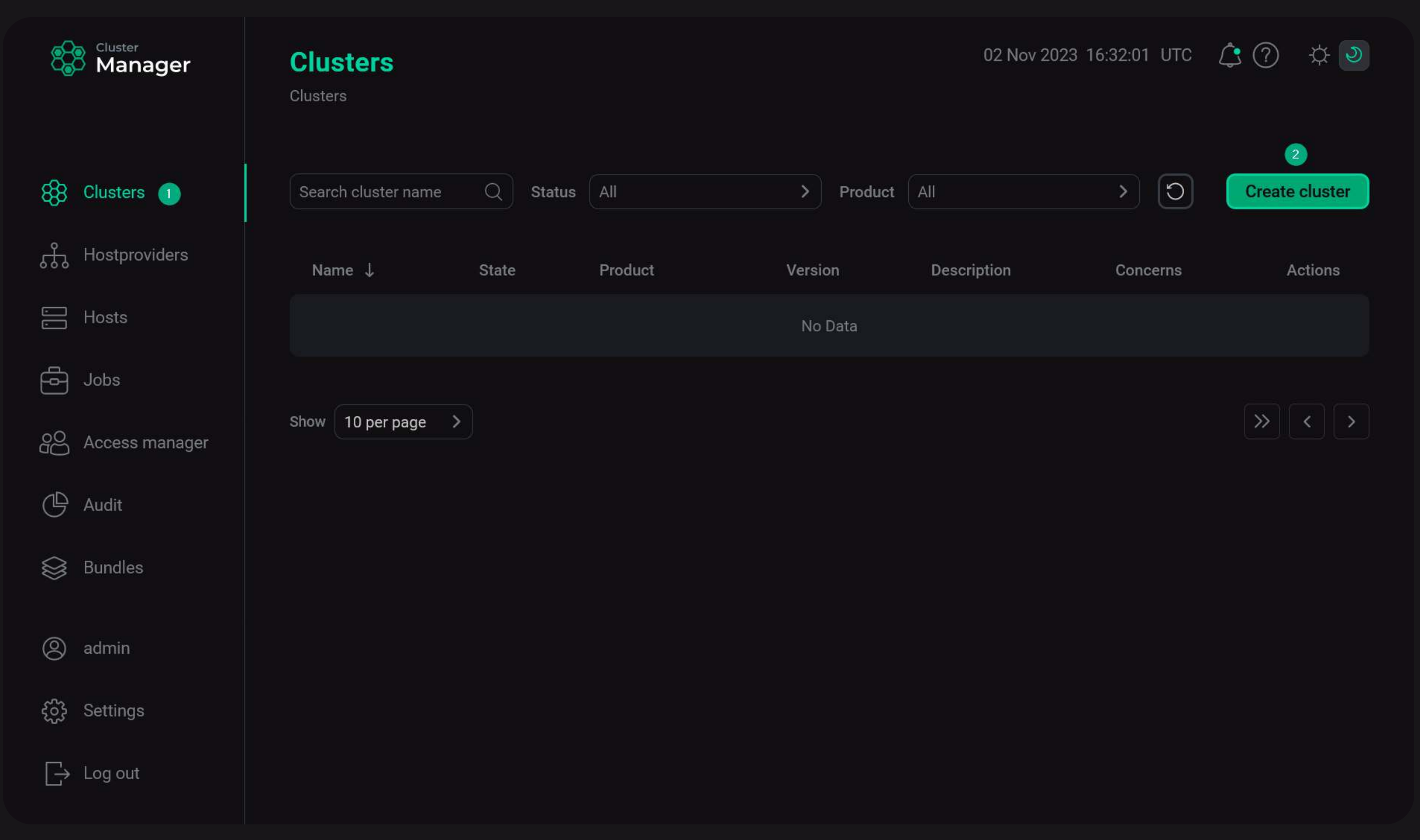
Step 3. Create a cluster based on the uploaded bundle

IMPORTANT

When creating a cluster via the ADCM web interface, you generate a new ADCM cluster instance. It means only adding the cluster to ADCM — at this stage no services are deployed.

The steps for cluster creation are given below:

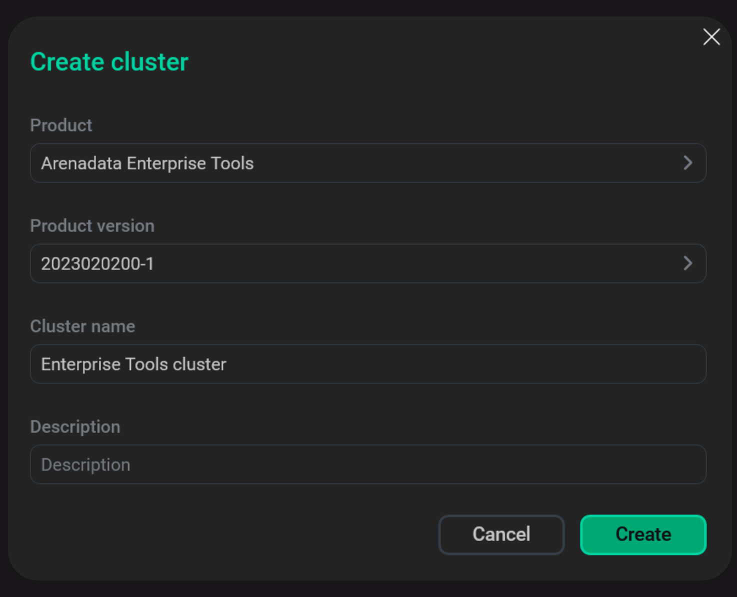
1. Select the ***Clusters*** item in the left navigation menu and click ***Create cluster***.



Cluster Manager
Clusters
Clusters
02 Nov 2023 16:32:01 UTC
Clusters 1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
2
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
No Data
Show
10 per page

Switch to cluster creation

2. In the opened dialog, describe a new cluster instance:
   - Select an uploaded cluster bundle in the ***Product*** field.
   - Select a bundle version in the ***Product version*** field. Several versions become available in the case of different versions of the same bundle being uploaded.
   - Enter a cluster name in the ***Cluster name*** field.
   - Enter a cluster description in the ***Description*** field if necessary.
   - Set the ***I accept Terms of Agreement*** flag to sign the license agreement. This flag is displayed only for Enterprise bundles. To read the agreement text, refer to the ***Terms of Agreement*** link. You can also sign the license agreement on the ***Bundles*** page by clicking the desired product bundle.
   - Click ***Create***.



Create cluster
Description
Cancel
Create

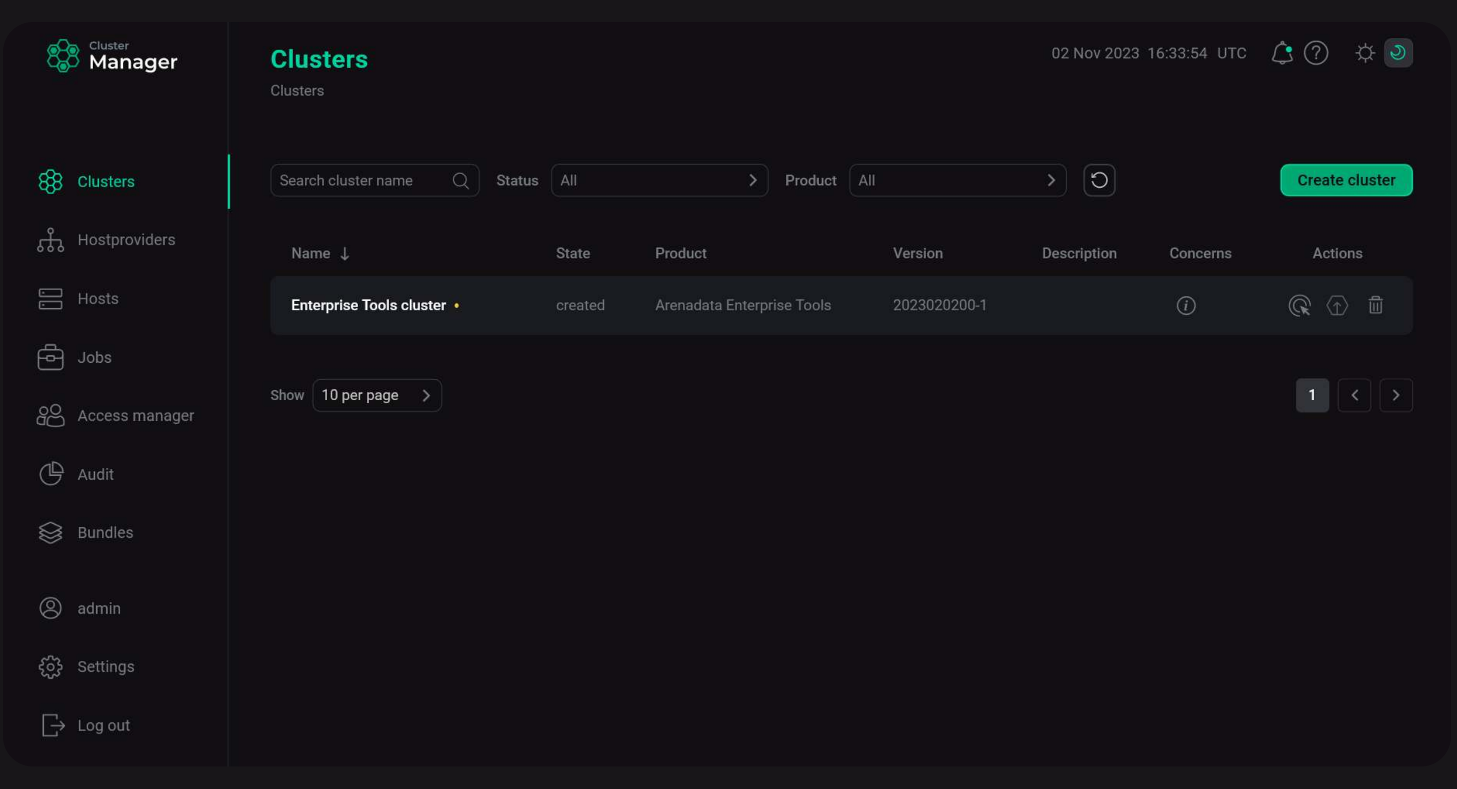
Product: Arenadata Enterprise Tools

Product version: 2023020200-1

Cluster name: Enterprise Tools cluster

Create a cluster

3. As a result, the created cluster instance is displayed on the ***Clusters*** page.



Cluster Manager
Clusters
Clusters
02 Nov 2023 16:33:54 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster •	created	Arenadata Enterprise Tools	2023020200-1			

The result of successful cluster creation

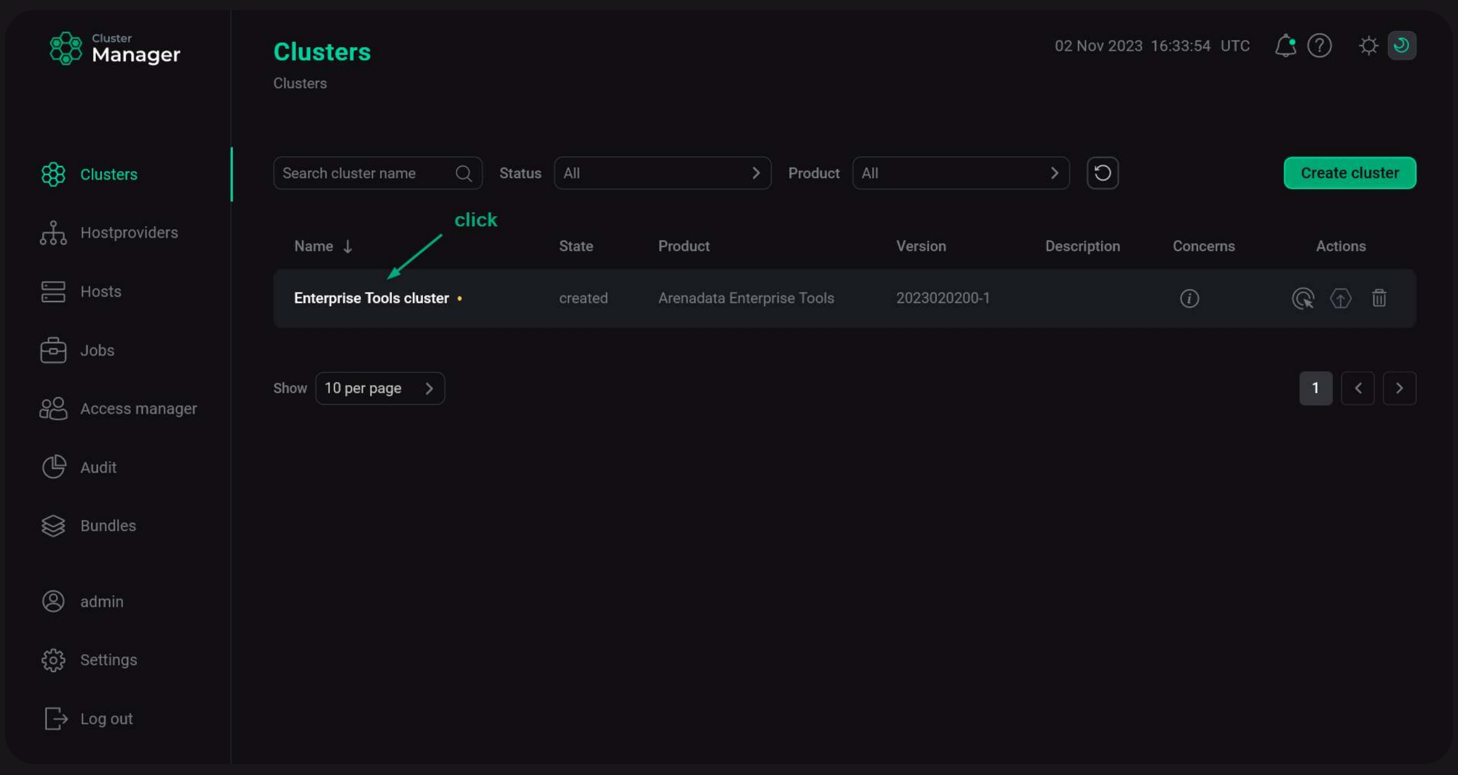
Add services

Daria Barysheva

To Table of Contents

In ADCM a **service** means a software that performs some function. The steps for adding services to a cluster are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.

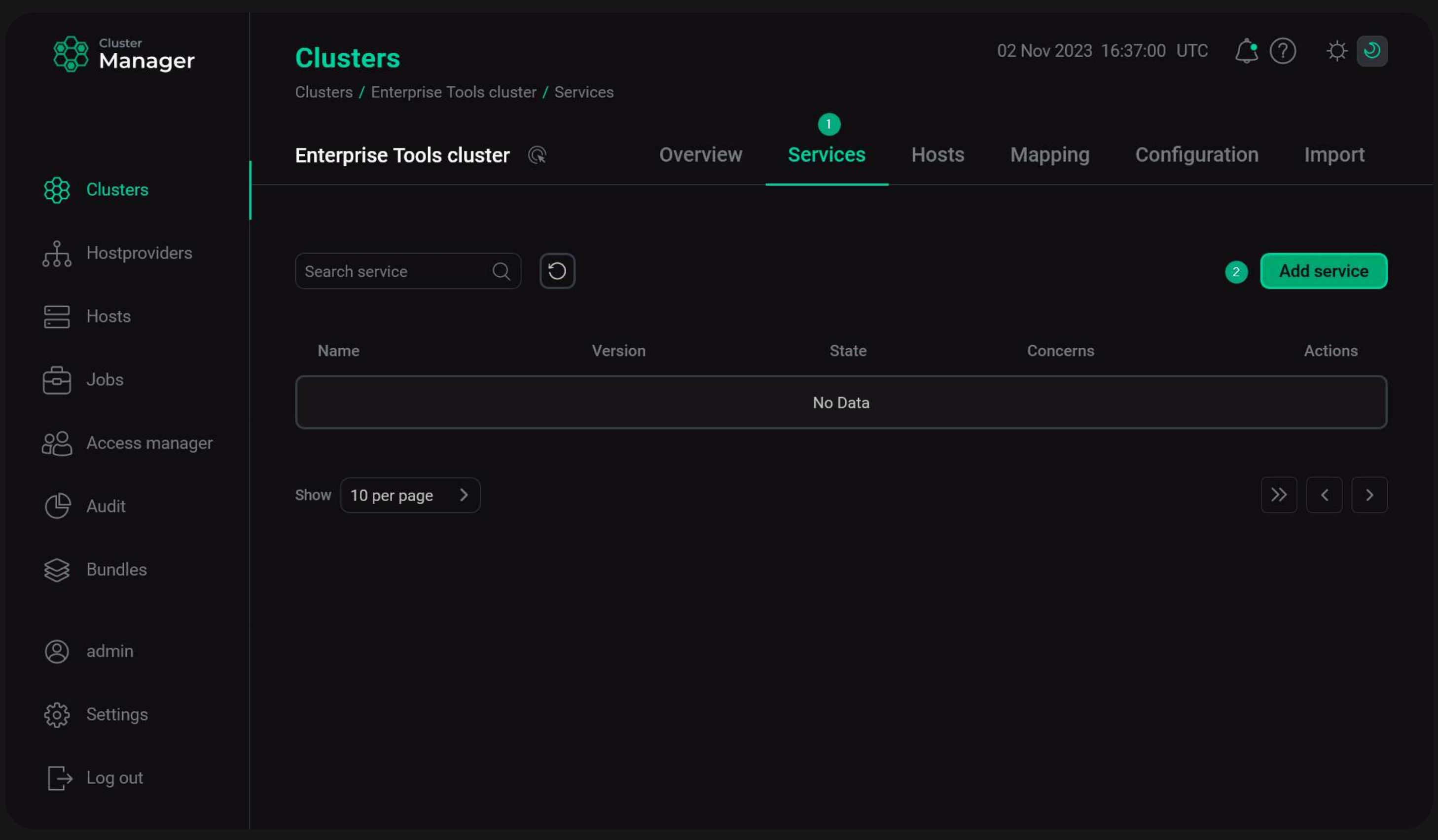


Cluster Manager
Clusters
02 Nov 2023 16:33:54 UTC
Clusters
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name ↓	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	created	Arenadata Enterprise Tools	2023020200-1			

Select a cluster

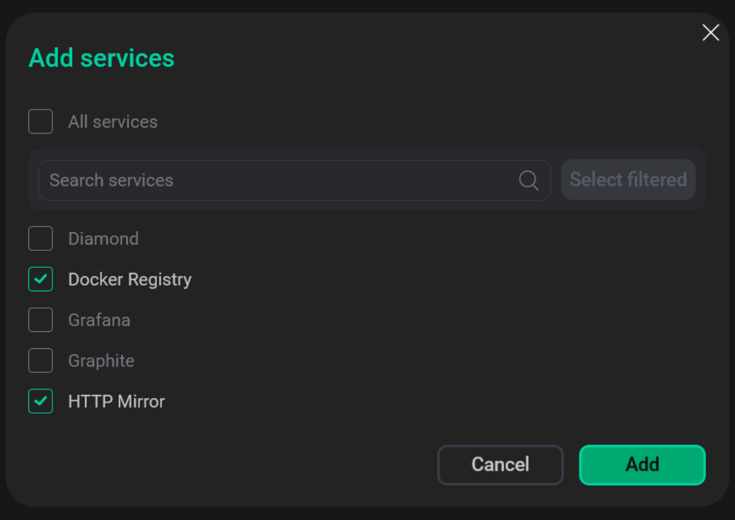
2. Open the ***Services*** tab on the cluster page and click ***Add service***.



Cluster Manager
Clusters
02 Nov 2023 16:37:00 UTC
Clusters / Enterprise Tools cluster / Services
Enterprise Tools cluster
1
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Switch to adding services

3. In the opened dialog, select services that should be added to the cluster and click ***Add***.



Add services
☐ All services
Search services
Select filtered
☐ Diamond
☑ Docker Registry
☐ Grafana
☐ Graphite
☑ HTTP Mirror
Cancel
Add

Select services

The brief description of available services is listed below.

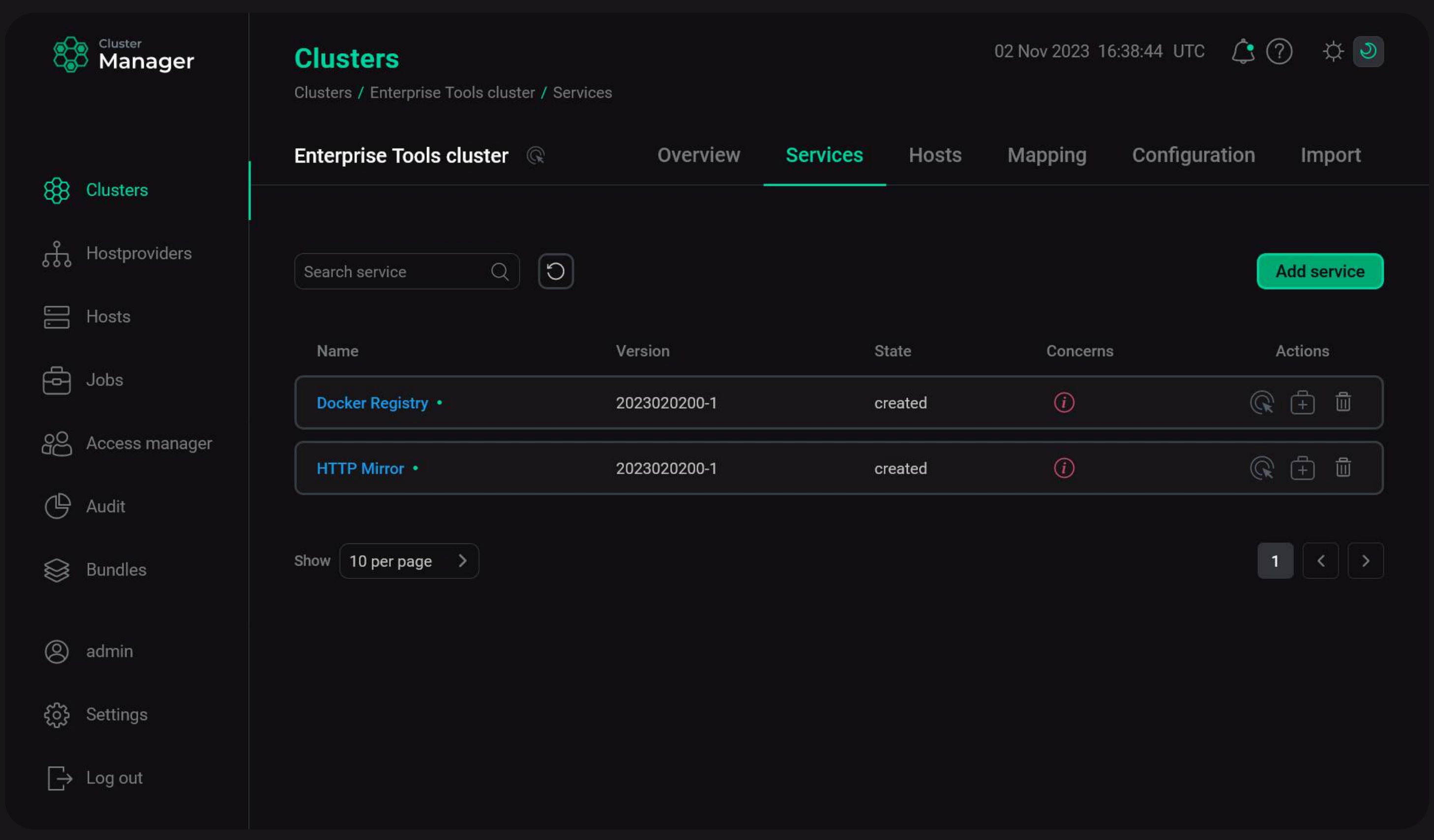
Services that should be added to the Enterprise Tools cluster

Service	Purpose
HTTP Mirror	A service used for hosting packages of the repositories that are needed for offline installation of products
Docker Registry	A service used to store Docker Images that are needed for offline installation of products

NOTE

The CoreDNS service is deprecated — do not select it. The Diamond, Graphite, and Grafana services are used for building monitoring — they will be described separately.

4. As a result, the added services are displayed on the ***Services*** tab.



Cluster Manager
Clusters
02 Nov 2023 16:38:44 UTC
Clusters / Enterprise Tools cluster / Services
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
Docker Registry	2023020200-1	created		
HTTP Mirror	2023020200-1	created		

The result of successful adding services to a cluster

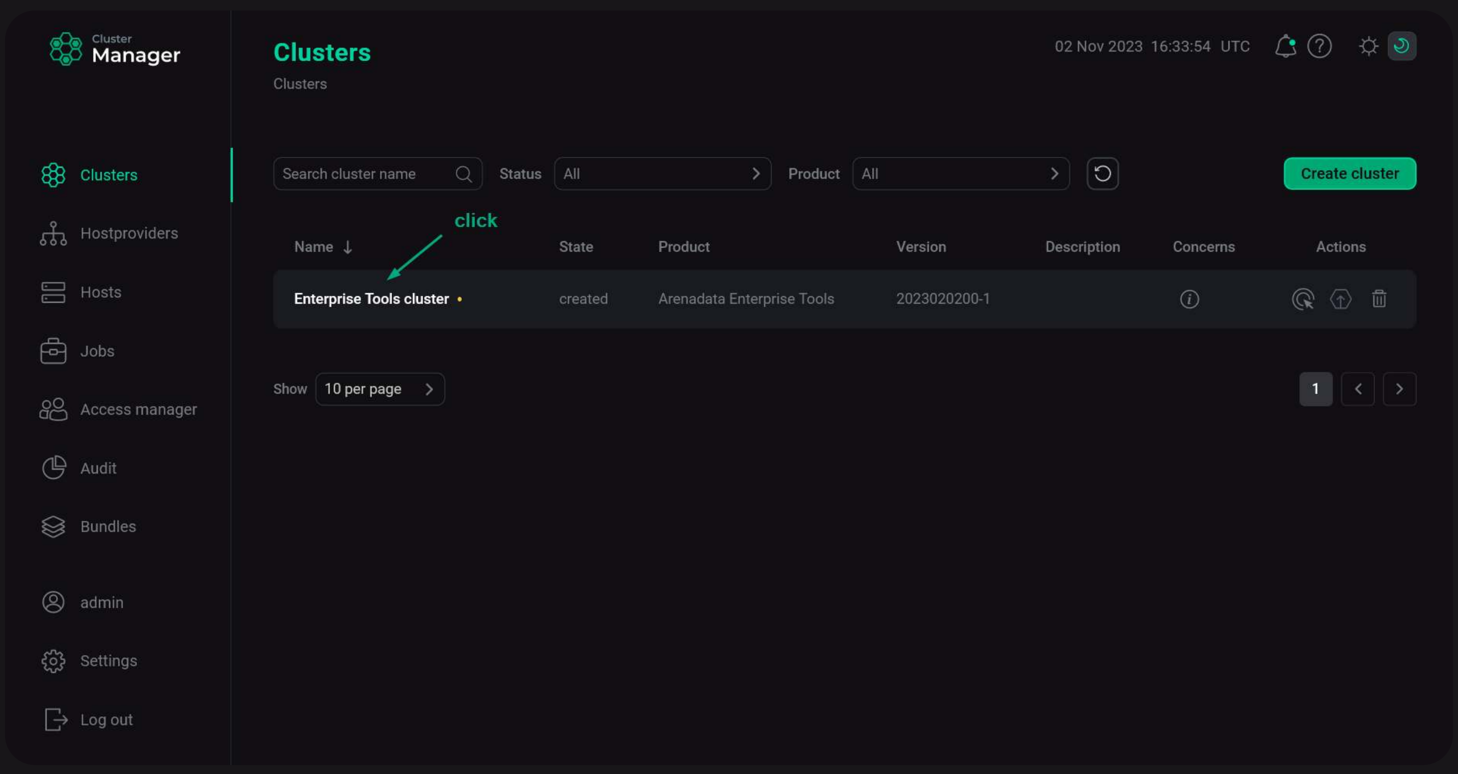
Add hosts to a cluster

Daria Barysheva

To Table of Contents

The steps for adding hosts to a cluster are listed below:

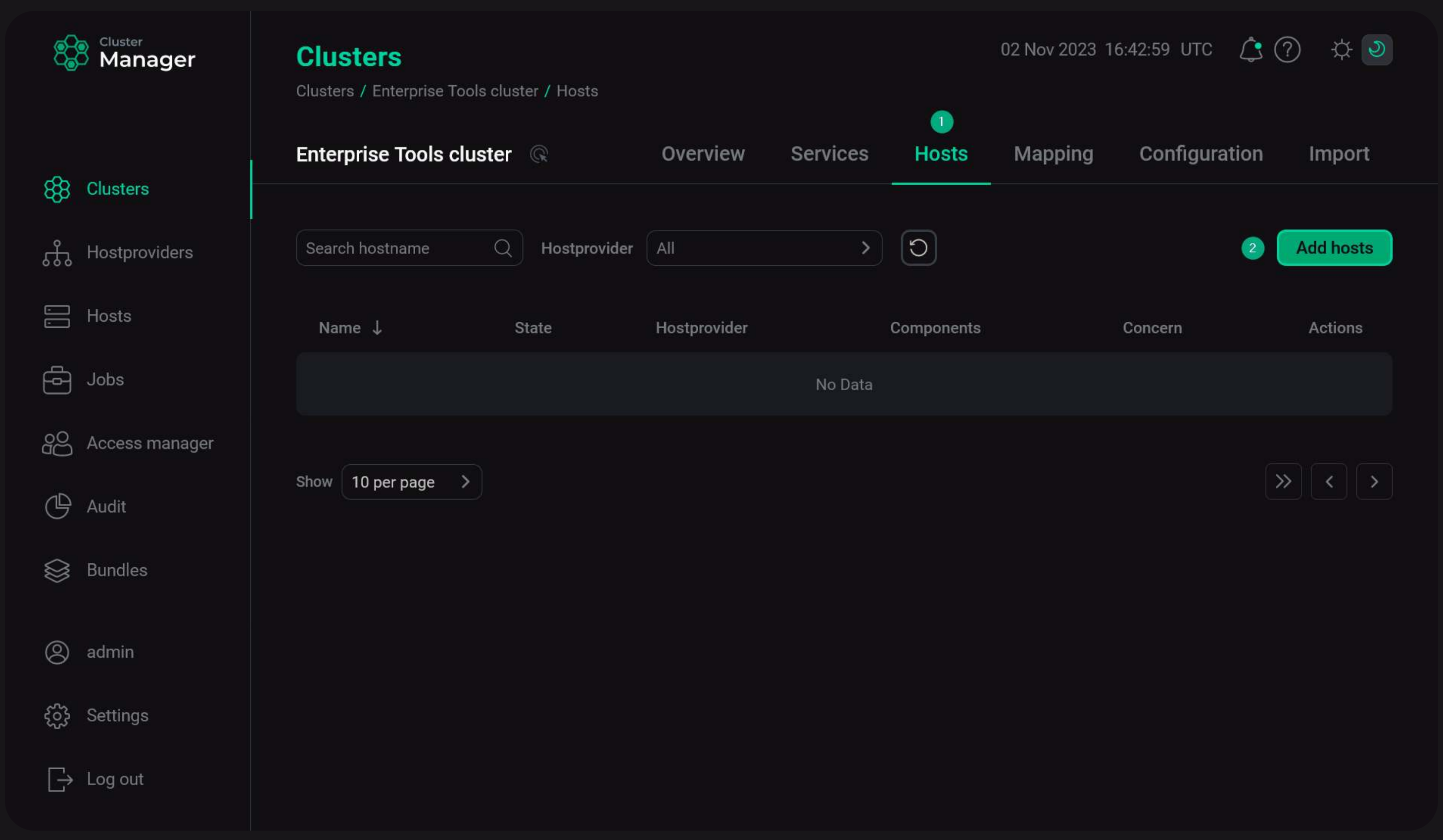
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Nov 2023 16:33:54 UTC
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	created	Arenadata Enterprise Tools	2023020200-1			

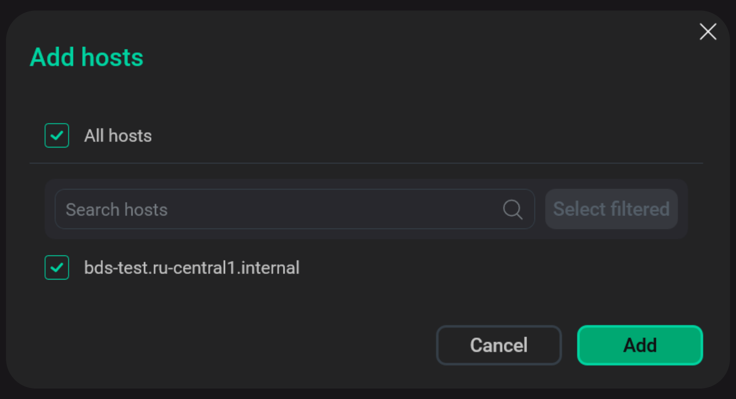
Select a cluster

2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Hosts
02 Nov 2023 16:42:59 UTC
Enterprise Tools cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
2
Add hosts
Name
State
Hostprovider
Components
Concern
Actions
No Data
Show
10 per page

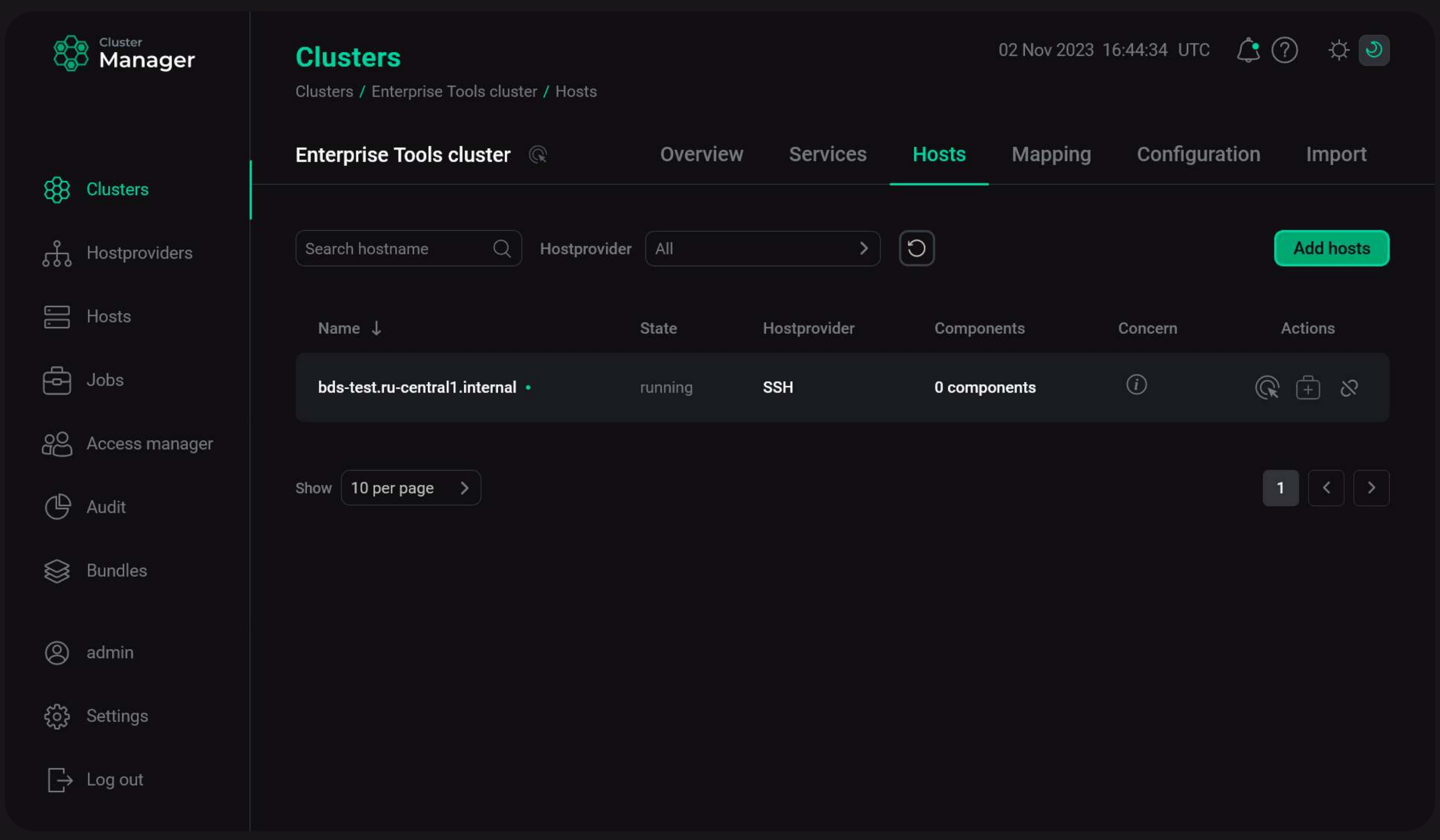
Switch to adding hosts

3. In the opened dialog, select hosts that should be added to the cluster and click ***Add***.


Add hosts
☑ All hosts
Search hosts
Select filtered
☑ bds-test.ru-central1.internal
Cancel
Add

Select hosts

4. As a result, the added hosts are displayed on the ***Hosts*** tab.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Enterprise Tools cluster / Hosts
02 Nov 2023 16:44:34 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Add hosts
Show
10 per page
1

Name	State	Hostprovider	Components	Concern	Actions
bds-test.ru-central1.internal	running	SSH	0 components		

The result of successful adding hosts to a cluster

Add components

Daria Barysheva

To Table of Contents

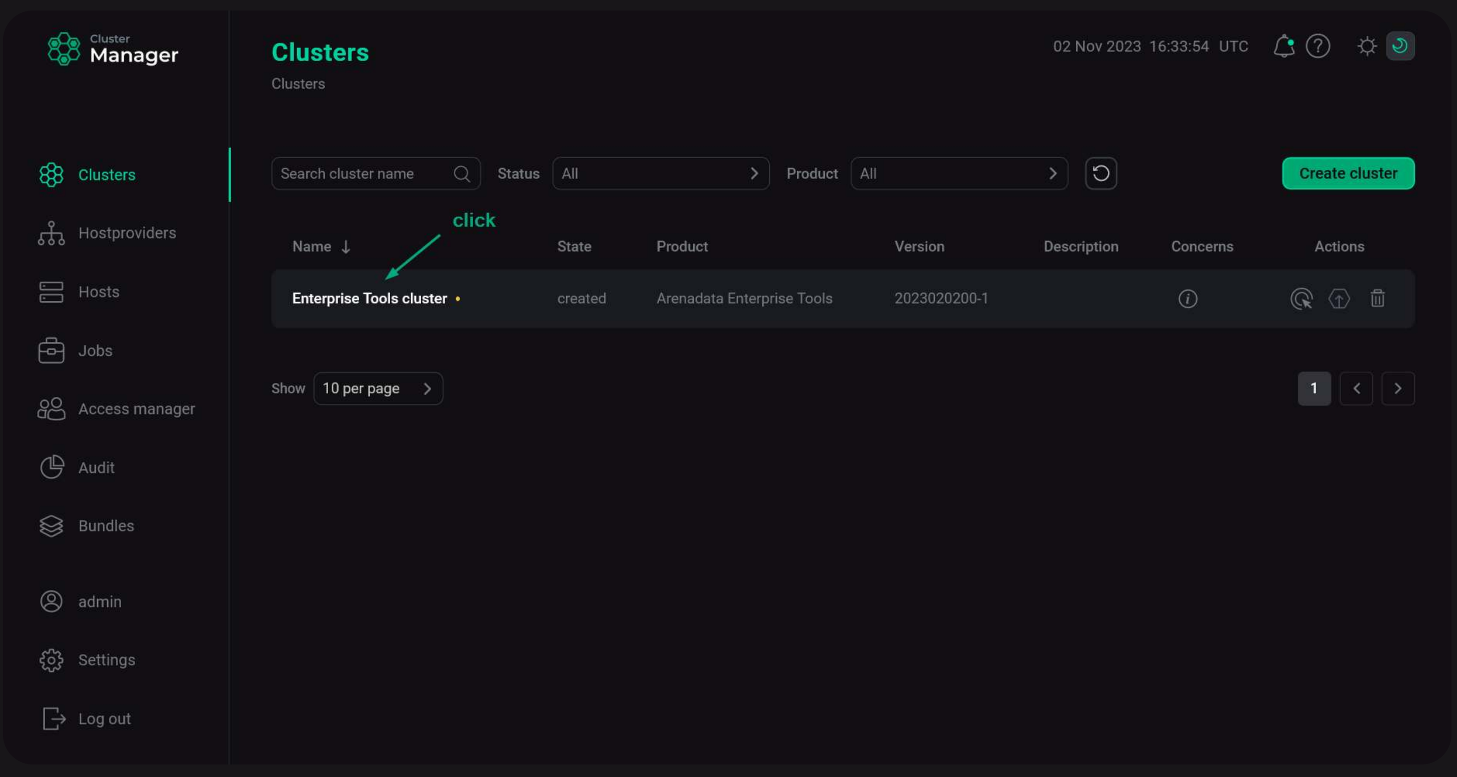
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts.

IMPORTANT

Initially, there are no components on any of cluster hosts. You should map components to hosts manually.

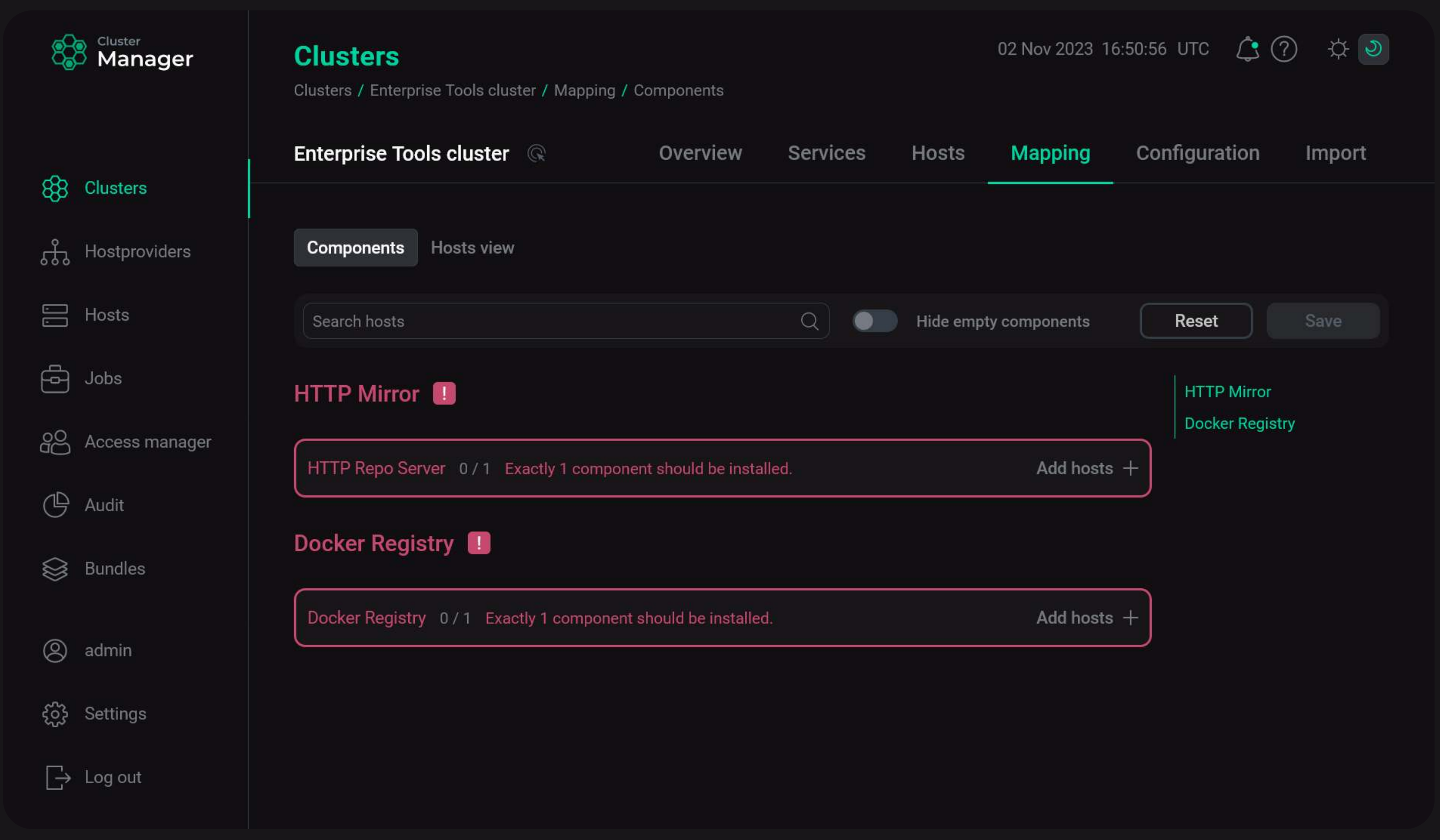
The steps for adding components to hosts are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
02 Nov 2023 16:33:54 UTC
Search cluster name
Status All
Product All
Create cluster
click
Name
State
Product
Version
Description
Concerns
Actions
Enterprise Tools cluster
created
Arenadata Enterprise Tools
2023020200-1
Show 10 per page
1

Select a cluster

2. Open the ***Mapping*** tab on the cluster page.

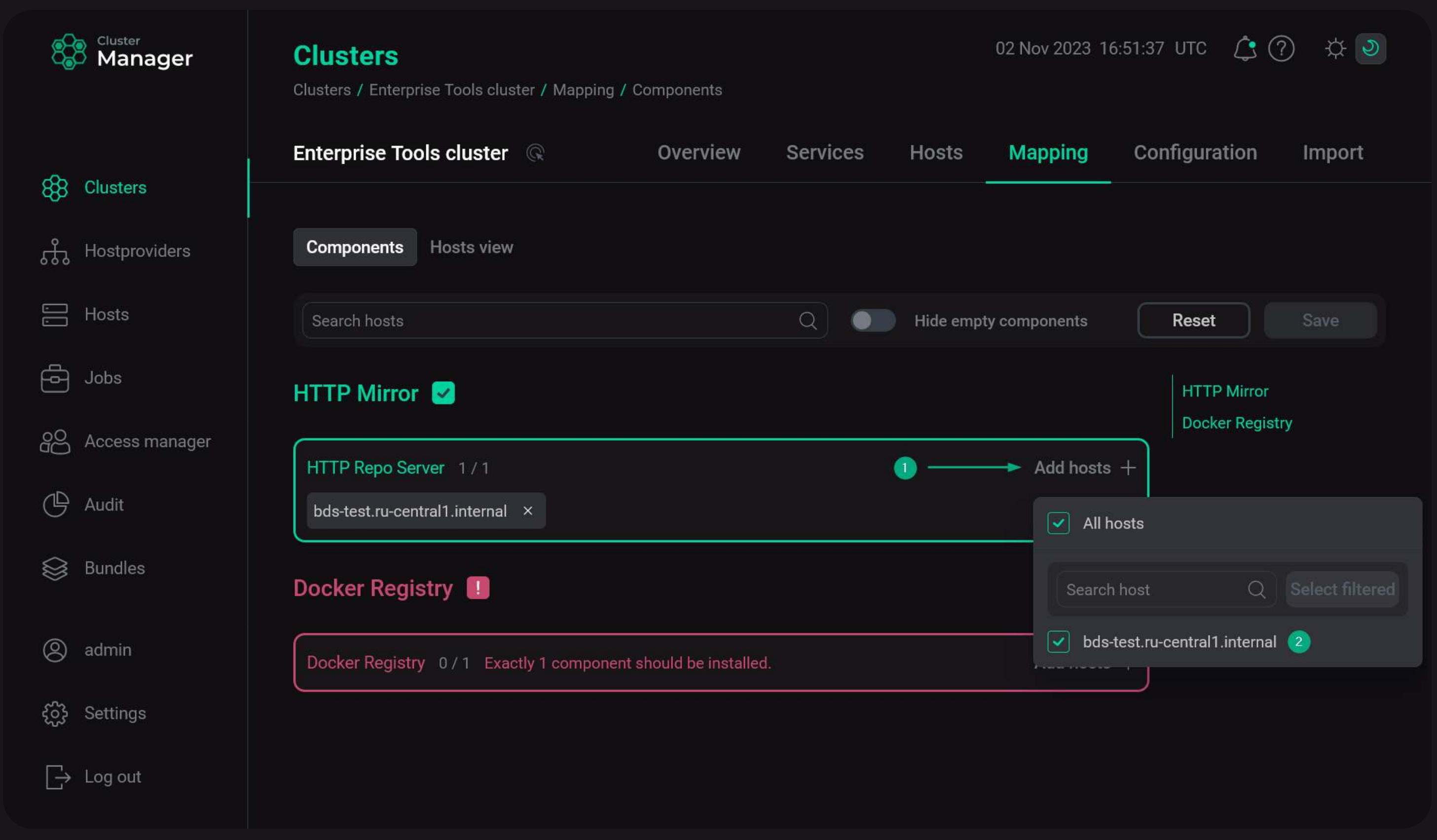

Cluster Manager
Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 16:50:56 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
HTTP Mirror
HTTP Repo Server 0 / 1
Add hosts
Docker Registry
Docker Registry 0 / 1
Add hosts
HTTP Mirror
Docker Registry

HTTP Repo Server 0 / 1 Exactly 1 component should be installed.

Docker Registry 0 / 1 Exactly 1 component should be installed.

Switch to mapping service components

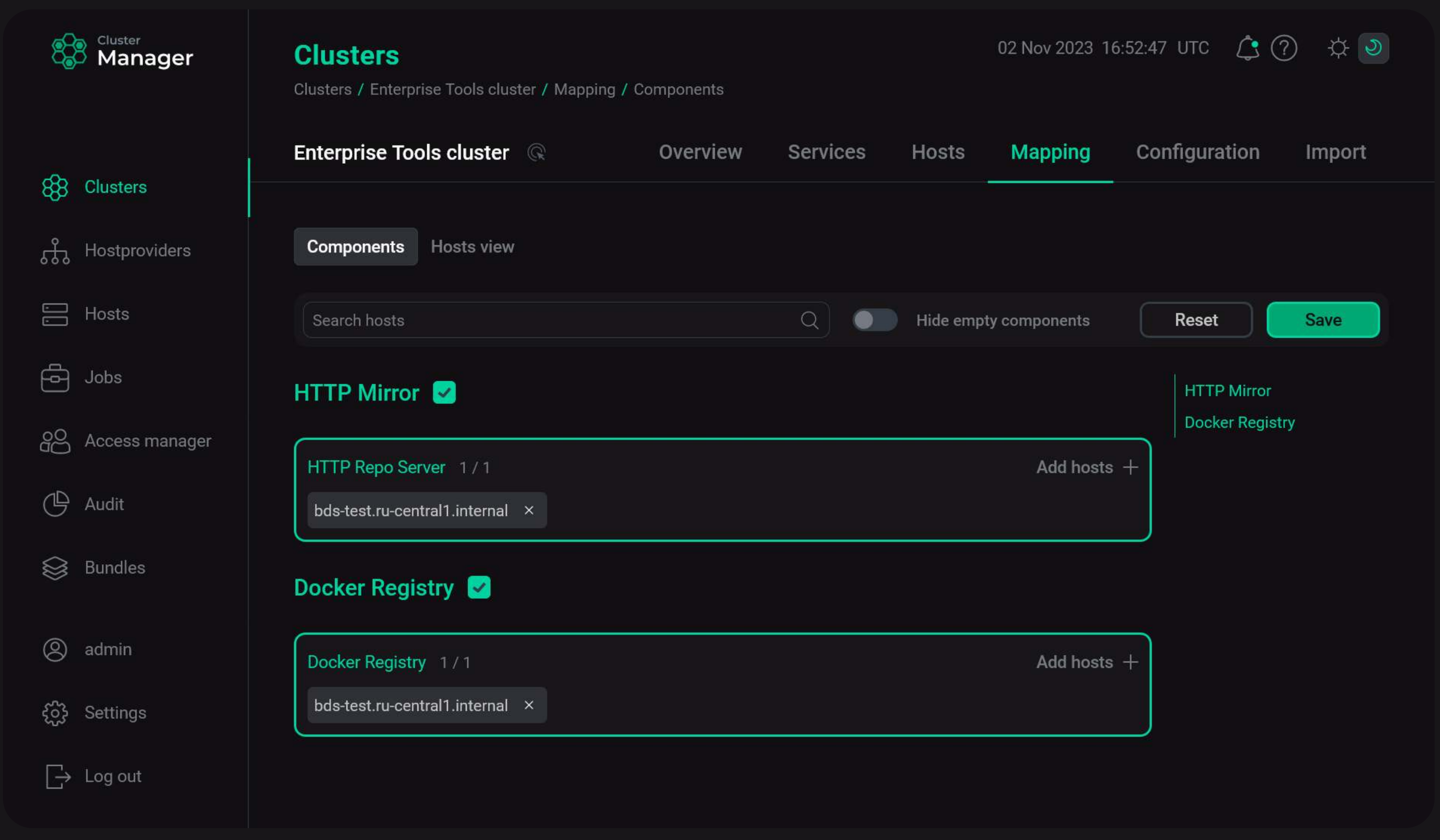
3. In the window that opens, map components to hosts. To assign hosts for each component, click ***Add hosts*** and select hosts in the pop-up window.


Cluster Manager
Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 16:51:37 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
HTTP Mirror
HTTP Repo Server 1 / 1
bds-test.ru-central1.internal
1
Add hosts
All hosts
Search host
Select filtered
bds-test.ru-central1.internal
2
Docker Registry
Docker Registry 0 / 1
HTTP Mirror
Docker Registry

Docker Registry 0 / 1 Exactly 1 component should be installed.

Select hosts for a component

4. After component distribution is completed, click ***Save***.


Cluster Manager
Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 16:52:47 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
HTTP Mirror
HTTP Repo Server 1 / 1
Add hosts
bds-test.ru-central1.internal
Docker Registry
Docker Registry 1 / 1
Add hosts
bds-test.ru-central1.internal
HTTP Mirror
Docker Registry

Save mapping of components

Install a cluster

Daria Barysheva

Contents

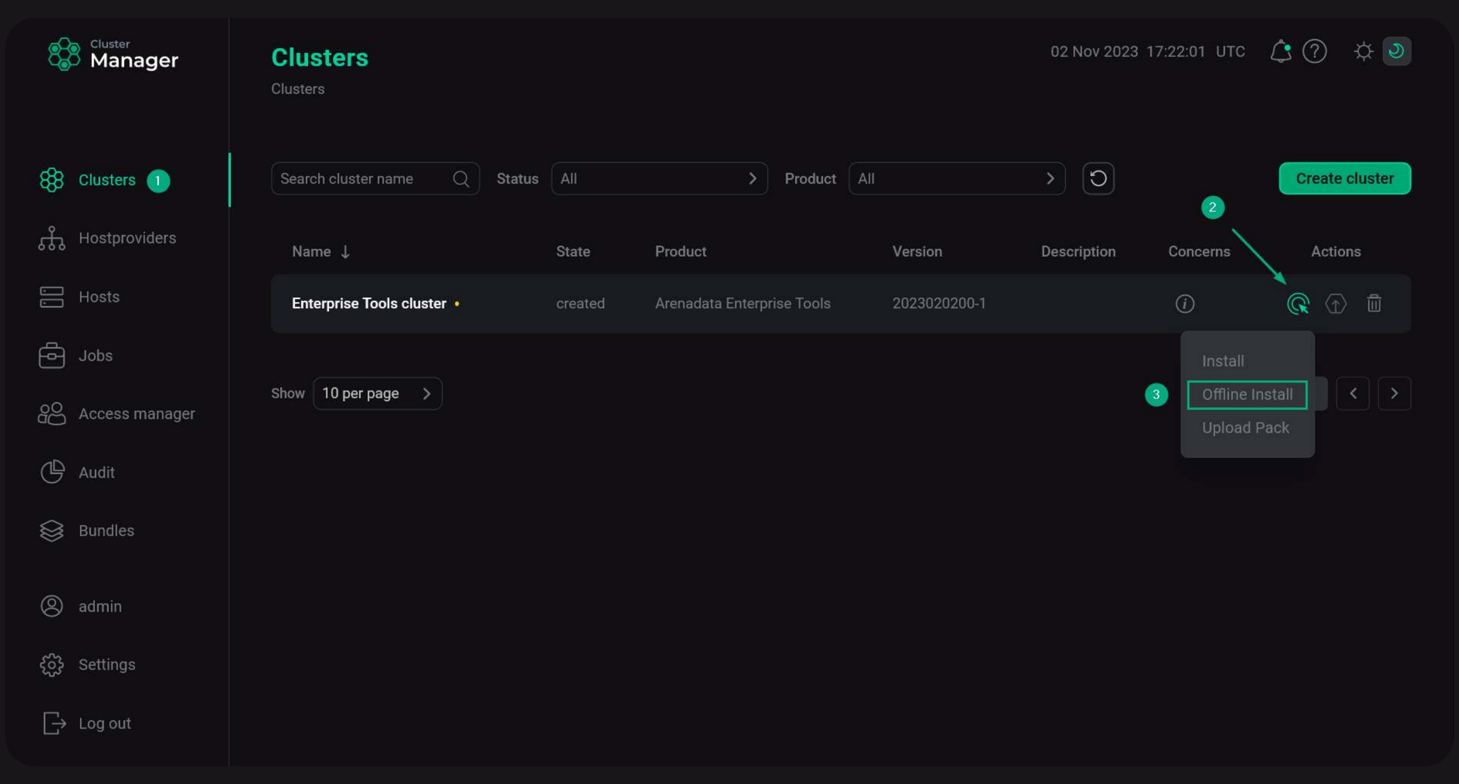
To Table of Contents

Step 1. Run installation
Step 2. Upload packages for offline installation
Step 3. Verify results

Step 1. Run installation

The last step of adding a new Enterprise Tools cluster is to install its services. To do this:

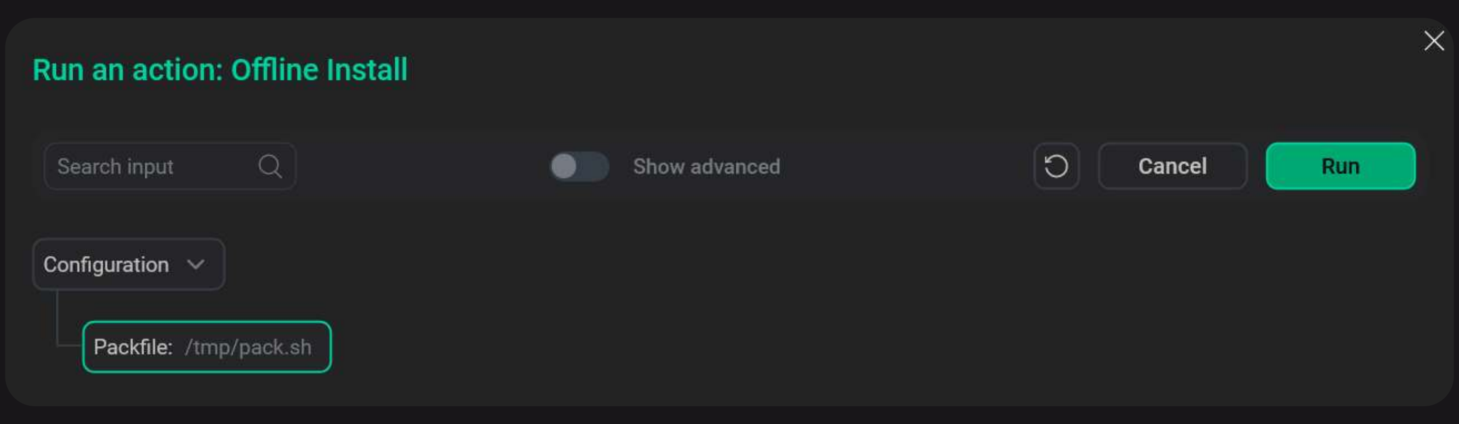
1. Apply the ***Offline Install*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.


Cluster
Manager
Clusters
Clusters
02 Nov 2023 17:22:01 UTC
Clusters 1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
2
3
Show
10 per page
Install
Offline Install
Upload Pack

Name ↓	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	created	Arenadata Enterprise Tools	2023020200-1			

Switch to cluster installation

2. In the opened window, enter a full path to the shell script that is received from the Arenadata support team and click ***Run***.


Run an action: Offline Install
Search input
Show advanced
Cancel
Run
Configuration

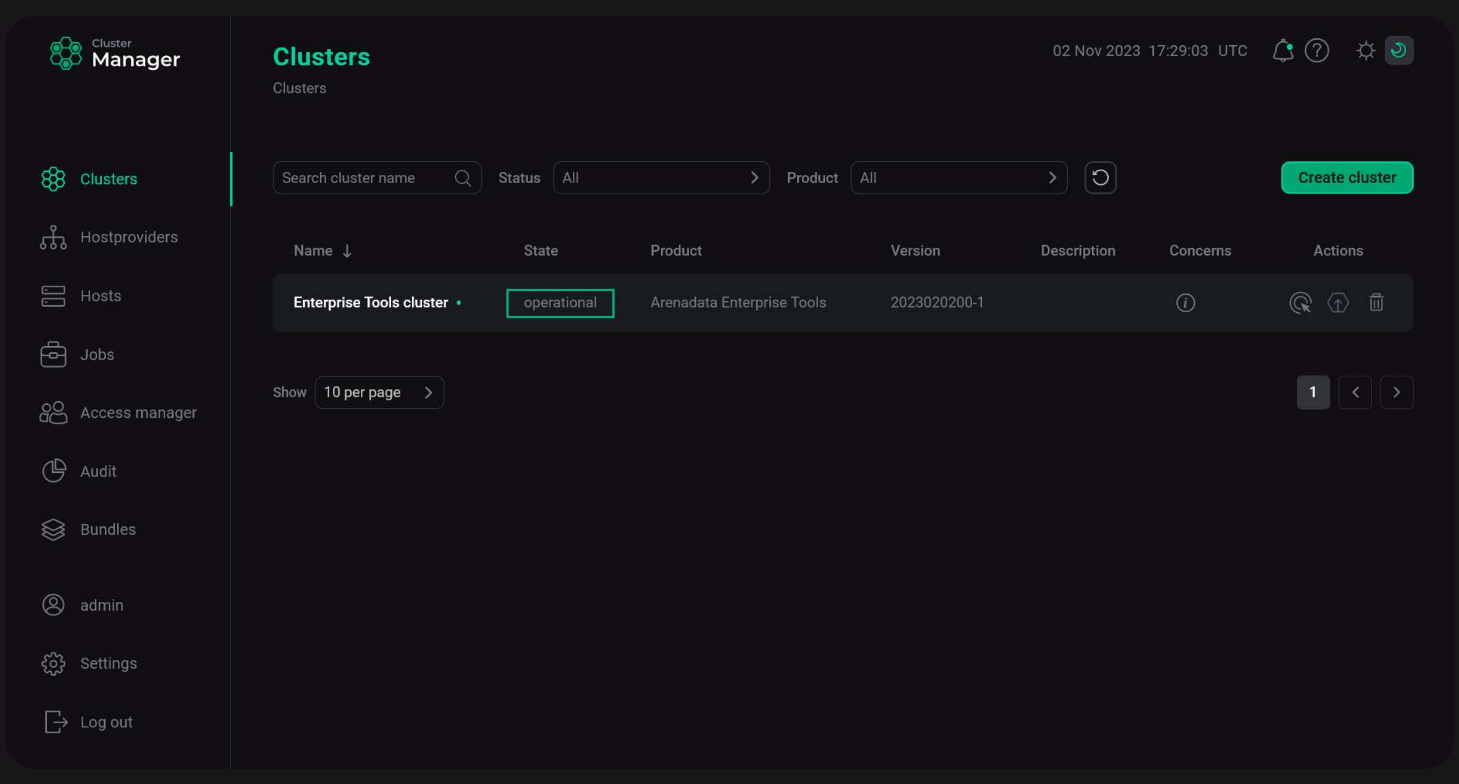
Packfile: /tmp/pack.sh

Verify installation

3. Wait until the installation process is completed.

NOTE

In case of errors, you can find logs on the ***Jobs*** page.


Cluster
Manager
Clusters
Clusters
02 Nov 2023 17:29:03 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1

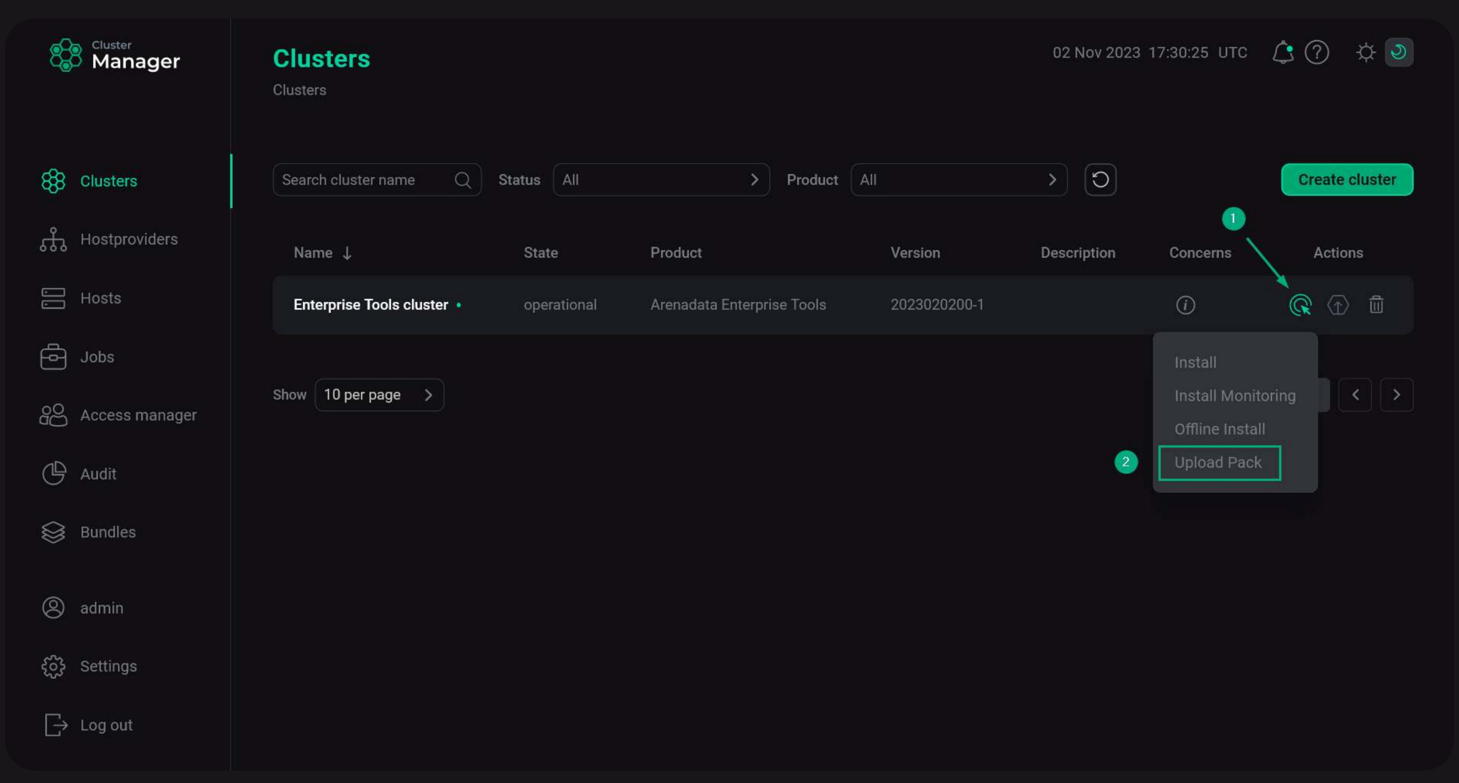
Name ↓	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	operational	Arenadata Enterprise Tools	2023020200-1			

Cluster installation is completed

Step 2. Upload packages for offline installation

In addition to installing the cluster, you should process the shell script that is received from the Arenadata support team. It is necessary to enable offline installation of products in the future. To do this:

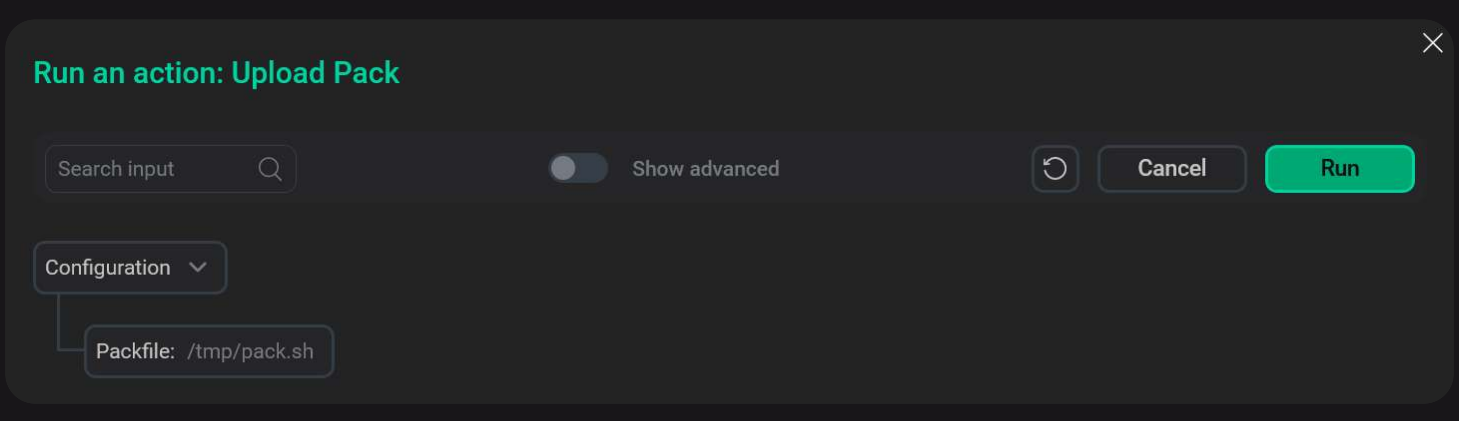
1. Apply the ***Upload Pack*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.


Cluster
Manager
Clusters
Clusters
02 Nov 2023 17:30:25 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
1
2
Show
10 per page
Install
Install Monitoring
Offline Install
Upload Pack

Name ↓	State	Product	Version	Description	Concerns	Actions
Enterprise Tools cluster	operational	Arenadata Enterprise Tools	2023020200-1			

Switch to uploading packages

2. In the opened window, enter a full path to the shell script received from the Arenadata support team and click ***Run***.


Run an action: Upload Pack
Search input
Show advanced
Cancel
Run
Configuration

Packfile: /tmp/pack.sh

Verify upload

3. Wait until the upload process is completed.

NOTE

In case of errors, you can find logs on the ***Jobs*** page.

Step 3. Verify results

The steps for checking the installation results are listed below:

1. Check that ports `443` and `81` are listened at the host where cluster is installed. The first port is used by the Docker Registry service, the second one — by the HTTP Mirror service.

```
$ sudo netstat -ntlp|grep L|grep -E "81|443"
```

   The result should look like this:

```
tcp6       0      0 :::443                  :::*                    LISTEN      1694/docker-proxy-c
tcp6       0      0 :::81                   :::*                    LISTEN      1708/docker-proxy-c
```

2. Check that the */opt/arenadata/etools/httprepo/arenadata-repo* folder exists at the host where the cluster is installed. This folder should contain the data necessary for offline installation of the products that you have chosen before getting a shell script from the Arenadata support team.

```
$ ls -la /opt/arenadata/etools/httprepo/arenadata-repo
```

   The following example displays the data that can be placed in this folder for ADH installation.

```
total 0
drwxr-xr-x 5 root root 45 Sep 16 13:35 .
drwxr-xr-x 3 root root 28 Sep 16 13:35 ..
drwxr-xr-x 4 root root 38 Sep 16 13:35 ADH
drwxr-xr-x 4 root root 28 Sep 16 13:35 ADM
drwxr-xr-x 3 root root 20 Sep 16 13:35 zookeeper
```

Create a cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

Step 1. Get the Enterprise version of a cluster bundle

Step 2. Upload a cluster bundle to ADCM

Step 3. Create a cluster based on the uploaded bundle

In ADCM a **cluster** means a set of services running on a set of hosts.

Clusters distributions for ADCM are made in the form of so-called bundles. Generally a bundle is a regular archive that includes a description and software logic of cluster deployment and exploitation.

The steps for adding a cluster to ADCM are listed below.

Step 1. Get the Enterprise version of a cluster bundle

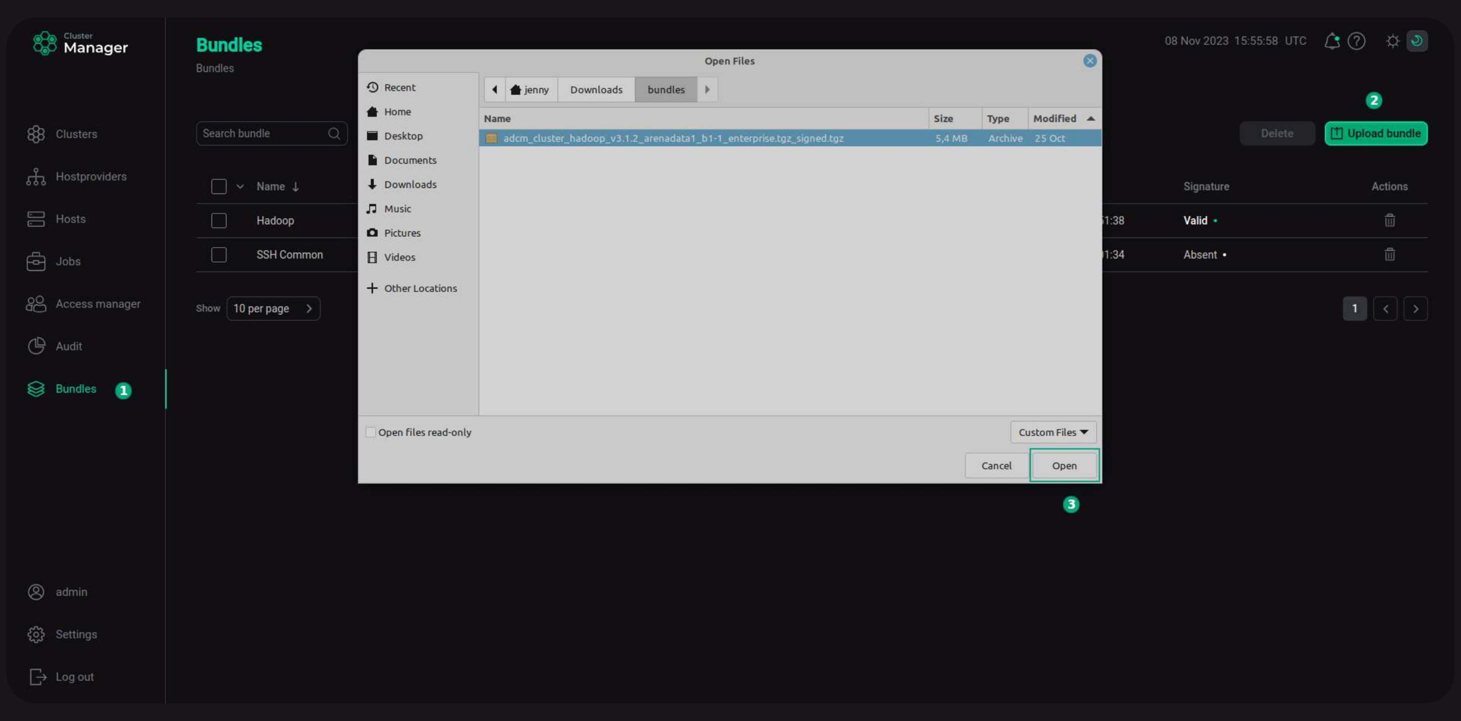
Please get a cluster bundle from the Arenadata support team.

Step 2. Upload a cluster bundle to ADCM

To upload a bundle to ADCM, follow the steps:

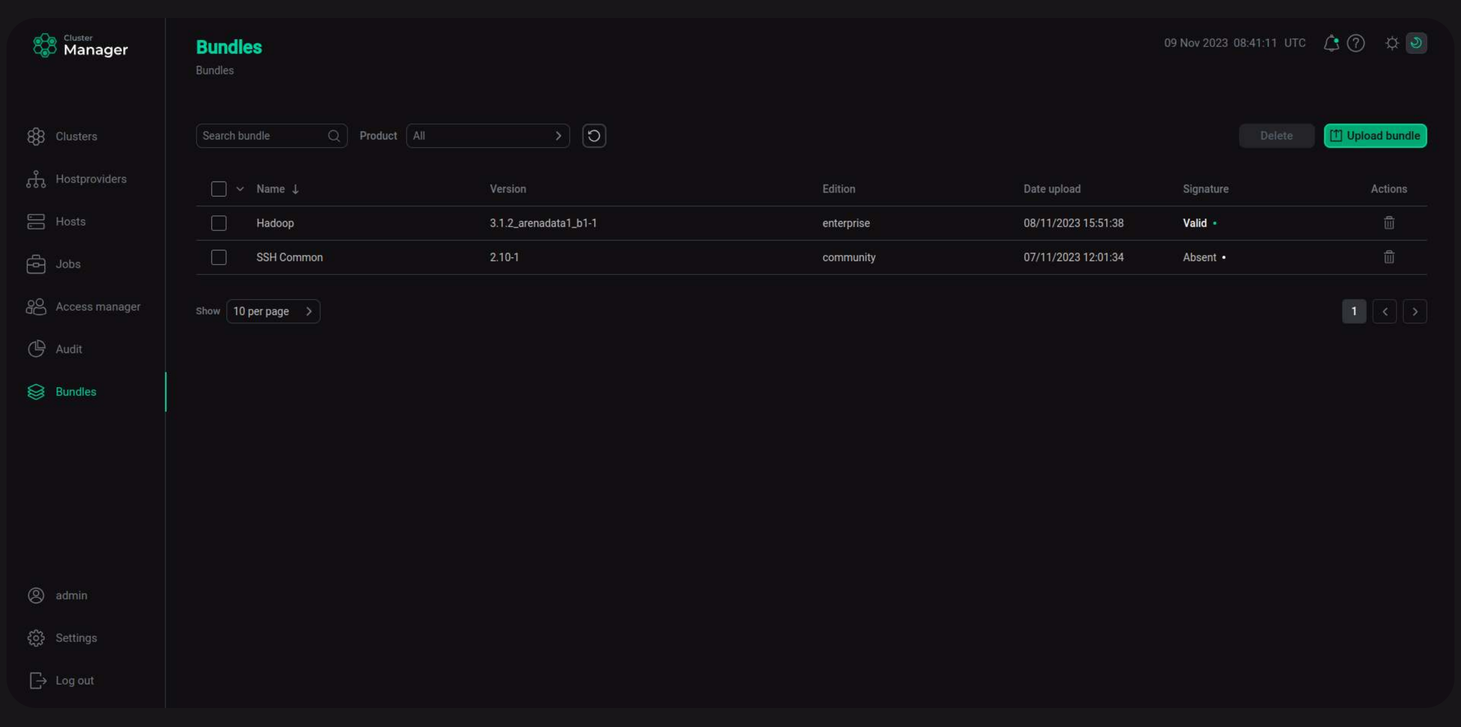
1. Select the ***Bundles*** item in the left navigation menu and click ***Upload bundle***.

2. Select a bundle in the Open File Dialog.



Upload a bundle

3. As a result of the performed actions, a bundle is displayed on the ***Bundles*** page.



The result of successful uploading a bundle

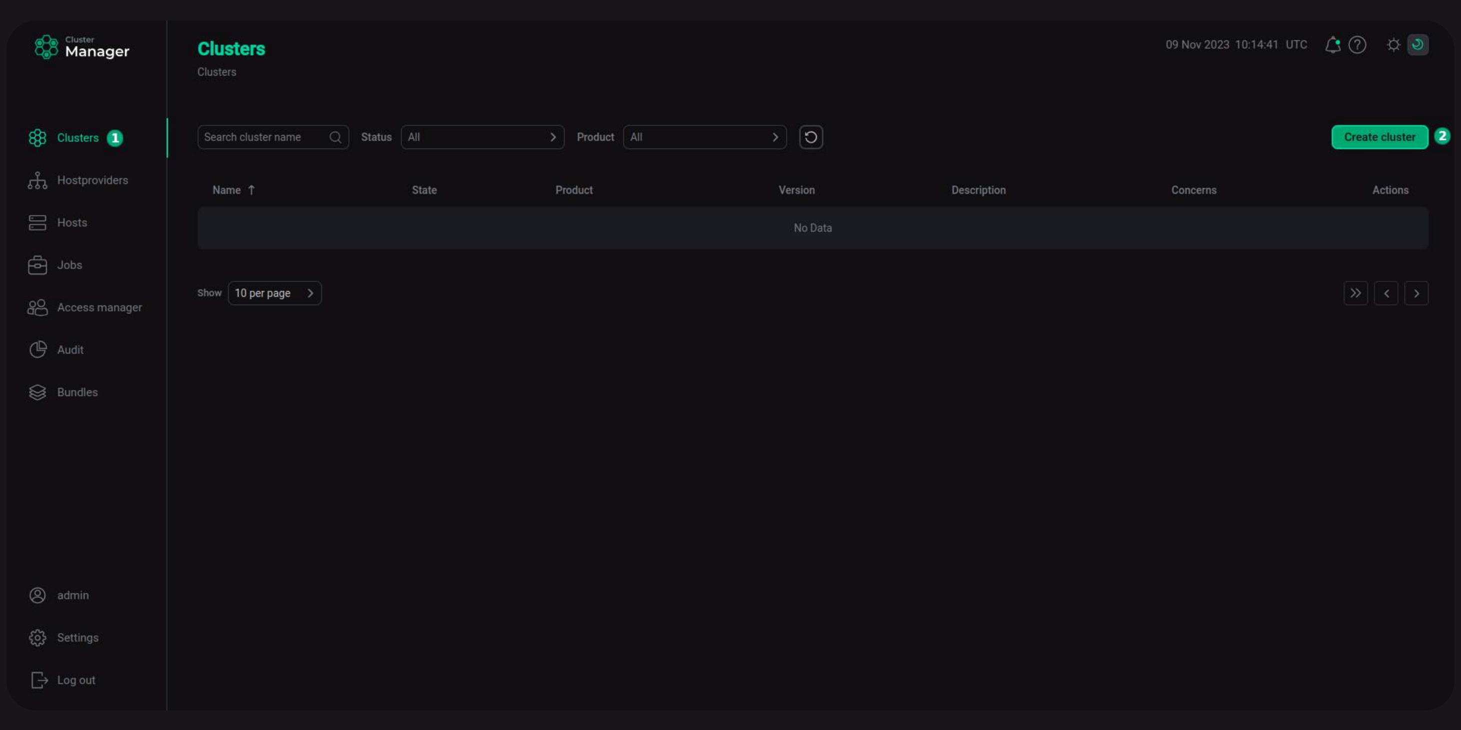
Step 3. Create a cluster based on the uploaded bundle

IMPORTANT

When creating a cluster via the ADCM web interface, you generate a new ADCM cluster instance. It means only adding the cluster to ADCM — at this stage no services are deployed.

The steps for cluster creation are given below:

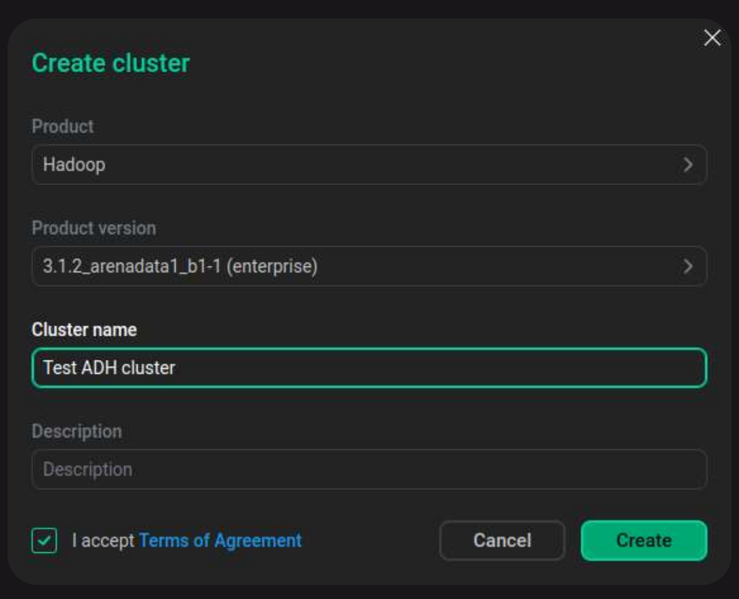
1. Select the ***Clusters*** item in the left navigation menu and click ***Create cluster***.



Switch to cluster creation

2. In the opened dialog, describe a new cluster instance:

   - Select an uploaded cluster bundle in the ***Product*** field.
   - Select a bundle version in the ***Product version*** field. Several versions become available in the case of different versions of the same bundle being uploaded.
   - Enter a cluster name in the ***Cluster name*** field.
   - Enter a cluster description in the ***Description*** field if necessary.
   - Set the ***I accept Terms of Agreement*** flag to sign the license agreement. This flag is displayed only for Enterprise bundles. To read the agreement text, refer to the ***Terms of Agreement*** link. You can also sign the license agreement on the ***Bundles*** page by clicking the desired product bundle.
   - Click ***Create***.



Create cluster

Product: Hadoop

Product version: 3.1.2_arenadata1_b1-1 (enterprise)

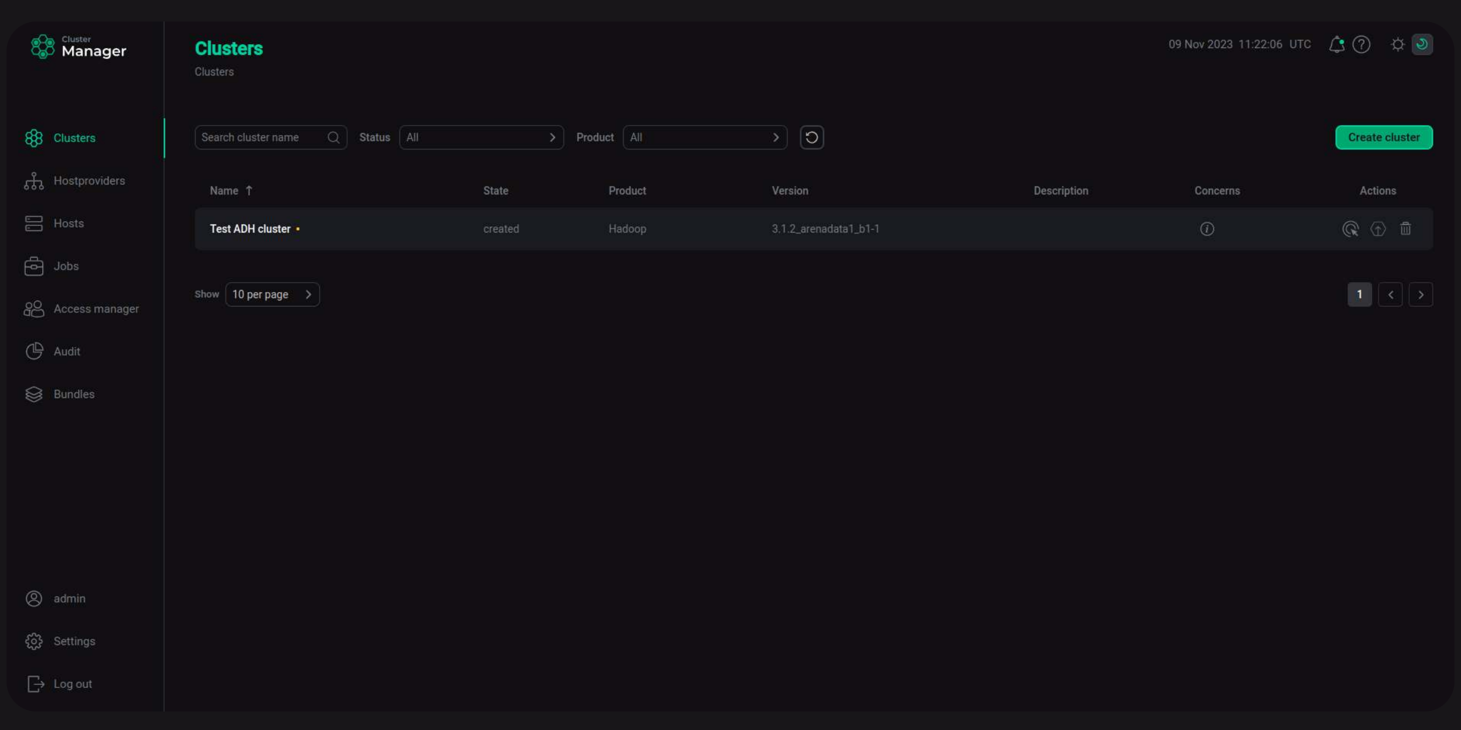
Cluster name: Test ADH cluster

Description: Description

☑ I accept Terms of Agreement

Create a cluster

3. As a result, the created cluster instance is displayed on the ***Clusters*** page.



The result of successful cluster creation

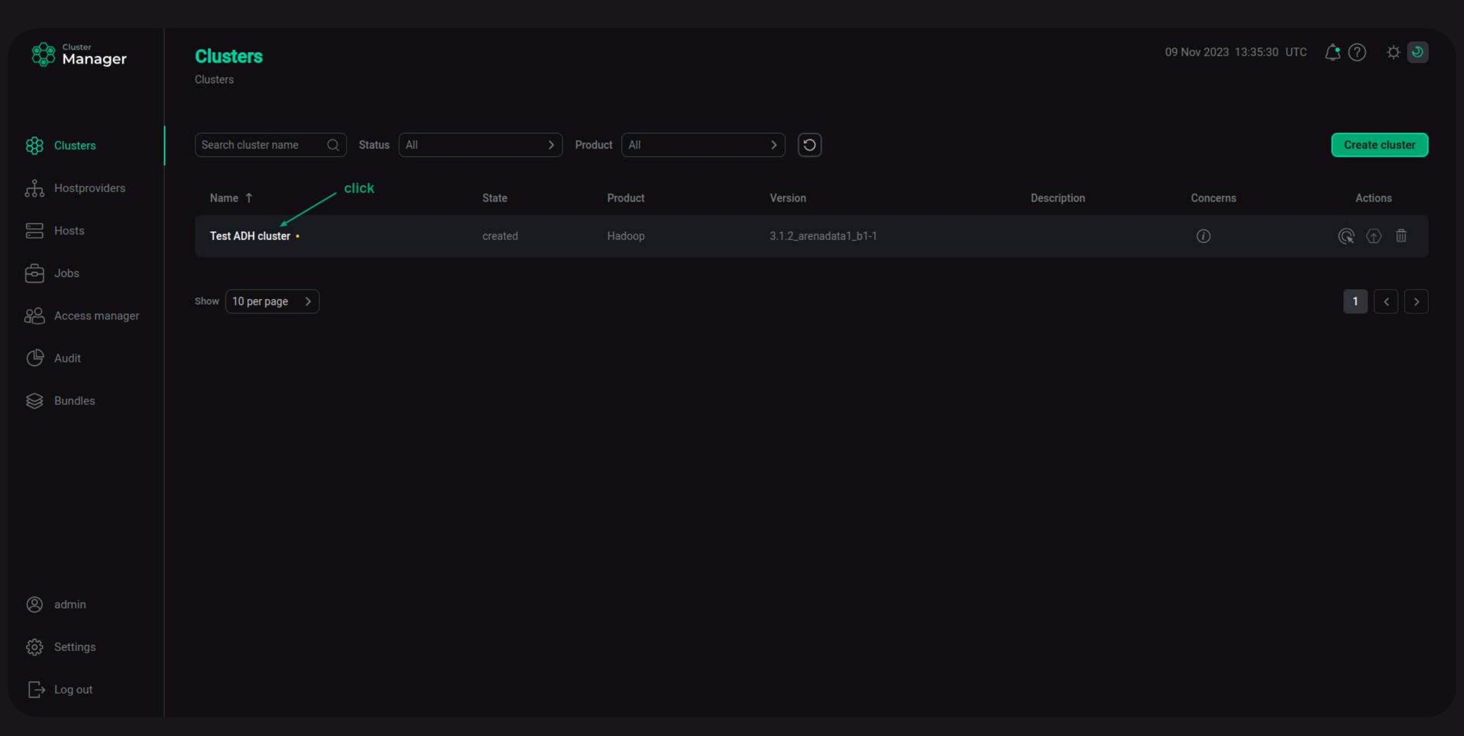
Add services

Eugenia Kuzina, Daria Barysheva

To Table of Contents

In ADCM a **service** means a software that performs some function. Examples of services for ADH clusters: HDFS, HBase, Hive, etc. The steps for adding services to a cluster are listed below:

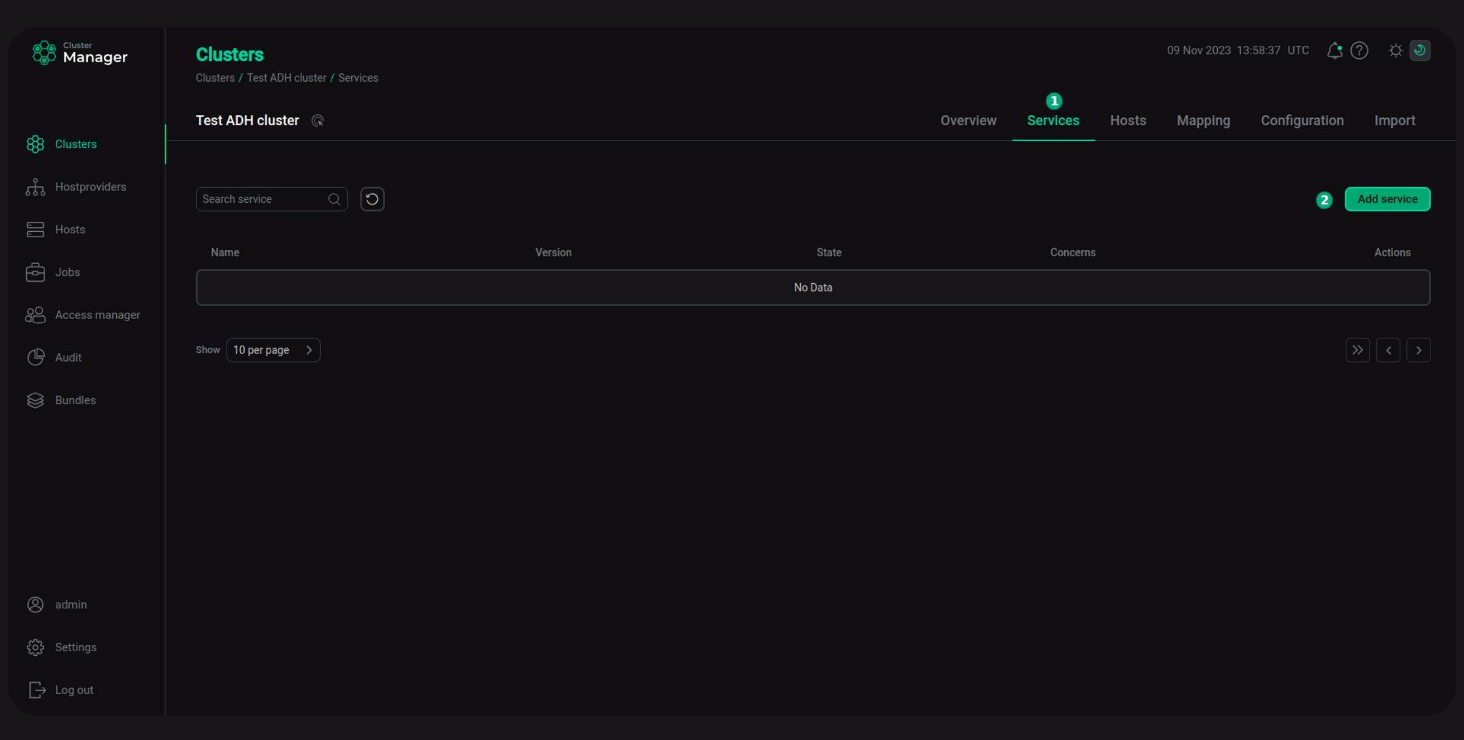
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

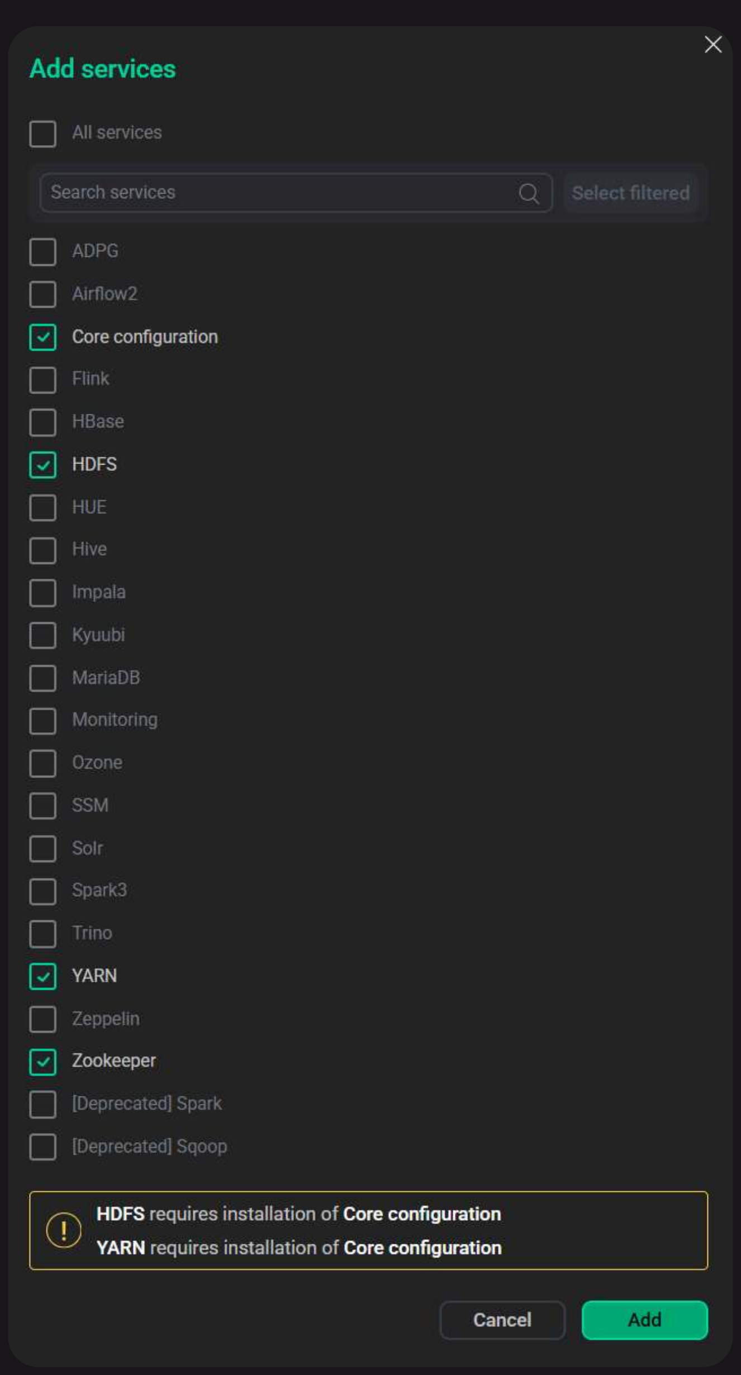
Select a cluster

2. Open the ***Services*** tab on the cluster page and click ***Add service***.


Cluster Manager
Clusters
Clusters / Test ADH cluster / Services
09 Nov 2023 13:58:37 UTC
Test ADH cluster
1
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
2
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page

Switch to adding services

3. In the opened dialog, select services that should be added to the cluster and click ***Add***.


Add services
Search services
Select filtered
Cancel
Add

☐ All services
☐ ADPG
☐ Airflow2
☑ Core configuration
☐ Flink
☐ HBase
☑ HDFS
☐ HUE
☐ Hive
☐ Impala
☐ Kyuubi
☐ MariaDB
☐ Monitoring
☐ Ozone
☐ SSM
☐ Solr
☐ Spark3
☐ Trino
☑ YARN
☐ Zeppelin
☑ Zookeeper
☐ [Deprecated] Spark
☐ [Deprecated] Sqoop

HDFS requires installation of **Core configuration**
YARN requires installation of **Core configuration**

Select services

The brief description of available services is listed below.

Services that can be added to the ADH cluster

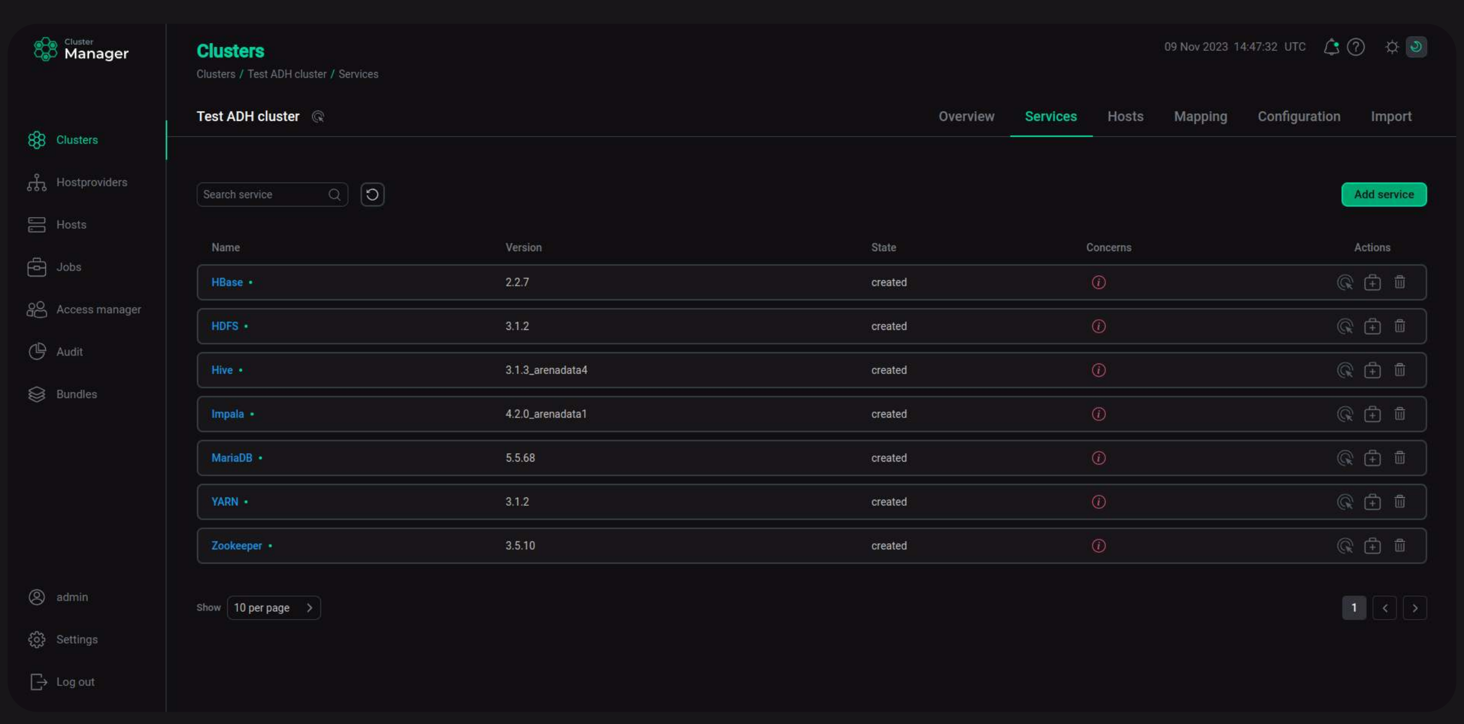
Service	Purpose
Airflow2	In comparison with the Airflow 1.x version, it offers the following features: High Availability, lowered task latency, full REST API, TaskFlow API, task groups, independent providers, and others
Core configuration	A service that contains common Hadoop packages required by every other service except ADPG, Solr, Zookeeper, and Monitoring. This service is installed on all nodes. If an already installed Hadoop service is expanded to another host, the Core configuration service is automatically installed on the host. The service is supported since ADH version 3.3.6.2.b1. Configuration parameters of this service are described in the Configuration parameters article. After applying any changes to this service configuration, you need to run the ***Update Core configuration*** action for this service
Flink	A distributed platform used in high-load Big Data applications for analyzing data stored in Hyperwave clusters. It can be used in different streaming use cases: event-driven applications, stream and batch analytics, data pipelines and ETL, etc.
Flink2	Flink-based service with version 2.x. This service is in the technical preview mode, may have a number of limitations, and is not recommended for production use
HBase	A non-relational, distributed database written in Java and used on the top of HDFS. Belongs to the class of column-oriented key-value storages. It is useful for random, real-time read/write access to Big Data
HDFS	A distributed file system used in Hyperwave clusters for storing large files. Provides the possibility of the streaming access to the information distributed block-by-block across cluster nodes
Hive	A software designed for building data warehouses (DWH) and analyzing Big Data. It runs on the top of HDFS and other compatible systems, such as Apache HBase. It facilitates writing, reading, and managing large datasets stored in distributed systems
HUE	Apache HUE is a UI-based SQL assistant that provides interpreters for various data sources, like Impala, Hive, Kyuubi, and many more. It has a handy query editor with intelligent autocomplete, drag and drop, quick browse, and other features
Impala	Impala provides fast, interactive SQL queries on data stored in HDFS, HBase, or S3. In addition to the unified storage platform, Impala also uses the same metadata, SQL syntax (Hive SQL), and JDBC driver as Apache Hive. It makes Impala a unified platform for real-time or batch-oriented queries
Kyuubi	Apache Kyuubi is a distributed and multi-tenant gateway to provide SQL on Data Warehouses and Lakehouses. It builds distributed SQL query engines on top of various kinds of modern computing frameworks, e.g. Apache Spark, Flink, Hive, Impala, etc., to query massive datasets distributed over fleets of machines from heterogeneous data sources
Monitoring	A service that should be added if monitoring of the ADH cluster is planned
Ozone	Apache Ozone (O3) is a distributed object storage that is optimized for working with both Hadoop services and S3 storages. It’s highly scalable, can share the same cluster and security policies with HDFS, and is capable of effectively handling files no matter the size
Solr	A search platform based on the Apache Lucene project. Its main features include full-text search, faceted search, highlighting search results, distributed indexing, integration with databases, processing documents with a complex format (Word, PDF, etc.), load-balanced querying, centralized configuration, and others
Spark3	A fast analytics engine used for large-scale data processing and is compatible with Hadoop data. It can run in Hyperwave clusters using YARN or Spark’s standalone mode. It can process data in HDFS, HBase, Cassandra, Hive, and other Hadoop input formats. It supports both batch processing and new workloads like streaming, machine learning, interactive queries, etc. Spark 3.x offers such new features as adaptive execution of Spark SQL, dynamic partition pruning (DPP), graph processing, enhanced support for deep learning, and others
Spark4	Spark-based service with version 4.x. This service is in the technical preview mode, may have a number of limitations, and is not recommended for production use
SSM	Smart Storage Manager is a service that aims to optimize the efficiency of storing and managing data in the Hadoop Distributed File System. SSM collects HDFS operation data and system state information, and based on the collected metrics can automatically use methodologies such as cache, storage policies, heterogeneous storage management (HSM), data compression, and Erasure Coding. In addition, SSM provides the ability to configure asynchronous replication of data and namespaces to a backup cluster for the purpose of organizing DR
Trino	Trino is an open-source SQL query engine used for processing data in parallel, distributed over multiple storages, such as object storages, databases, and file systems
YARN	A service needed for managing cluster resources and scheduling/monitoring jobs. Uses a special daemon (ResourceManager) that abstracts all the computing resources of the cluster and manages their provision to distributed applications
Zeppelin	A service that plays a role of a web-based notebook and enables interactive data analytics. Allows to create queries to data in Hyperwave clusters and display the results in the form of tables, graphs, charts, etc.
Zookeeper	A centralized coordination service for distributed applications. It is used in Hyperwave clusters for failure detection, active NameNode election, health monitoring, session management, etc.

The minimal set of services recommended for ADH clusters is described below:

- Core configuration;
- HDFS/Ozone;
- YARN;
- Zookeeper (optional for the Community Edition of ADH).

These services make up the core of Hyperwave and are sufficient to organize distributed data storage and processing. The full list of services depends on the requirements of a particular project.

4. As a result, the added services are displayed at the *Services* tab.


Cluster Manager
Clusters
Clusters / Test ADH cluster / Services
09 Nov 2023 14:47:32 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
HBase	2.2.7	created		
HDFS	3.1.2	created		
Hive	3.1.3_arenadata4	created		
Impala	4.2.0_arenadata1	created		
MariaDB	5.5.68	created		
YARN	3.1.2	created		
Zookeeper	3.5.10	created		

The result of successful adding services to a cluster

NOTE

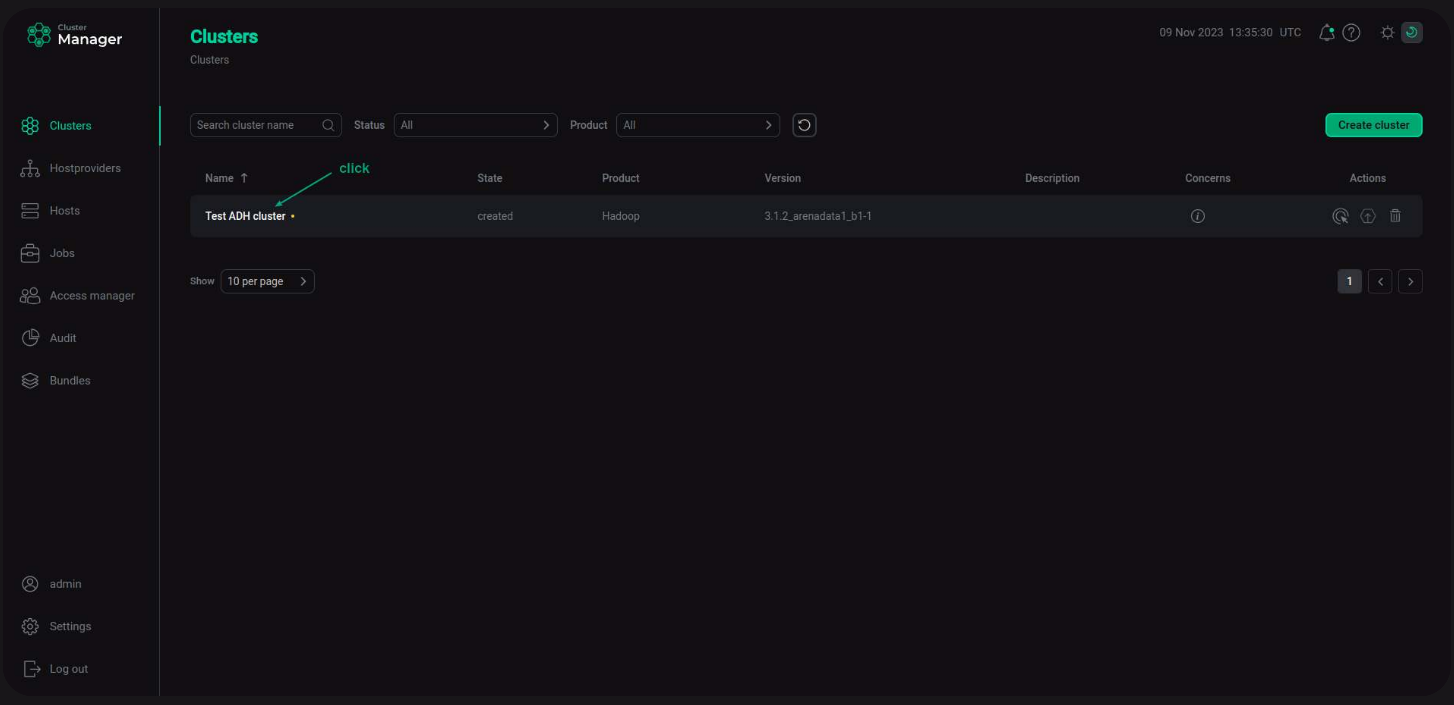
You can also add services later. The process of adding new services to already running cluster does not differ from installing a service from scratch.

Add hosts to a cluster

Eugenia Kuzina, Daria Barysheva

The steps for adding hosts to a cluster are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.

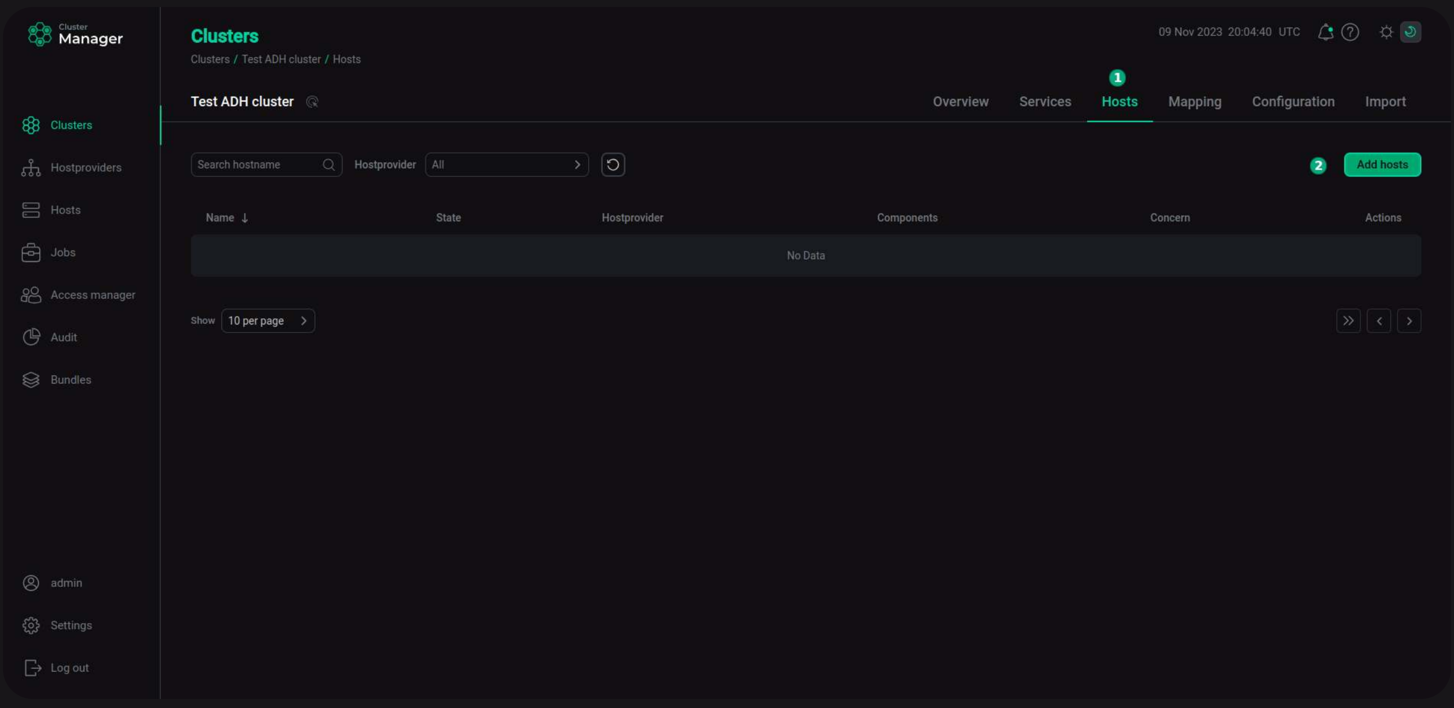


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Search cluster name
Status
Product
Create cluster
click
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

Select a cluster

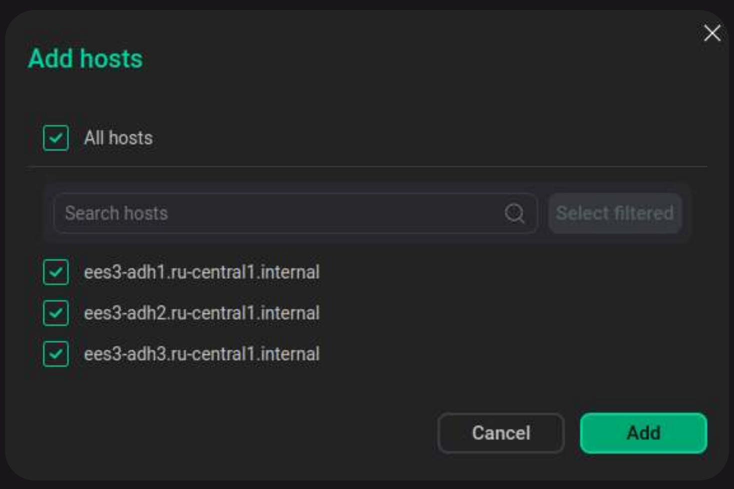
2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.



Cluster Manager
Clusters
Clusters / Test ADH cluster / Hosts
Test ADH cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
2
Add hosts
Name
State
Hostprovider
Components
Concern
Actions
No Data
Show 10 per page

Switch to adding hosts

3. In the opened dialog, select hosts that should be added to the cluster and click ***Add***.



Add hosts

☑ All hosts

Search hosts — Select filtered

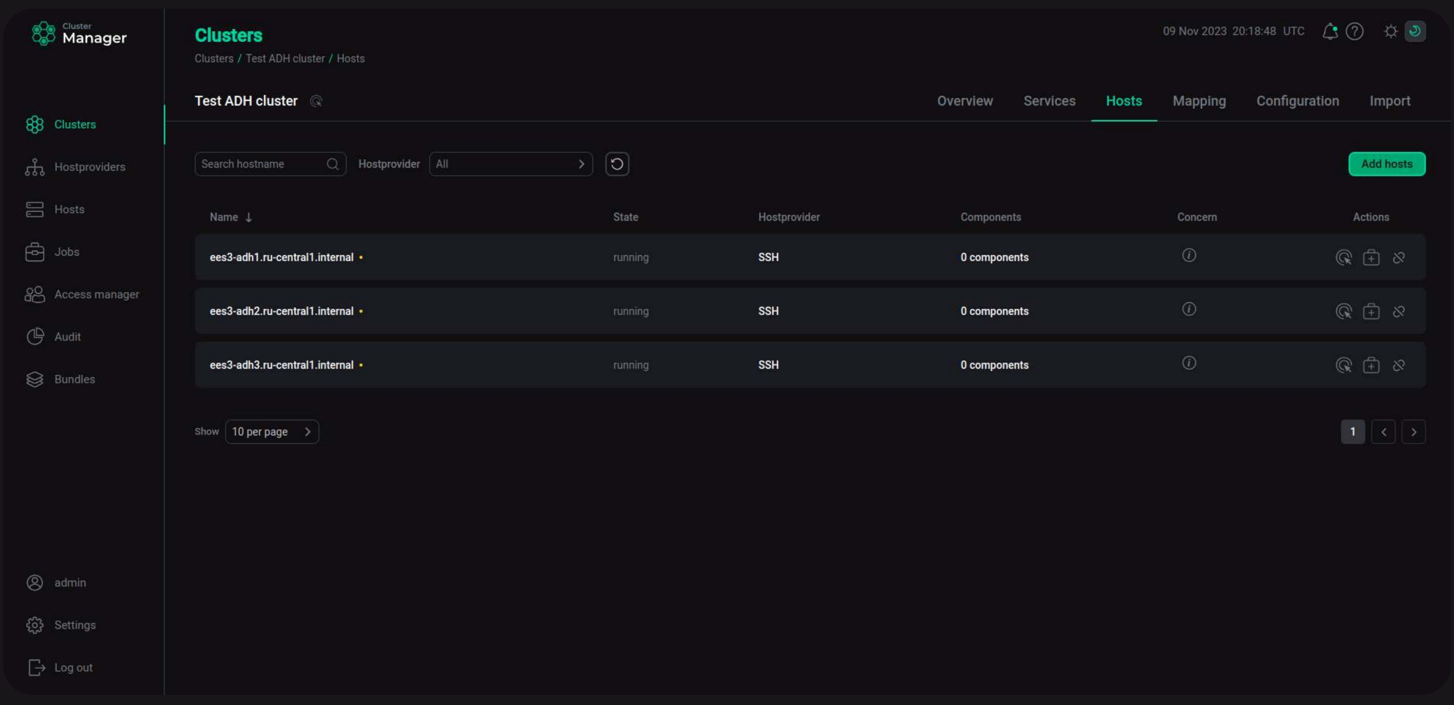
☑ ees3-adh1.ru-central1.internal

☑ ees3-adh2.ru-central1.internal

☑ ees3-adh3.ru-central1.internal

Select hosts

4. As a result, the added hosts are displayed on the ***Hosts*** tab.



Cluster Manager
Clusters
Clusters / Test ADH cluster / Hosts
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Add hosts
Show 10 per page

Name	State	Hostprovider	Components	Concern	Actions
ees3-adh1.ru-central1.internal	running	SSH	0 components		
ees3-adh2.ru-central1.internal	running	SSH	0 components		
ees3-adh3.ru-central1.internal	running	SSH	0 components		

The result of successful adding hosts to a cluster

Add components

Eugenia Kuzina, Daria Barysheva

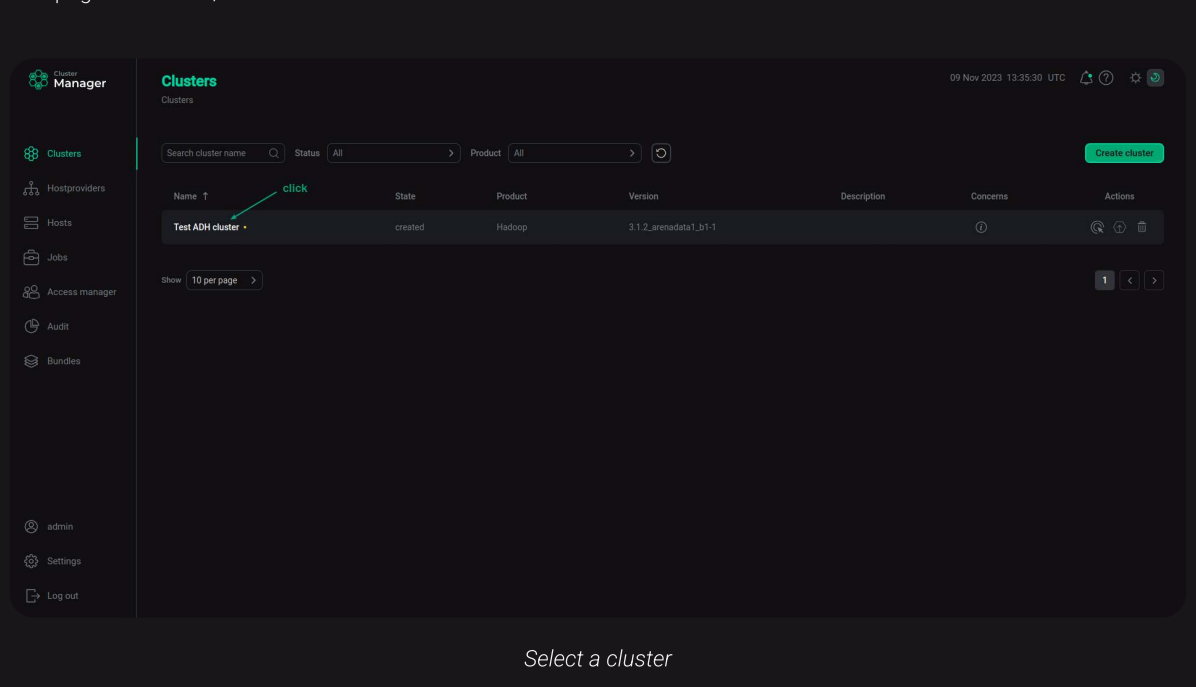
To Table of Contents

In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts. For example, HDFS service can consist of such components as DataNode, NameNode, Client, and others.

> **IMPORTANT**
>
> Initially, there are no components on any of cluster hosts. You should map components to hosts manually.

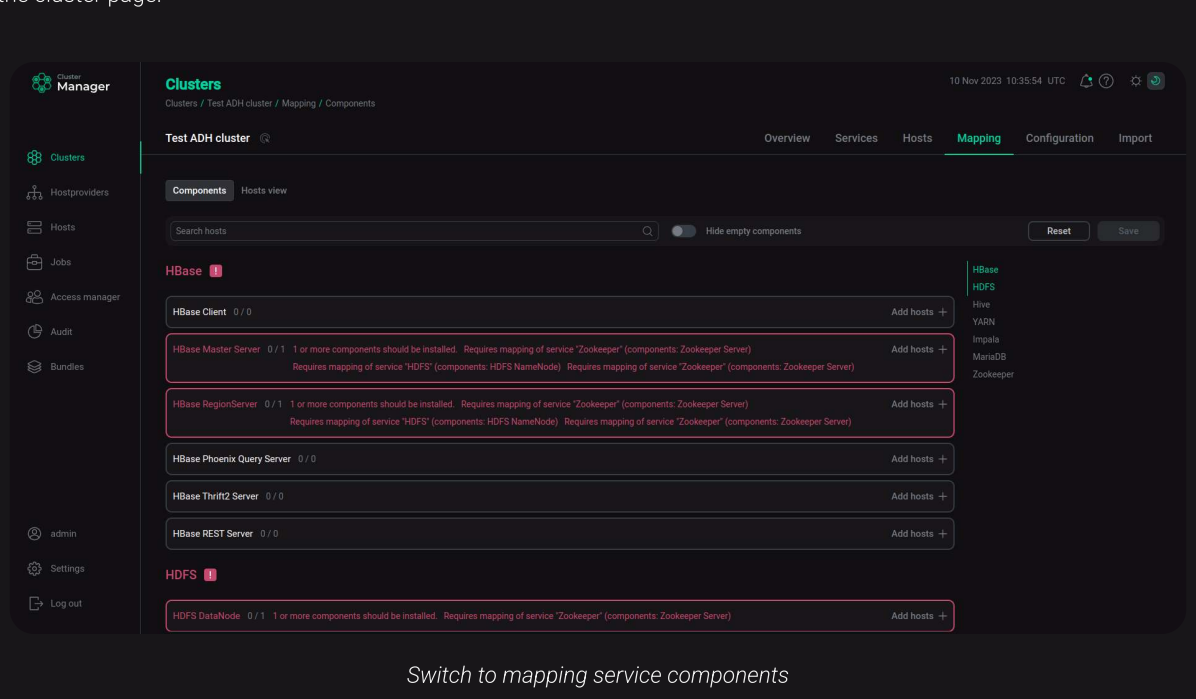
The steps for adding components to hosts are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.



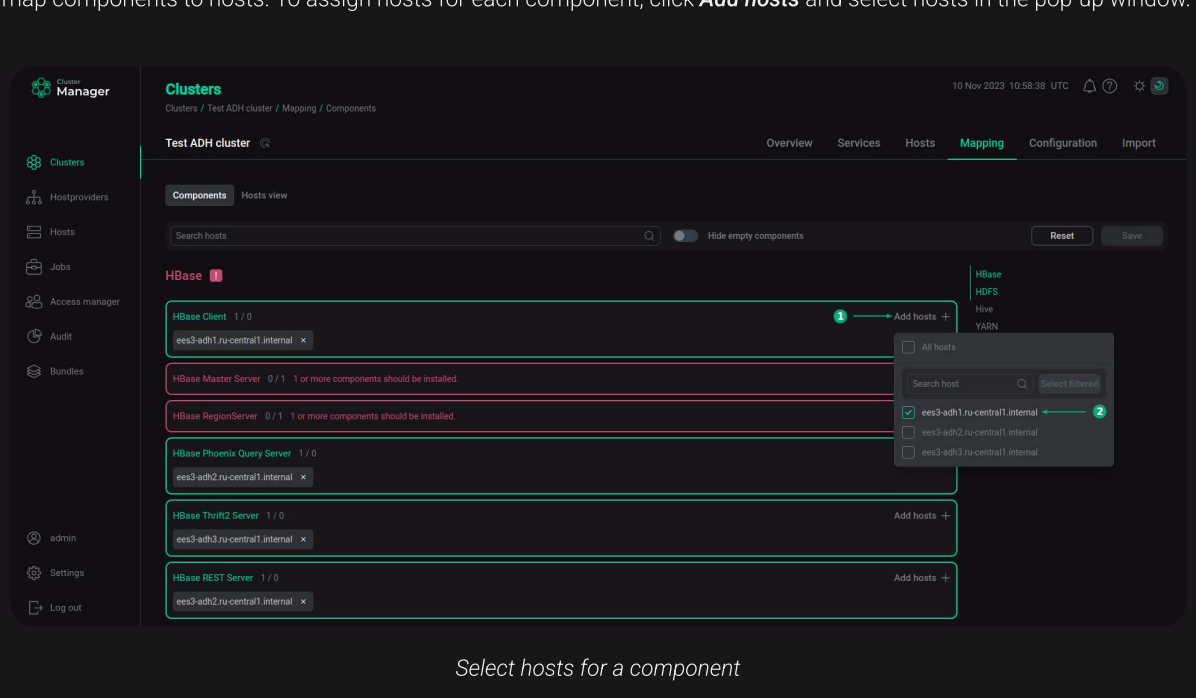
Select a cluster

2. Open the ***Mapping*** tab on the cluster page.



Switch to mapping service components

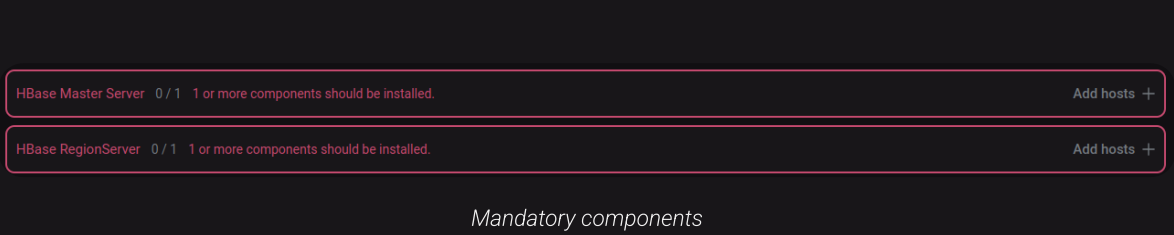
3. In the window that opens, map components to hosts. To assign hosts for each component, click ***Add hosts*** and select hosts in the pop-up window.



Select hosts for a component

Mandatory components

Each service can have mandatory and optional components. Mandatory components are highlighted in red. The ADCM UI also displays how many mandatory components should be installed.



HBase Master Server 0 / 1 1 or more components should be installed. Add hosts

HBase RegionServer 0 / 1 1 or more components should be installed. Add hosts

Mandatory components

> **CAUTION**
>
> Without assigning a necessary number of hosts to mandatory components, the component mapping cannot be saved.

The examples of distribution components in ADH clusters consisting of 3, 6, and 9 hosts are described in the table below. Examples are based on the Enterprise Edition of ADH.

Examples of components distribution

3 hosts

Host	Service	Component
Host 1	Core configuration	Configuration server
	Flink	Flink Client Flink JobManager Flink TaskManager Flink SQL-Gateway
	HBase	HBase Client HBase RegionServer
	HDFS	HDFS Client HDFS DataNode HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Client Hive Metastore Hive Tez
	Impala	Impala Daemon Impala Statestore Impala Catalog Service
	Monitoring	Grafana Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Recon Ozone HttpFS Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client Spark3 History server
	Trino	Trino Coordinator Trino Client
	YARN	MapReduce History server YARN Client YARN NodeManager
	ZooKeeper	ZooKeeper Server
Host 2	Airflow2	Airflow2 Broker Airflow2 Flower Airflow2 Scheduler Airflow2 Server Airflow2 Worker
	Flink	Flink Client Flink History Server
	HBase	HBase Client HBase Master server HBase Phoenix query server HBase RegionServer
	HDFS	HDFS Client HDFS DataNode HDFS HttpFS server HDFS JournalNode
	Hive	Hive Client Hive HiveServer2 Hive Tez Hive TezUI
	Impala	Impala Daemon
	Monitoring	Prometheus Server Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Datanode
	Solr	Solr Server Solr prometheus exporter
	Spark3	Spark3 Client Spark3 Livy server
	Trino	Trino Worker Trino Client
	YARN	YARN Client YARN NodeManager YARN ResourceManager
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 3	Arenadata PostgreSQL	Arenadata PostgreSQL
	Flink	Flink Client
	HBase	HBase Client HBase RegionServer HBase Thrift2 server HBase REST Server
	HDFS	HDFS Client HDFS DataNode HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Client Hive Tez
	HUE	HUE Server
	Impala	Impala Daemon Impala Client
	Kyuubi	Kyuubi Server
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone S3G Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client
	SSM	SSM Server SSM Agent
	Trino	Trino Worker Trino Client
	YARN	YARN Client YARN NodeManager YARN Timeline server
	ZooKeeper	ZooKeeper Server

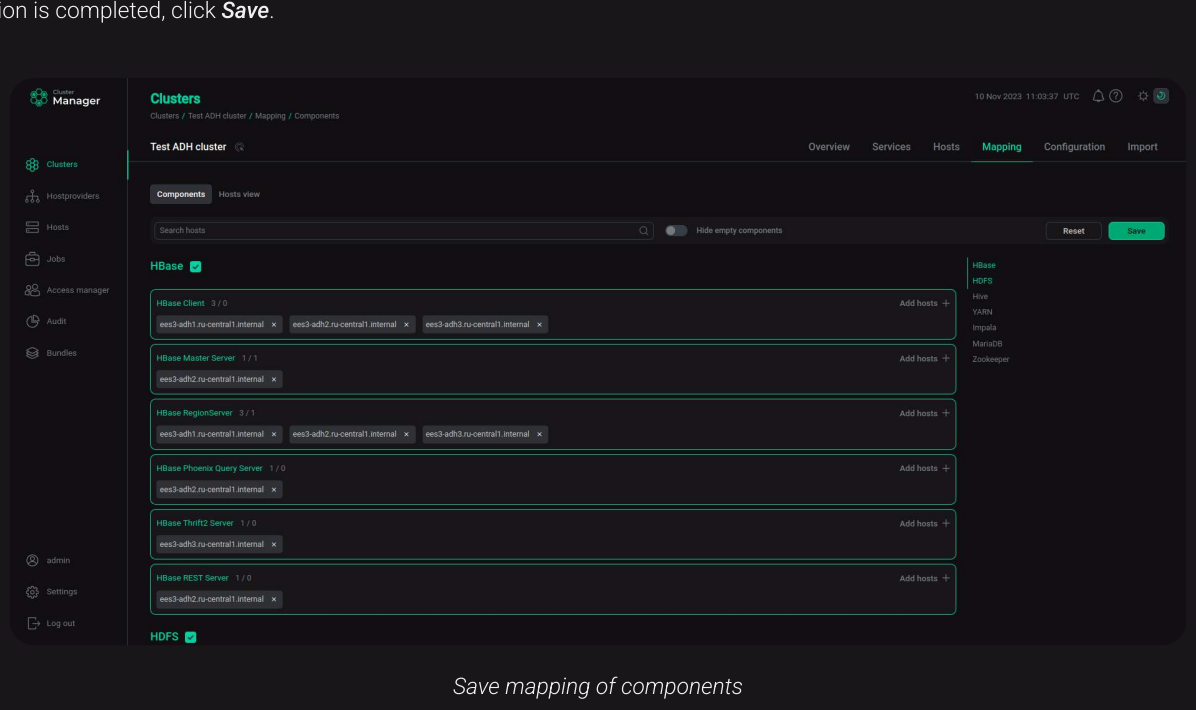
6 hosts

Host	Service	Component
Host 1	Core configuration	Configuration server
	Flink	Flink JobManager Flink TaskManager Flink SQL-Gateway
	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Metastore
	Impala	Impala Statestore Impala Catalog Service
	Monitoring	Grafana Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Recon Ozone HttpFS
	Solr	Solr Server
	Spark3	Spark3 History server
	Trino	Trino Coordinator Trino Client
	YARN	MapReduce History server
	ZooKeeper	ZooKeeper Server
Host 2	Airflow2	Airflow2 Broker Airflow2 Flower Airflow2 Scheduler Airflow2 Server Airflow2 Worker
	HBase	HBase Master server HBase Phoenix query server
	HDFS	HDFS JournalNode HDFS HttpFS server
	Hive	Hive HiveServer2 Hive TezUI
	Monitoring	Prometheus Server Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server Solr prometheus exporter
	Spark3	Spark3 Livy server
	Trino	Trino Worker Trino Client
	YARN	YARN ResourceManager
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 3	Arenadata PostgreSQL	Arenadata PostgreSQL
	HBase	HBase Thrift2 server HBase REST Server
	HDFS	HDFS NameNode HDFS ZKFC HDFS JournalNode
	HUE	HUE Server
	Kyuubi	Kyuubi Server
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	SSM	SSM Server SSM Agent
	YARN	YARN Timeline server
	ZooKeeper	ZooKeeper Server
Host 4	Flink	Flink Client Flink History Server
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon Impala Client
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 5	Flink	Flink Client
	HBase	HBase Client HBase RegionServer
	HDFS	HDFS DataNode HDFS Client
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 6	Flink	Flink Client
	HBase	HBase Client HBase RegionServer
	HDFS	HDFS DataNode HDFS Client
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone S3G
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server

9 hosts

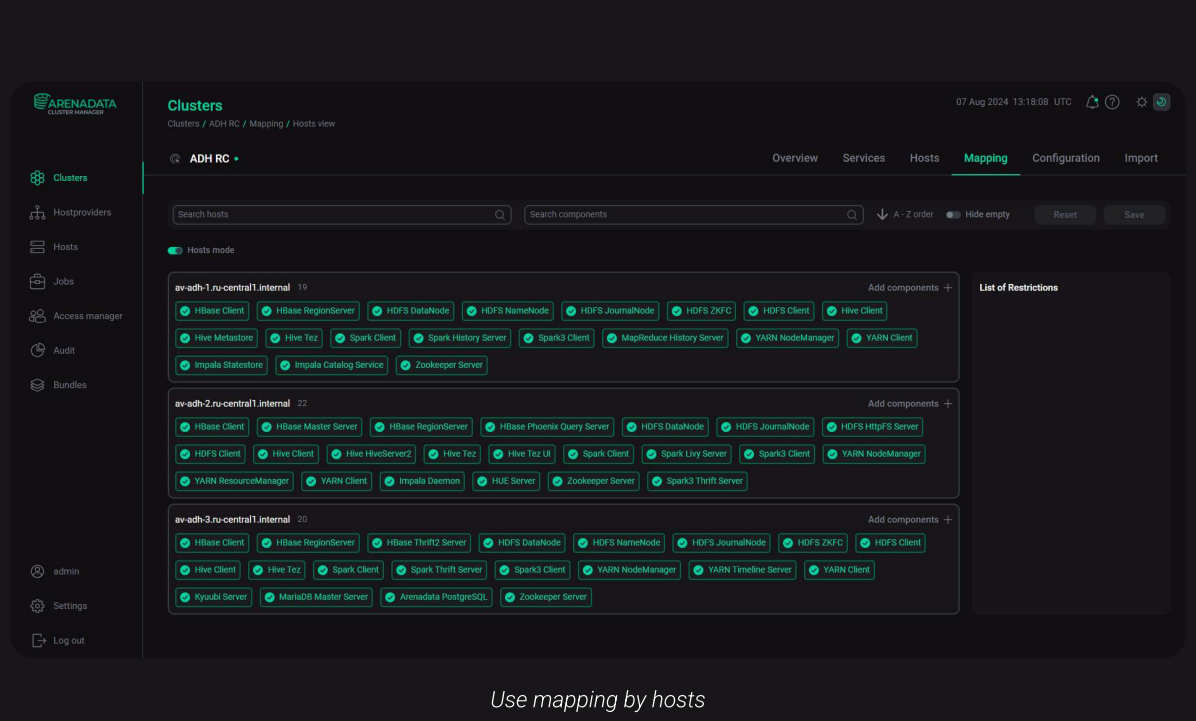
Host	Service	Component
Host 1	Core configuration	Configuration server
	Flink	Flink JobManager Flink TaskManager Flink SQL-Gateway
	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC
	Hive	Hive Metastore
	Impala	Impala Statestore Impala Catalog Service
	Monitoring	Grafana Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Recon Ozone HttpFS
	Solr	Solr Server
	Spark3	Spark3 History server
	Trino	Trino Coordinator Trino Client
	YARN	MapReduce History server
	ZooKeeper	ZooKeeper Server
Host 2	Airflow2	Airflow2 Broker Airflow2 Flower Airflow2 Scheduler Airflow2 Server Airflow2 Worker
	Flink	Flink TaskManager
	HBase	HBase Master server HBase Phoenix query server
	HDFS	HDFS JournalNode HDFS HttpFS server
	Hive	Hive HiveServer2 Hive TezUI
	Monitoring	Prometheus Server Node Exporter
	Solr	Solr Server Solr prometheus exporter
	Spark3	Spark3 Livy server
	YARN	YARN ResourceManager
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 3	HBase	HBase Thrift2 server HBase REST Server
	HDFS	HDFS NameNode HDFS ZKFC HDFS JournalNode
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone Datanode
	Solr	Solr Server
	Trino	Trino Worker Trino Client
	YARN	YARN Timeline server
	Zeppelin	Zeppelin Server
	ZooKeeper	ZooKeeper Server
Host 4	Arenadata PostgreSQL	Arenadata PostgreSQL
	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC HDFS HttpFS server
	HBase	HBase Master server
	HUE	HUE Server
	Kyuubi	Kyuubi Server
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	SSM	SSM Server SSM Agent
	Trino	Trino Worker Trino Client
	ZooKeeper	ZooKeeper Server
Host 5	HDFS	HDFS JournalNode HDFS NameNode HDFS ZKFC
	HBase	HBase Master server
	Monitoring	Node Exporter
	Ozone	Ozone Manager Ozone Storage Container Manager Ozone S3G
	Solr	Solr Server
	Trino	Trino Worker Trino Client
	ZooKeeper	ZooKeeper Server
Host 6	Flink	Flink Client Flink History Server
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon Impala Client
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 7	Flink	Flink Client
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Solr	Solr Server
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 8	Flink	Flink Client
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server
Host 9	Flink	Flink Client
	HDFS	HDFS DataNode HDFS Client
	HBase	HBase Client HBase RegionServer
	Hive	Hive Tez Hive Client
	Impala	Impala Daemon
	Monitoring	Node Exporter
	Ozone	Ozone Datanode
	Spark3	Spark3 Client
	Trino	Trino Worker Trino Client
	YARN	YARN NodeManager YARN Client
	ZooKeeper	ZooKeeper Server

4. After component distribution is completed, click ***Save***.



Save mapping of components

Starting with ADCM 2.2.0, you can use another mapping mode — by hosts. To do this, switch on the ***Hosts mode*** toggle and map components to hosts by clicking ***Add components*** for each host sequentially.



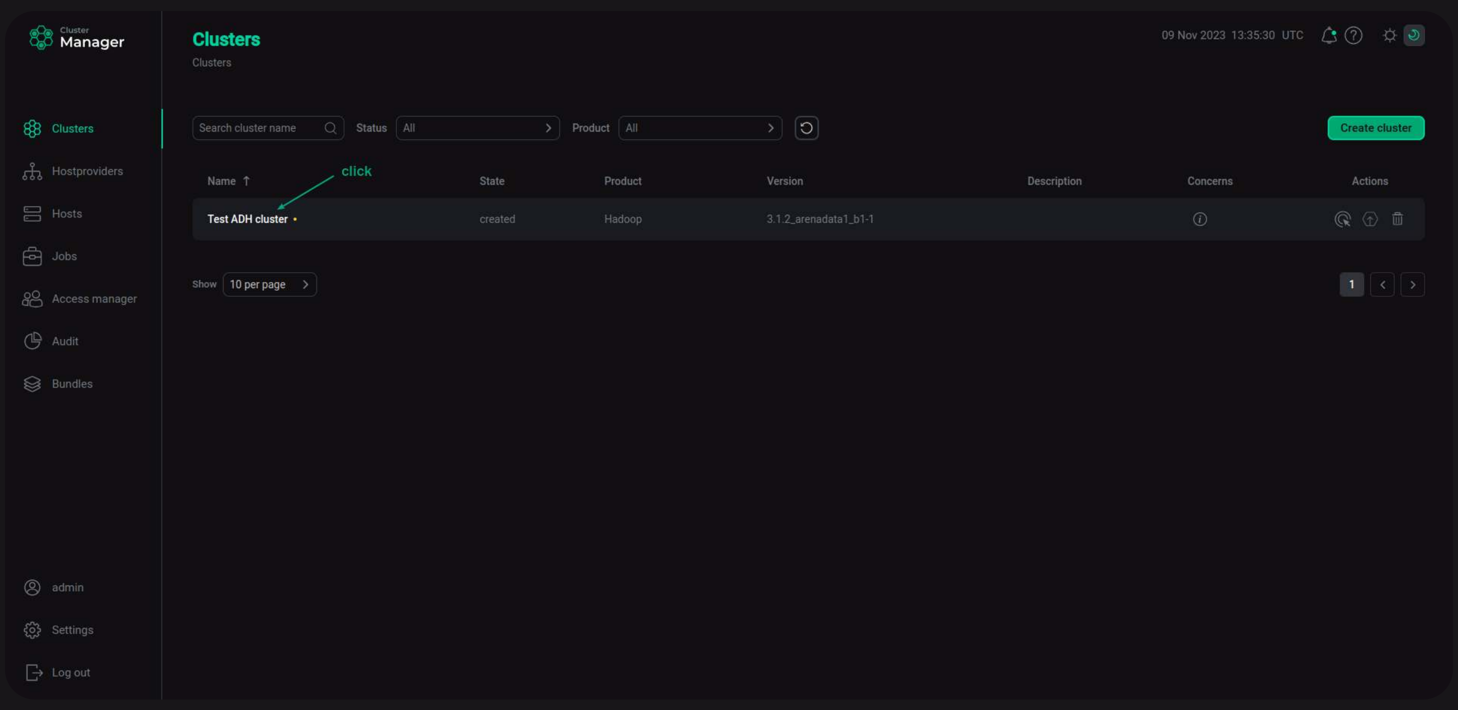
Use mapping by hosts

Configure services

Eugenia Kuzina, Daria Barysheva

To Table of Contents

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.

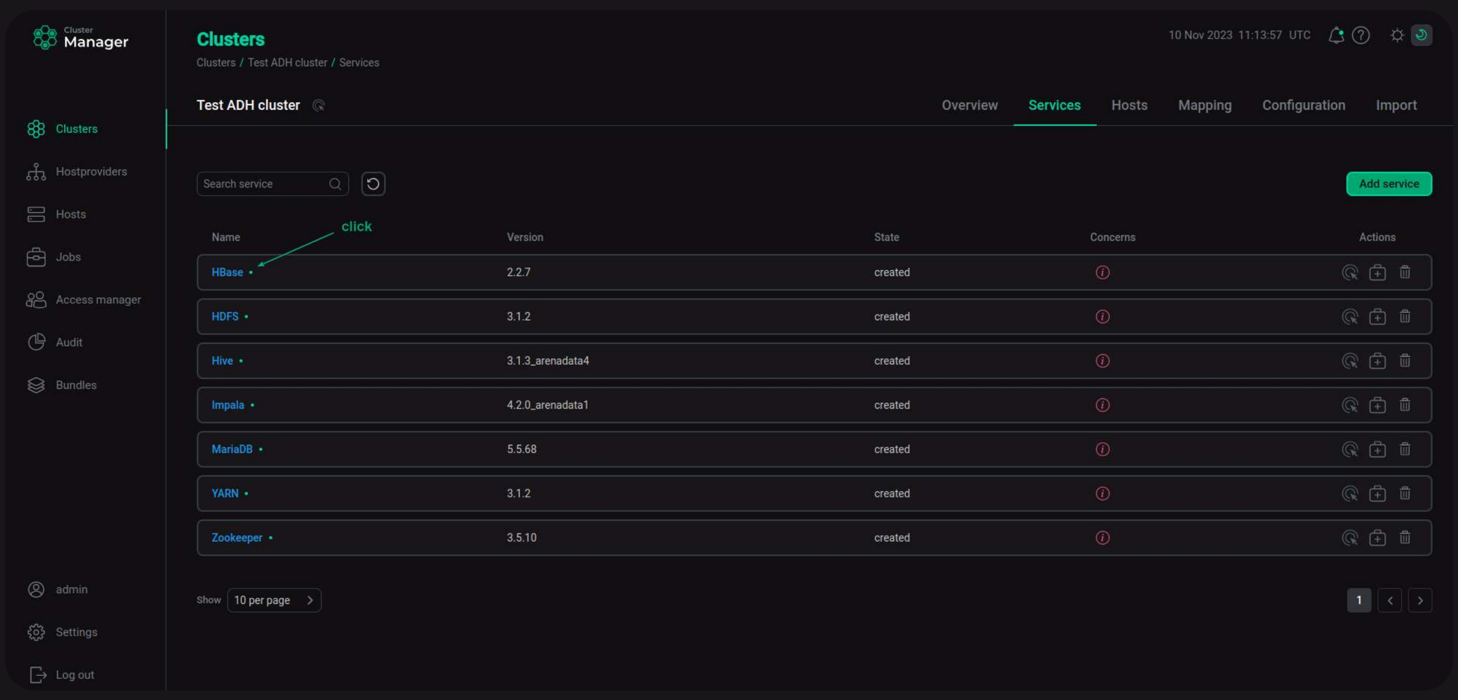


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name ↑	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

Select a cluster

2. Open the ***Services*** tab on the cluster page, find the row that contains the desired service, and click its name in the ***Name*** column.

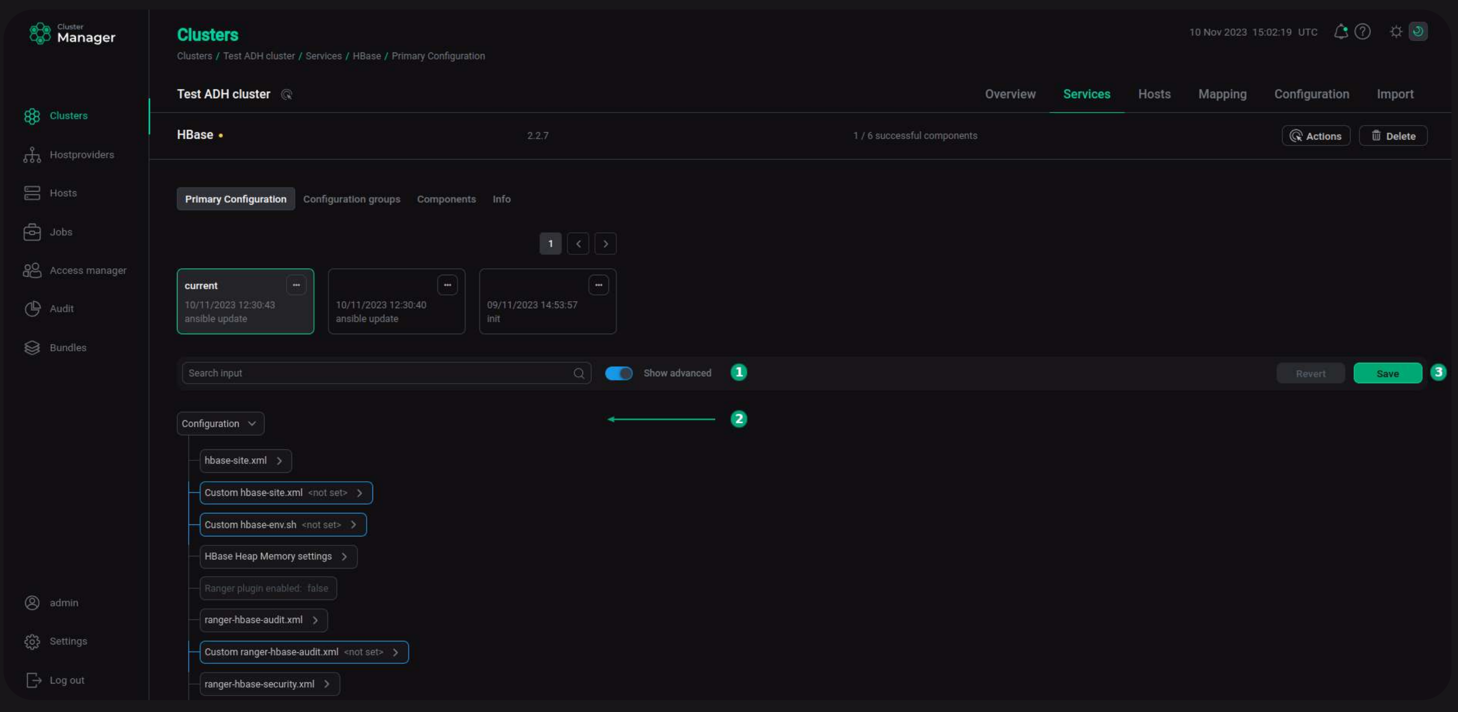


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Services
10 Nov 2023 11:13:57 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
click
Show
10 per page
1

Name	Version	State	Concerns	Actions
HBase	2.2.7	created		
HDFS	3.1.2	created		
Hive	3.1.3_arenadata4	created		
Impala	4.2.0_arenadata1	created		
MariaDB	5.5.68	created		
YARN	3.1.2	created		
Zookeeper	3.5.10	created		

Switch to configuring a service

3. In the window that opens:
   - Open the ***Primary Configuration*** tab.
   - Switch on the ***Show advanced*** toggle.
   - Fill in configuration parameters of the selected service. The fields highlighted in red are mandatory.
   - Click ***Save***.



Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Services / HBase / Primary Configuration
10 Nov 2023 15:02:19 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
HBase
2.2.7
1 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1
current
10/11/2023 12:30:43
ansible update
10/11/2023 12:30:40
ansible update
09/11/2023 14:53:57
init
Search input
Show advanced
1
Revert
Save
3
2
Configuration
hbase-site.xml
Custom hbase-site.xml <not set>
Custom hbase-env.sh <not set>
HBase Heap Memory settings
Ranger plugin enabled: false
ranger-hbase-audit.xml
Custom ranger-hbase-audit.xml <not set>
ranger-hbase-security.xml

Configure a service

NOTE

For more information about services configuration parameters, see Configuration parameters in the References.

Configure a cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

Primary configuration

Configuration groups

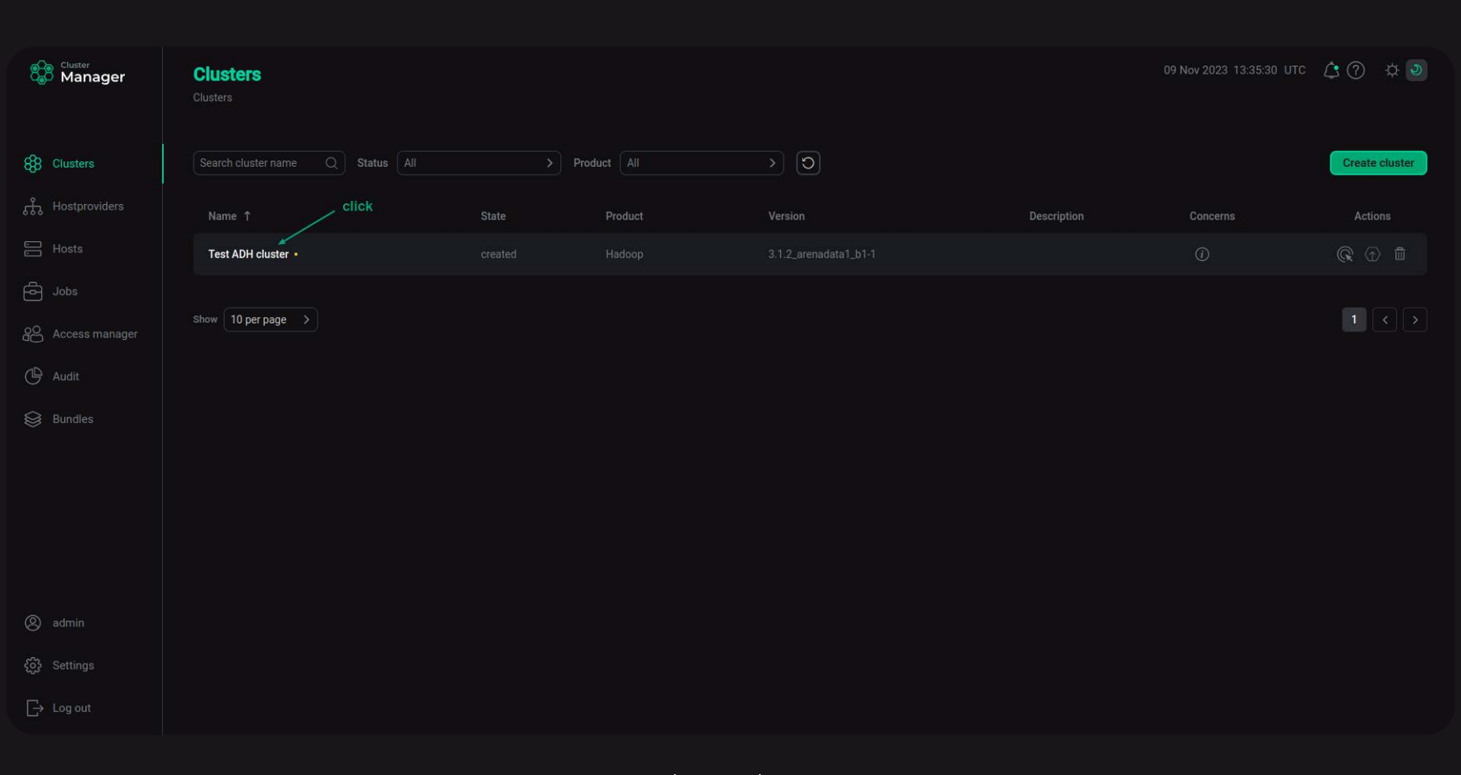
Ansible settings

IMPORTANT

There is usually no need to change cluster configuration parameters. You can leave all parameters at the default values.

After creation a new cluster, you can configure it by performing the following steps:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.



Cluster Manager
Clusters
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
09 Nov 2023 13:35:30 UTC
Search cluster name
Status
All
Product
All
Create cluster
Name
click
State
Product
Version
Description
Concerns
Actions
Test ADH cluster
created
Hadoop
3.1.2_arenadata1_b1-1
Show
10 per page
1

Select a cluster

2. Open the ***Configuration*** tab on the cluster page. The ***Configuration*** tab includes the following sections: ***Primary configuration***, ***Configuration groups***, ***Ansible settings***.
3. Fill in all necessary parameters on the selected tab and click ***Save***.

Primary configuration

The ***Configuration*** → ***Primary configuration*** tab allows you to configure the general cluster settings.

Available configuration parameters are listed below:

- The ***Reliability control*** section. It is displayed if the ***Show advanced*** switch is activated. The section contains the following parameters:
  - ***retries*** — the number of retries for cluster operations (20 by default);
  - ***delay*** — the delay between retries in seconds (10 by default);
  - ***timeout*** — timeout for checks of network ports in seconds (60 by default).
- Paths to the repositories that are used in the process of installing — depending on the cluster operating system (CentOS 7, Red Hat 7, Ubuntu 22, ALT Linux 8, Astra Linux, RedOS 7):
  - ***monitoring*** — monitoring components repository;
  - ***arenadata*** — Arenadata YUM repository;
  - ***arenadata_postgres*** — Arenadata PostgreSQL YUM repository;
  - ***prometheus_repo*** — Prometheus YUM repository;
  - ***arenadata_enterprise*** — Arenadata Enterprise YUM repository;
- The ***Docker registry*** parameter that specifies a Docker Registry address. The default value is *hub.adsw.io* — Docker Registry from the Arenadata repository.
- The ***Kerberos*** section that includes settings listed in the table below. Some options are visible only with the ***Show advanced*** option enabled.

Parameter	Description	Default value
Enable Kerberos	Enables the Kerberos authentication	False
Authentication on WEB UIs	Enables the Kerberos authentication on WEB UIs	False
SPNEGO Signature Value	A secret key that is used to generate and verify the signature in the HTTP authentication protocol SPNEGO (in the Base64 format)	—
Kerberos KDC type	A KDC type. For more information, see Kerberos	MIT
KDC hosts	One or more KDC hosts with running FreeIPA server(s)	—
Realm	A Kerberos realm to connect to the FreeIPA server	—
Domains	One or more domains associated with FreeIPA	—
Kadmin server	A host where `kadmin` is running	—
Kadmin principal	A principal name used to connect via `kadmin`, for example `admin@RU-CENTRAL1.INTERNAL`	—
Kadmin password	An IPA Admin password	—
Keytabs directory	A directory with keytab files	/etc/security/keytabs
Additional realms	Additional Kerberos realms	—
Delay between kinit invocation attempts	Delay between kinit invocation attempts	5
Number of retries for kinit invocation attempts	Number of retries for kinit invocation attempts	10
Trusted Active Directory server	An Active Directory server for one-way cross-realm trust from the MIT Kerberos KDC	—
Trusted Active Directory realm	An Active Directory realm for one-way cross-realm trust from the MIT Kerberos KDC	—
Custom krb5.conf	Enables adding custom parameters to the *krb5.conf* file	False
krb5.conf	Additional parameters to write to the *krb5.conf* file	—
Admin DN	A full DistinguishedName of the admin user with rights to `create/modify/delete/pwdchange` user accounts in the target Organizational Unit	—
LDAP URL	LDAP URL that consists of the `ldap://` or `ldaps://` protocol prefix, hostname or IP address and port of an AD server. For example, `ldaps://192.168.4.2:636`	—
Container DN	A container distinguished name	—
TLS CA certificate Path	A CA certificate path on the host filesystem that is pre-located or should be written from the ***TLS CA certificate*** field	—
TLS CA certificate (optional)	A CA certificate that is written on a host by the path from ***TLS CA certificate Path*** during the execution of the ***Manage Kerberos*** action. The certificate is not saved in ADCM	—
Custom ldap.conf	Enables adding custom parameters to the *ldap.conf* file	False
ldap.conf	Additional parameters that are written to the *ldap.conf* file	—
IpaClient No NTP Autoconf	Disables the NTP configuration during the IPA client installation	False
IpaClient No DNS Lookup	Disables the DNS lookup for the FreeIPA server during the IPA client installation	True
Custom smb.conf	Usage of custom configuration file *smb.conf*	False
smb.conf	Custom configuration file *smb.conf*	—
Set up Kerberos utils	Enables installation or removal of Kerberos clients and utils. Affects the ***Expand*** and ***Install*** actions	True
Configure Kerberos on hosts	Enables cluster configuration, including *krb5.conf*, *ldap.conf*	True
Set up principals and keytabs	Enables creation, recreation, or removal of principals and keytabs. Passwords for principals are generated randomly before keytab creation. Affects the ***Expand*** and ***Install*** actions. ADCM bundle will set up owner and permissions for keytabs only if this checkbox is selected in the cluster configuration. In case of absence of admin permissions, a customer should provide the prepared keytabs with correctly set owner and permissions (see Custom keytab recommendations)	True
Configure services and clients	Enables updating of services and clients configuration	True
Run service checks	Enables service check runs	True

- The ***Credential encryption*** section with the encryption provider settings:
  - ***Credstore password*** — encryption provider password;
  - ***Credstore options*** — the way of storing the encryption provider password: `no password`, `password-file`, or `password in the environment`.
- The ***SSL*** section with the settings listed in the table below.

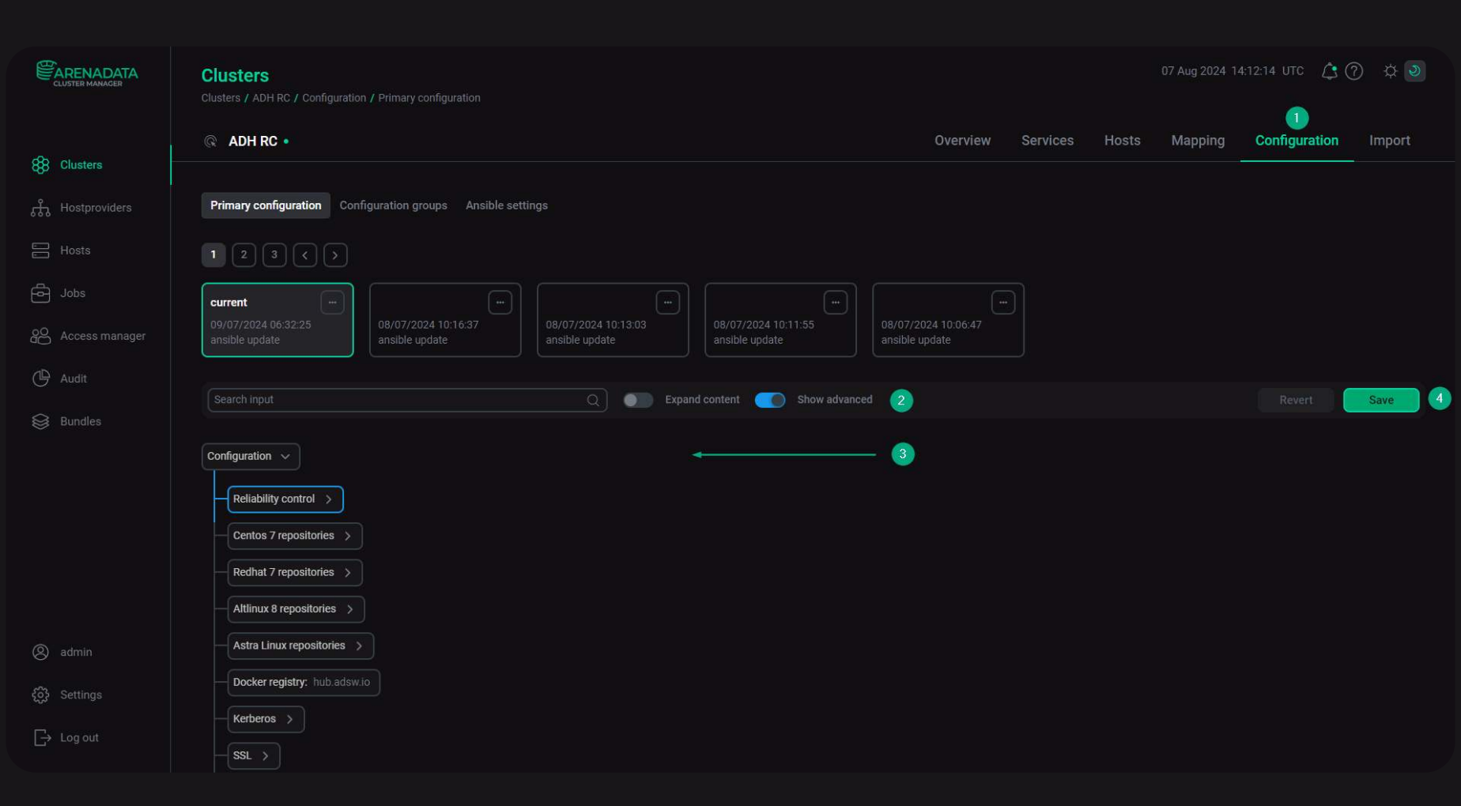
Parameter	Description	Default value
Enable SSL	Enables SSL	false
Keystore path	A path to the keystore file	/etc/ssl/keystore.jks
Keystore password	A password for the keystore file	—
Truststore path	A path to the truststore file	/etc/ssl/truststore.jks
Truststore password	A password for the truststore file	—
TLS Version	Version of the TLS protocol. See Set TLS version for an ADH cluster	TLSv1.2
Run service checks	Enables service check runs	true

- The ***java.io.tmpdir*** parameter that specifies an environment variable to control `java.io.tmpdir` for YARN, HBase, Hive, Spark3.
- The ***ad-runtime-utils*** parameter contains a configuration file template that specifies the runtime detection installation path settings for a given runtime (Java, Python, etc.). The template contains cluster settings as well as service-specific settings for Trino, Flink, and Ozone. You can use this parameter to explicitly define the Java version. For more information, see Software Requirements — Java.
  - The `default` section configures the default behavior for the cluster.
  - The `autodetect` section defines the algorithm for automatically detecting different Java versions. For each version (8, 17, 21, 23, 24), the following is configured:
    - `env_var` — an environment variable that specifies version (for example, `JAVA8_HOME`, `JAVA17_HOME`);
    - `paths` — a list of paths to search for the Java installation.
  - The `services` section defines services for which specific Java versions are required.

ad-runtime-utils

```
default:
  runtimes:
    java:
      version: "8"
      env_var: JAVA_HOME

autodetect:
  runtimes:
    java:
      "8":
        env_var: JAVA8_HOME
        paths:
          - /usr/java/jdk1.8
          - /usr/java/jre1.8
          - /usr/lib/jvm/j2sdk1.8-oracle
          - /usr/lib/jvm/j2sdk1.8-oracle/jre
          - /usr/lib/jvm/java-8-oracle
          - /usr/lib/jdk8-latest
          - /usr/lib/jvm/java-1.8.0
          - /usr/lib/jvm/java-1.8.0-oracle
          - /usr/lib/jvm/bellsoft-java8-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-8

      "17":
        env_var: JAVA17_HOME
        paths:
          - /usr/lib/jvm/bellsoft-java17-amd64
          - /usr/lib/jvm/bellsoft-java17
          - /usr/lib/jvm/liberica-jdk-17*
          - /usr/lib/jvm/liberica-jre-17*
          - /usr/lib/jvm/zulu-17*
          - /usr/lib/jvm/temurin-17*
          - /usr/lib/jvm/adoptopenjdk-17*
          - /usr/lib/jvm/jdk-17*
          - /usr/lib/jvm/java-17*
          - /usr/lib/jvm/openjdk-17*
          - /usr/lib/jvm/openjdk-17*
          - /opt/java/bellsoft-java17-amd64
          - /opt/java/temurin-17*
          - /usr/local/java/bellsoft-java17-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-17
          - /usr/lib64/jvm/java-17*

      "21":
        env_var: JAVA21_HOME
        paths:
          - /usr/lib/jvm/java-21*
          - /usr/lib/jvm/jdk-21*
          - /usr/lib/jvm/java-arenadata-openjdk-21

      "23":
        env_var: JAVA23_HOME
        paths:
          - /usr/lib/jvm/bellsoft-java23-amd64
          - /usr/lib/jvm/bellsoft-java23
          - /usr/lib/jvm/liberica-jdk-23*
          - /usr/lib/jvm/liberica-jre-23*
          - /usr/lib/jvm/zulu-23*
          - /usr/lib/jvm/temurin-23*
          - /usr/lib/jvm/adoptopenjdk-23*
          - /usr/lib/jvm/jdk-23*
          - /usr/lib/jvm/java-23-openjdk
          - /usr/lib/jvm/openjdk-23*
          - /opt/java/bellsoft-java23-amd64
          - /opt/java/temurin-23*
          - /usr/local/java/bellsoft-java23-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-23

      "24":
        env_var: JAVA24_HOME
        paths:
          - /usr/lib/jvm/bellsoft-java24-amd64
          - /usr/lib/jvm/bellsoft-java24
          - /usr/lib/jvm/liberica-jdk-24*
          - /usr/lib/jvm/liberica-jre-24*
          - /usr/lib/jvm/zulu-24*
          - /usr/lib/jvm/temurin-24*
          - /usr/lib/jvm/adoptopenjdk-24*
          - /usr/lib/jvm/jdk-24*
          - /usr/lib/jvm/java-24-openjdk
          - /usr/lib/jvm/openjdk-24*
          - /opt/java/bellsoft-java24-amd64
          - /opt/java/temurin-24*
          - /usr/local/java/bellsoft-java24-amd64
          - /usr/lib/jvm/java-arenadata-openjdk-24

services:
  TRINO:
      path: /etc/trino/conf/trino-java.yaml
  FLINK2:
      path: /etc/flink2/conf/flink2-java.yaml
  OZONE:
      path: /etc/ozone/conf/ozone-java.yaml
  SPARK4:
      path: /etc/spark4/conf/spark4-java.yaml
```

- The ***Precheck packages*** parameter that specifies whether the availability of the packages in the repository should be checked before installation. `False` by default.
- ***Cluster UUID*** — displays the universally unique identifier of the cluster.
- The ***Ranger plugins version*** parameter that allows you to override the version number for the Ranger plugins.



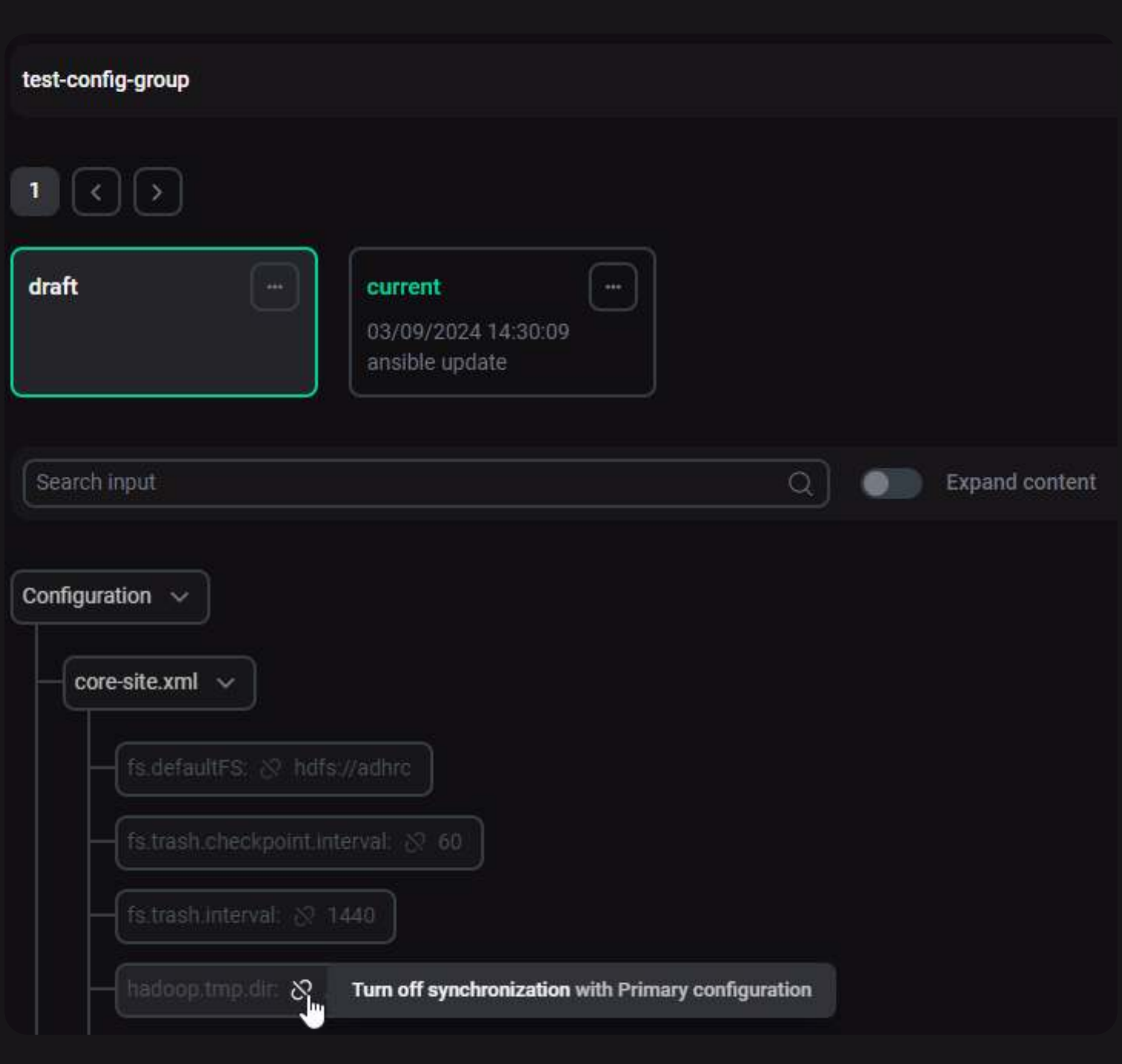
ARENADATA
Clusters
ADH RC
Overview
Services
Hosts
Mapping
Configuration
Import
Primary configuration
Configuration groups
Ansible settings
current
Search input
Expand content
Show advanced
Revert
Save
Configuration
Reliability control
Centos 7 repositories
Redhat 7 repositories
Altlinux 8 repositories
Astra Linux repositories
Docker registry: hub.adsw.io
Kerberos
SSL

The cluster configuration window

Configuration groups

The ***Configuration*** → ***Configuration groups*** tab is designed to set cluster configuration groups.

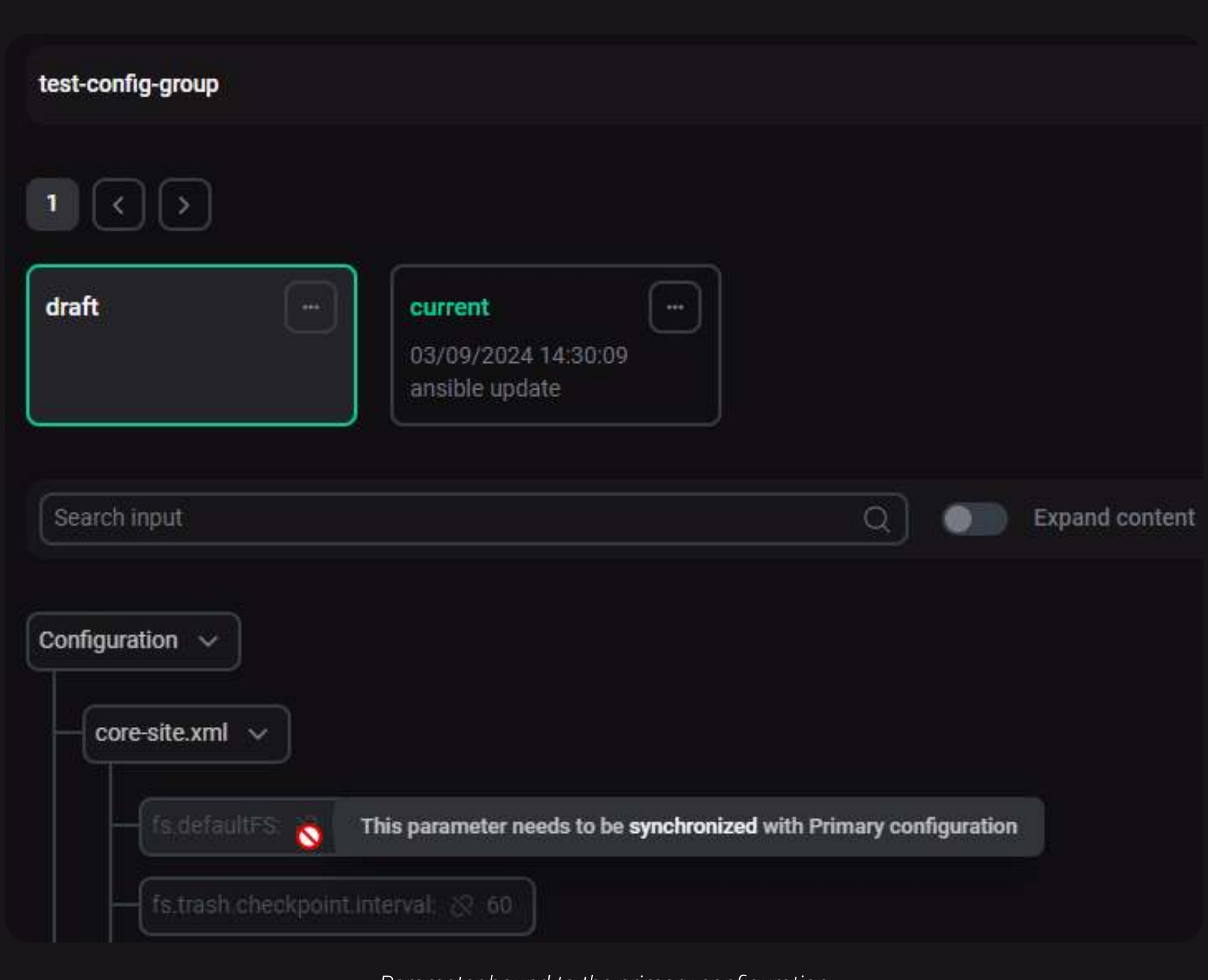
To configure parameters in a configuration group, hover over the parameters that you wish to configure differently, click the ***Turn off synchronization*** icon, edit them, and click ***Save***.



test-config-group
1
draft
current
03/09/2024 14:30:09
ansible update
Search input
Expand content
Configuration
core-site.xml
fs.defaultFS: hdfs://adhrc
fs.trash.checkpoint.interval: 60
fs.trash.interval: 1440
hadoop.tmp.dir:
Turn off synchronization with Primary configuration

Configuration tab on the config group page

Some parameters you cannot change within a configuration group as they must always be synchronized with the primary configuration.



test-config-group
1
draft
current
03/09/2024 14:30:09
ansible update
Search input
Expand content
Configuration
core-site.xml
fs.defaultFS:
This parameter needs to be synchronized with Primary configuration
fs.trash.checkpoint.interval: 60

Parameter bound to the primary configuration

Ansible settings

The ***Configuration*** → ***Ansible settings*** tab allows you to set Ansible configuration options at the cluster level. The tab is available starting with ADCM 2.2.0.

Ansible settings

Name	Description	Default
forks	The number of parallel processes to spawn when communicating with remote hosts	5

Import ET settings

Eugenia Kuzina, Daria Barysheva

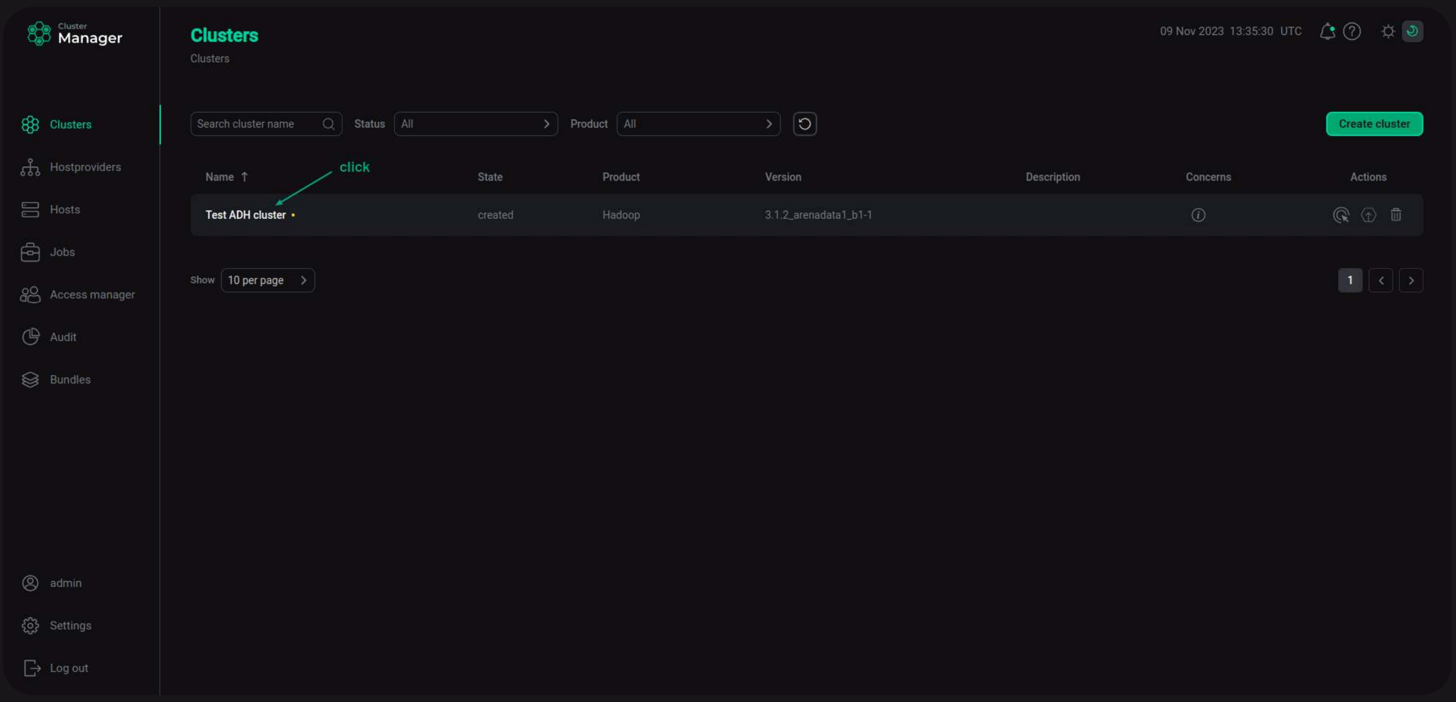
To Table of Contents

IMPORTANT

This step should be performed only in the case of offline installation.

In order to determine that a new cluster should be installed offline, it is necessary to integrate it with the previously created Enterprise Tools (ET) cluster. Follow the steps:

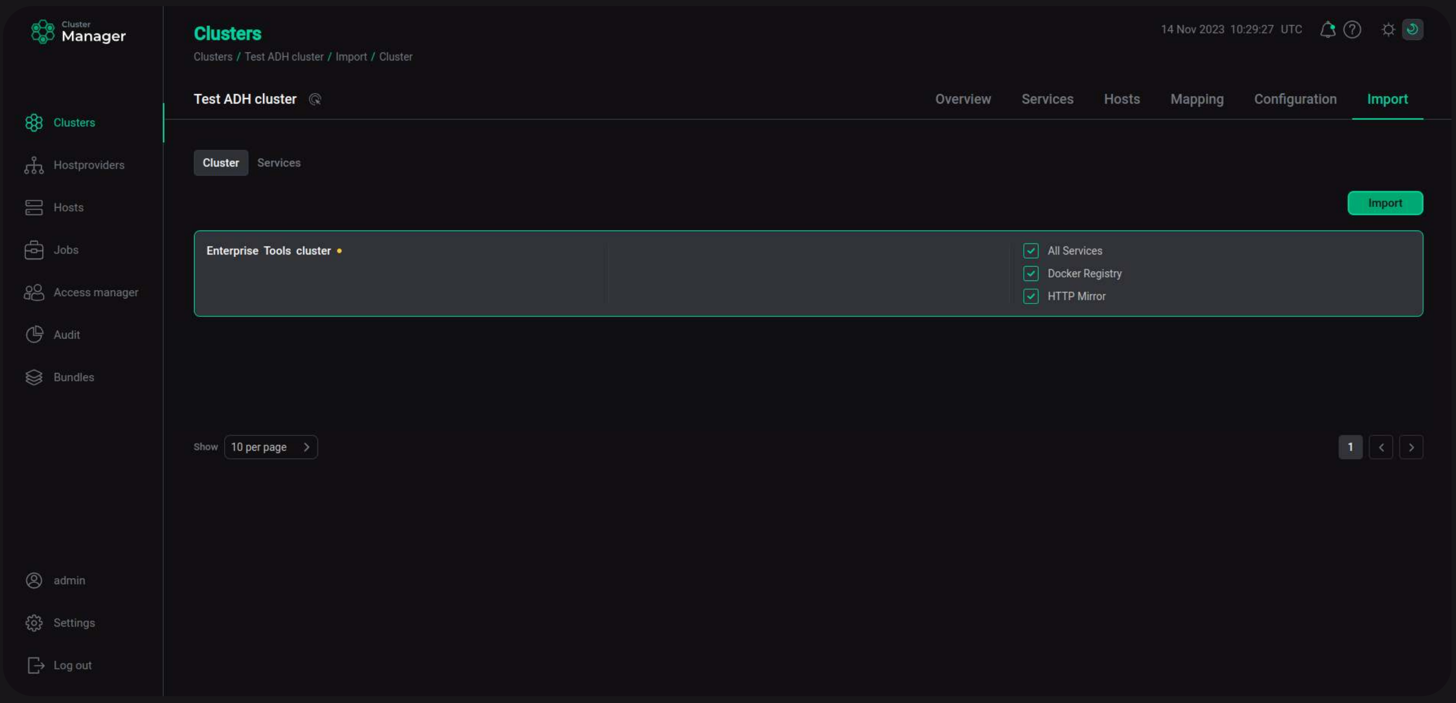
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
09 Nov 2023 13:35:30 UTC
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

Select a cluster

2. Open the ***Import*** tab in the cluster menu.
3. Select ***Docker Registry*** and ***HTTP Mirror*** services of the previously created ET cluster.
4. Click ***Import***.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADH cluster / Import / Cluster
14 Nov 2023 10:29:27 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Cluster
Services
Import
Show
10 per page
1

Enterprise Tools cluster

☑ All Services
☑ Docker Registry
☑ HTTP Mirror

Integration with a ET cluster

NOTE

After successful import, the data stored locally in the Enterprise Tools cluster can be used at the cluster installation stage. So there is no need to extract this data via Internet.

Install a cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

Step 1. Installation
- Whole cluster
- Single services

Step 2. View installation results

Step 3. Check the cluster state after the installation

Step 1. Installation

There are two ways to install cluster services:

1. **Whole cluster**. In this method all services are installed automatically one by one after applying the ***Install*** action to a cluster.
2. **Single services**. In this method each service is installed manually after applying one or more actions to it. This way is also suitable for installing new services in the already deployed cluster.

In both cases each service starts automatically after installation.

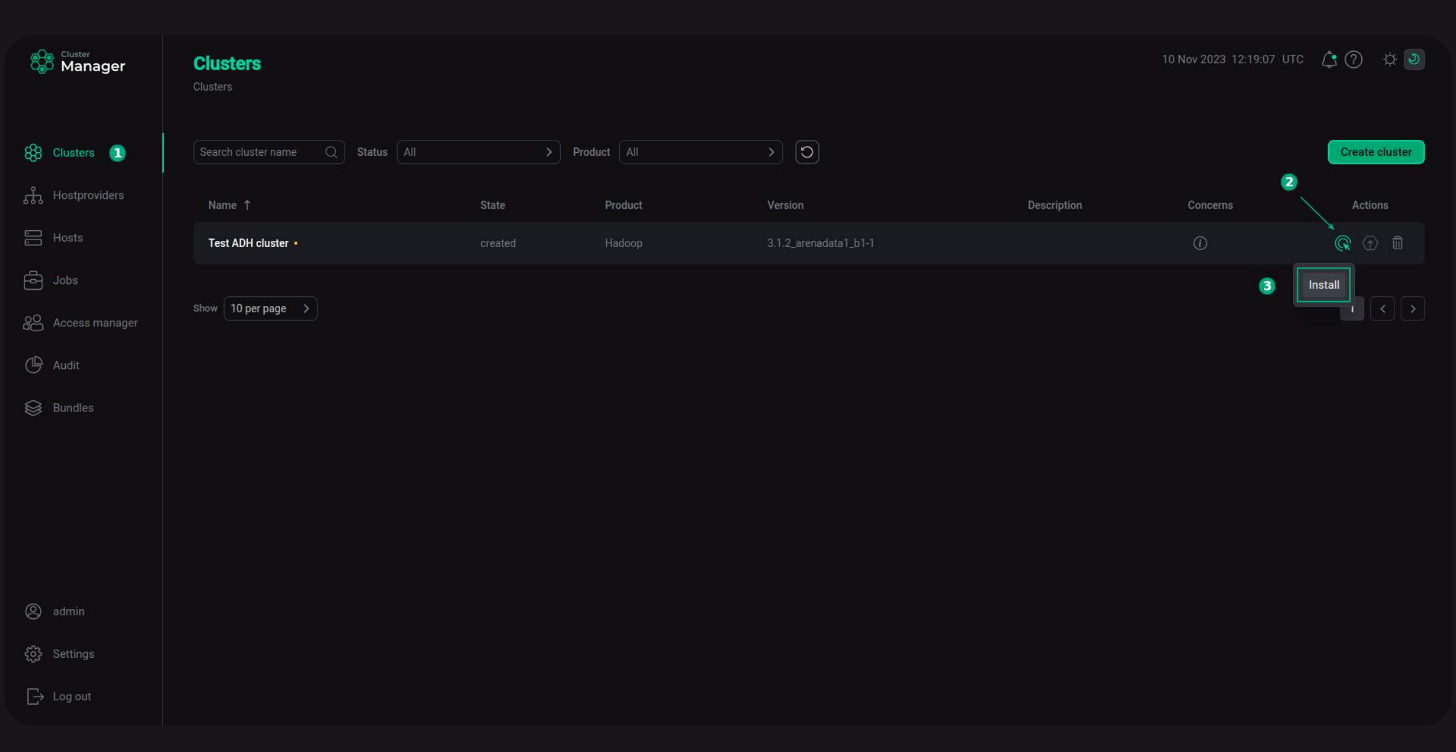
IMPORTANT

The installation must be performed on a clean host without prior installation of related or similar software.

Whole cluster

To install all cluster services within a single action, follow the steps:

1. Apply the ***Install*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.

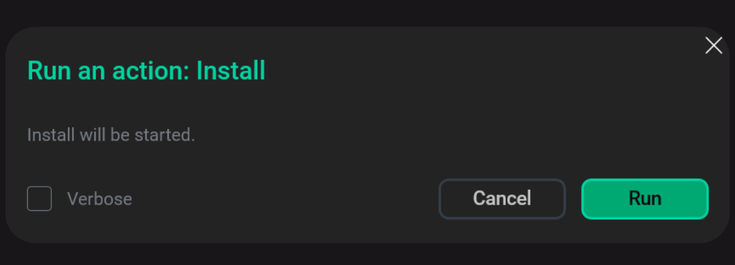


Cluster Manager
Clusters
Clusters
10 Nov 2023 12:19:07 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
2
3
Install
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	created	Hadoop	3.1.2_arenadata1_b1-1			

Switch to the whole cluster installation

2. Verify the requested action in the opened window.



Run an action: Install
☐ Verbose
Cancel
Run

Install will be started.

Verify the action

3. Wait until the job is completed.

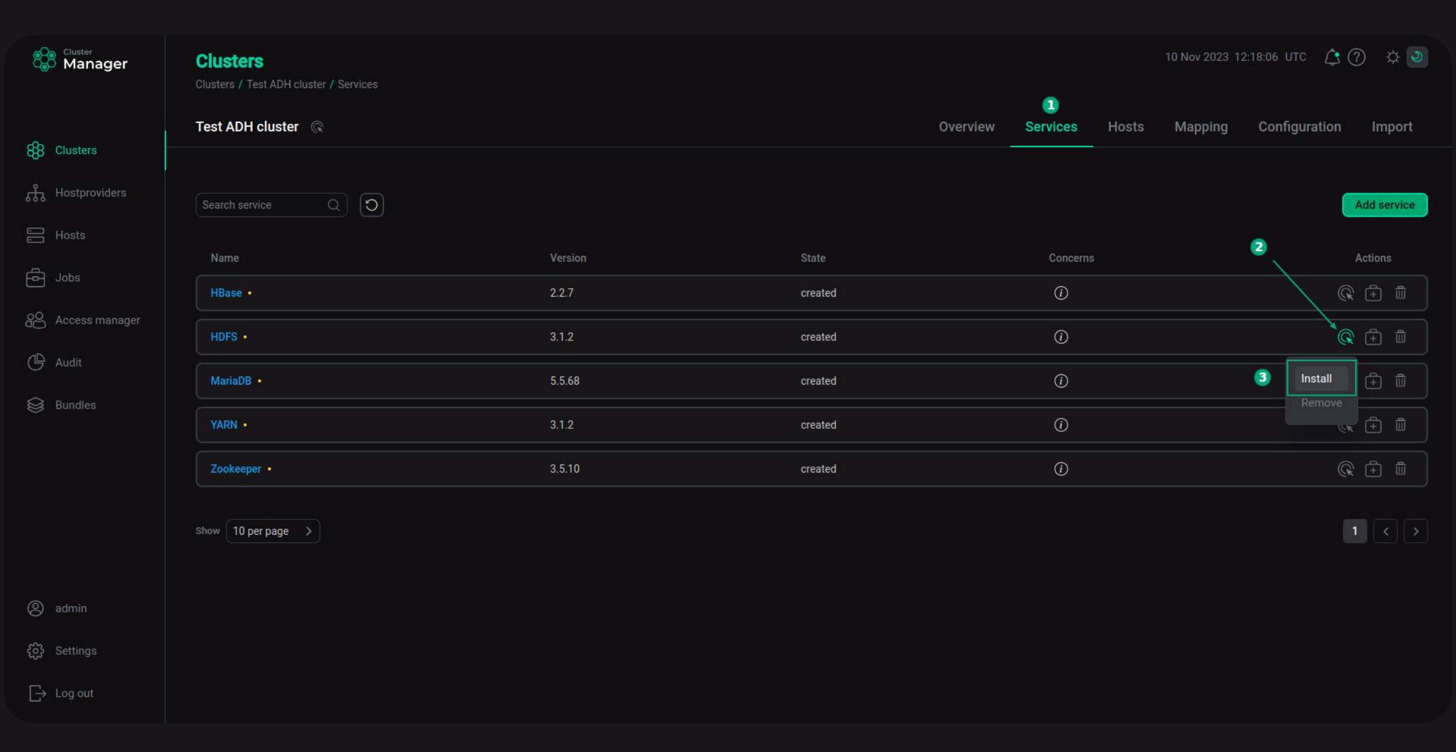
Single services

To install one or several cluster services manually, add these services to the cluster and then perform the actions described below.

CAUTION

Notice that services are installed one by one. Do not install another service until the installation of the previous one is completed.

1. Open the ***Services*** tab on the cluster page. For each service that should be installed, click the icon in the ***Actions*** column and select the ***Install*** action.

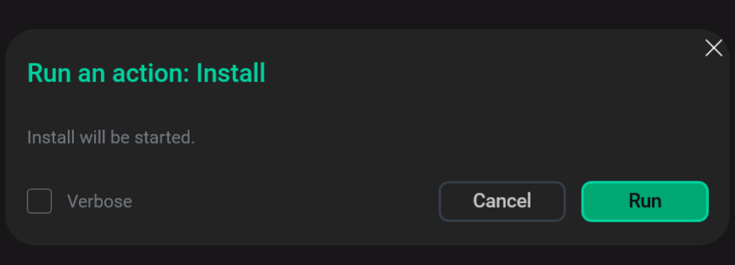


Cluster Manager
Clusters
Clusters / Test ADH cluster / Services
10 Nov 2023 12:18:06 UTC
Test ADH cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
2
3
Install
Remove
Show 10 per page

Name	Version	State	Concerns	Actions
HBase	2.2.7	created		
HDFS	3.1.2	created		
MariaDB	5.5.68	created		
YARN	3.1.2	created		
Zookeeper	3.5.10	created		

Switch to installation of single services

2. Verify the action in the opened window.



Run an action: Install
☐ Verbose
Cancel
Run

Install will be started.

Verify the action

3. Wait until the job is completed. Then check that the service state has changed from `created` to `installed`.
4. Repeat the previous actions for other services that should be installed.

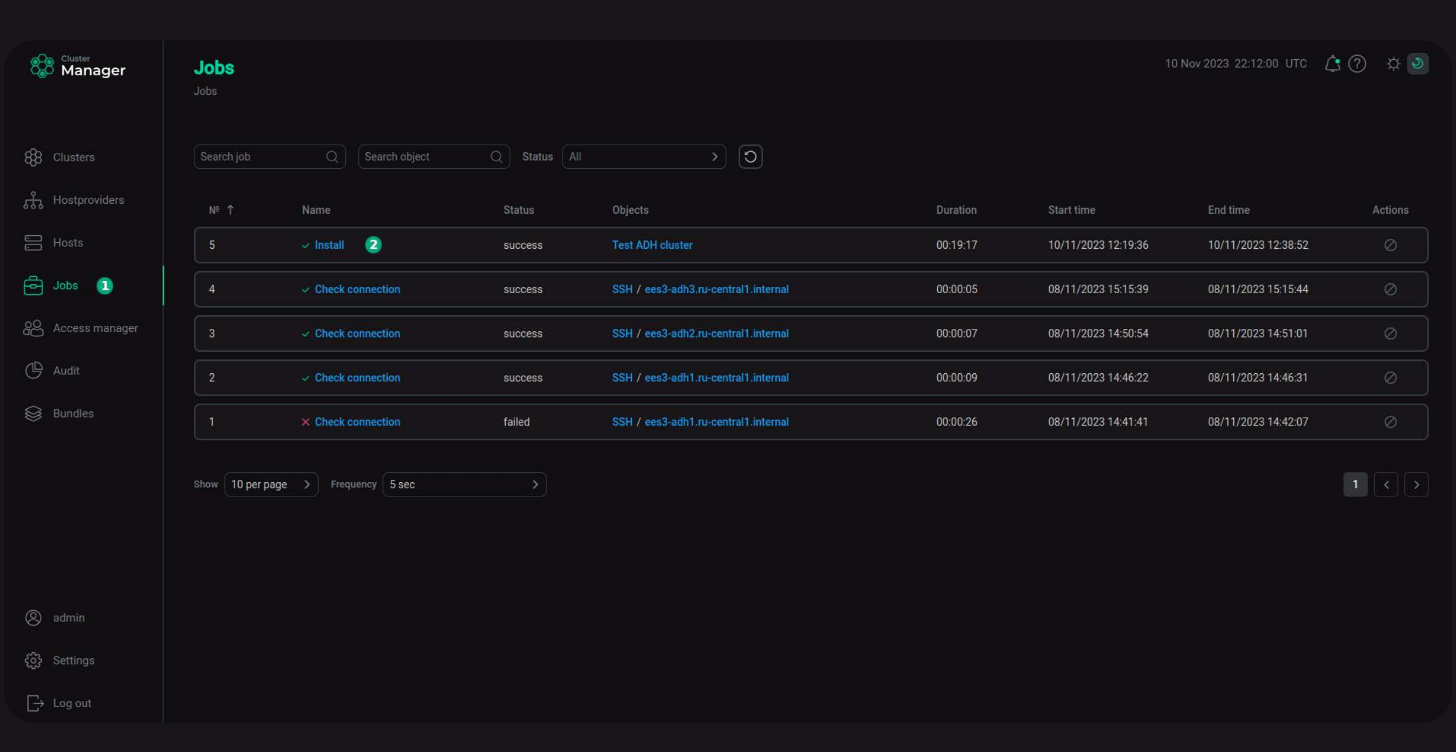
IMPORTANT

When you install services manually, one by one, use the following order:

1. Zookeeper
2. HDFS
3. YARN
4. HBase
5. Hive
6. Spark3
7. Monitoring

Step 2. View installation results

ADCM starts a single job for installation process, as well as for any other task. You can find out about the status of jobs more specifically on the ***Jobs*** page.

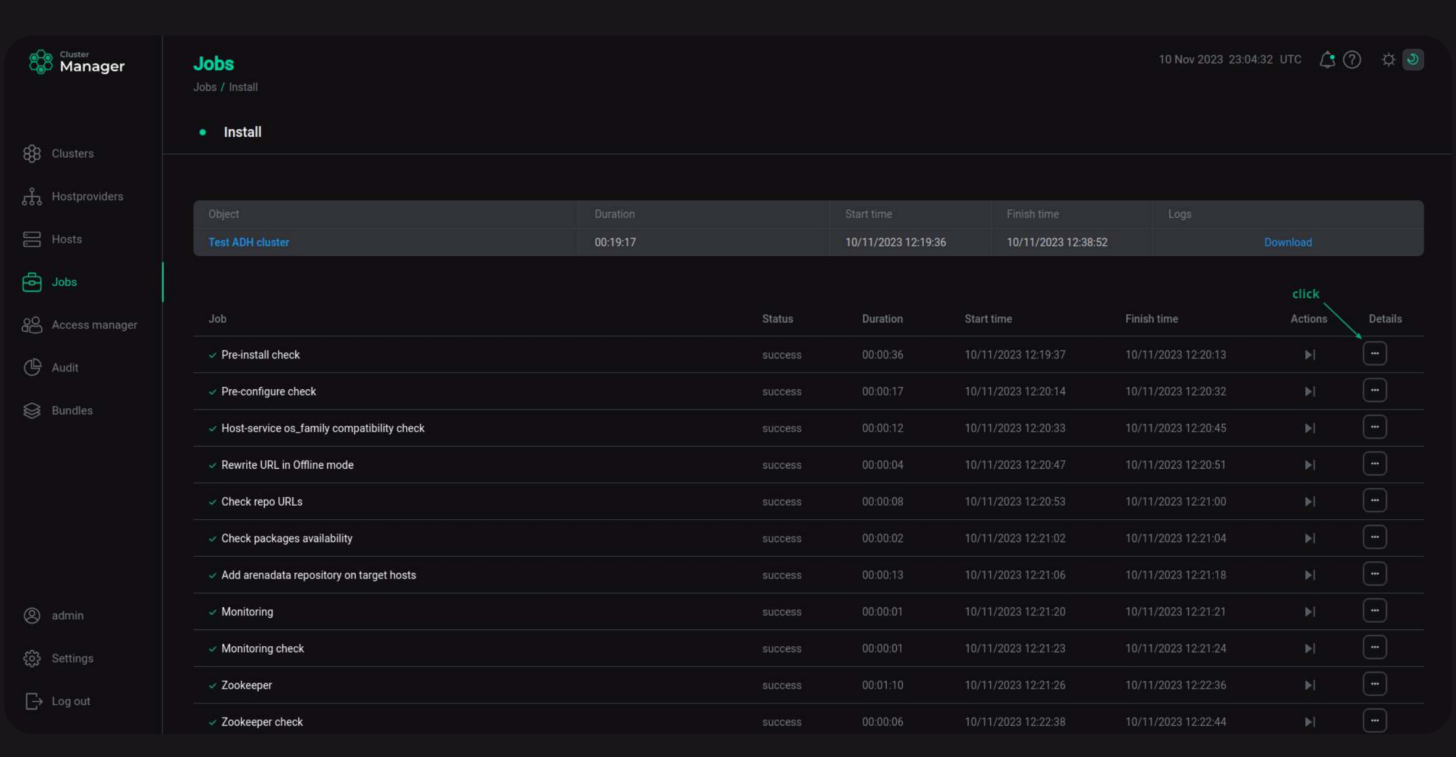


Cluster Manager
Jobs
Jobs
10 Nov 2023 22:12:00 UTC
Clusters
Hostproviders
Hosts
Jobs
1
Access manager
Audit
Bundles
admin
Settings
Log out
Search job
Search object
Status All
2
Show 10 per page
Frequency 5 sec

№	Name	Status	Objects	Duration	Start time	End time	Actions
5	✓ Install	success	Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	
4	✓ Check connection	success	SSH / ees3-adh3.ru-central1.internal	00:00:05	08/11/2023 15:15:39	08/11/2023 15:15:44	
3	✓ Check connection	success	SSH / ees3-adh2.ru-central1.internal	00:00:07	08/11/2023 14:50:54	08/11/2023 14:51:01	
2	✓ Check connection	success	SSH / ees3-adh1.ru-central1.internal	00:00:09	08/11/2023 14:46:22	08/11/2023 14:46:31	
1	✕ Check connection	failed	SSH / ees3-adh1.ru-central1.internal	00:00:26	08/11/2023 14:41:41	08/11/2023 14:42:07	

The Jobs page

The successful completion of the service installation is determined by the transition of the job from the `running` status to the `success` status on the ***Jobs*** page. If the job fails, it switches to the `failed` status. In this case it is possible to see more detailed information about the errors that occurred via clicking the failed job on the ***Jobs*** page. The window with job inner steps is opened. Click the ellipsis in the ***Details*** column for additional information on the current step.



Cluster Manager
Jobs
Jobs / Install
10 Nov 2023 23:04:32 UTC
Install
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
click

Object	Duration	Start time	Finish time	Logs
Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	Download

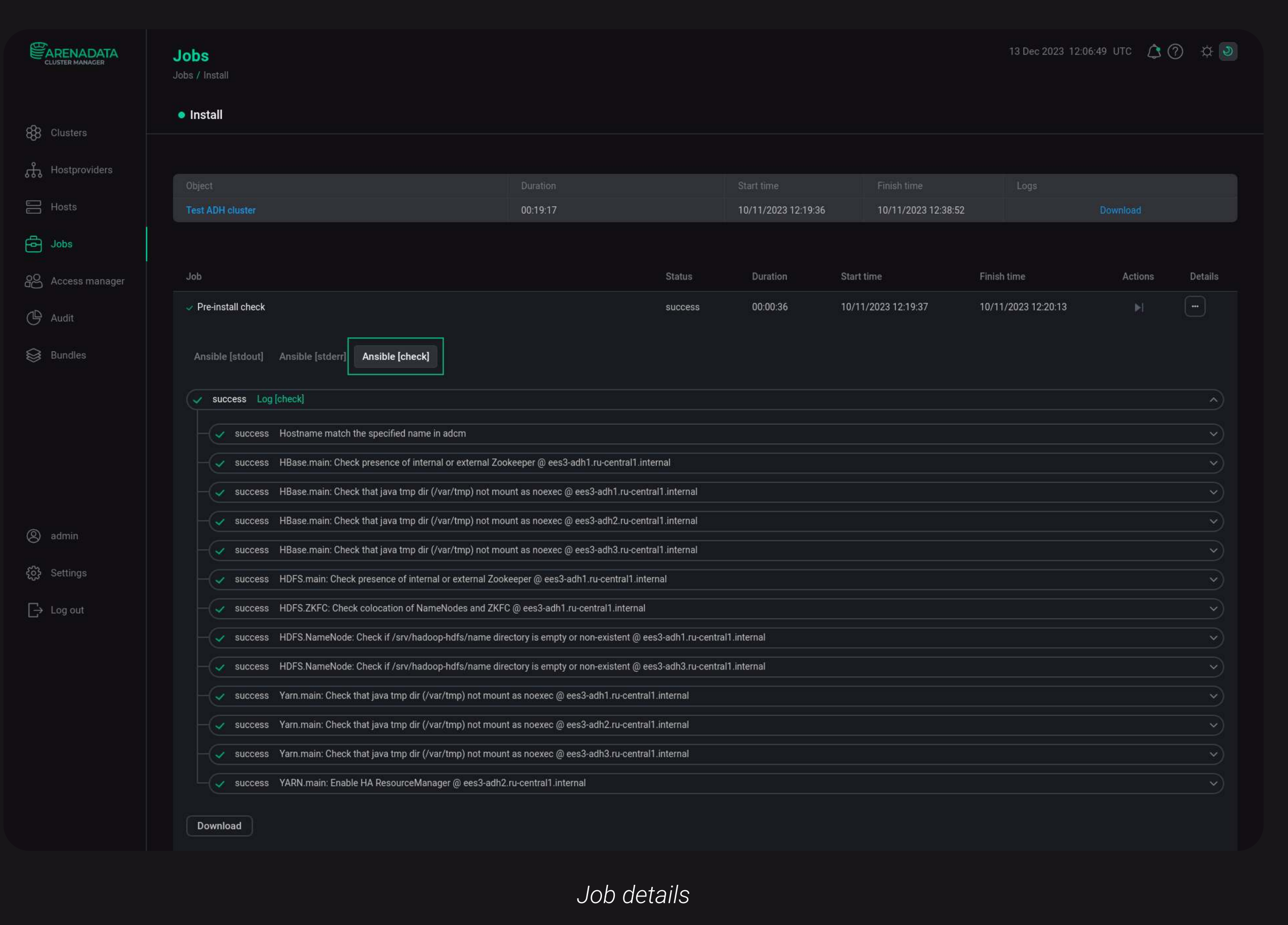
Job	Status	Duration	Start time	Finish time	Actions	Details
✓ Pre-install check	success	00:00:36	10/11/2023 12:19:37	10/11/2023 12:20:13		
✓ Pre-configure check	success	00:00:17	10/11/2023 12:20:14	10/11/2023 12:20:32		
✓ Host-service os_family compatibility check	success	00:00:12	10/11/2023 12:20:33	10/11/2023 12:20:45		
✓ Rewrite URL in Offline mode	success	00:00:04	10/11/2023 12:20:47	10/11/2023 12:20:51		
✓ Check repo URLs	success	00:00:08	10/11/2023 12:20:53	10/11/2023 12:21:00		
✓ Check packages availability	success	00:00:02	10/11/2023 12:21:02	10/11/2023 12:21:04		
✓ Add arenadata repository on target hosts	success	00:00:13	10/11/2023 12:21:06	10/11/2023 12:21:18		
✓ Monitoring	success	00:00:01	10/11/2023 12:21:20	10/11/2023 12:21:21		
✓ Monitoring check	success	00:00:01	10/11/2023 12:21:23	10/11/2023 12:21:24		
✓ Zookeeper	success	00:01:10	10/11/2023 12:21:26	10/11/2023 12:22:36		
✓ Zookeeper check	success	00:00:06	10/11/2023 12:22:38	10/11/2023 12:22:44		

Switch to job details

A job page contains two sections: ***ansible [stdout]*** and ***ansible [stderr]***. These are technical logs of the job that include information from standard I/O streams stdout/stderr. These logs can help to understand what caused the problems.

There can also be the optional third section ***ansible [check]***. These are the logs of the most frequent errors checks. The description of these errors is more simple and specific, than in two previous technical logs.

The contents of all three sections should be studied if errors occur.



ARENADATA CLUSTER MANAGER
Jobs
Jobs / Install
13 Dec 2023 12:06:49 UTC
Install
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Ansible [stdout]
Ansible [stderr]
Ansible [check]
Download

Object	Duration	Start time	Finish time	Logs
Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	Download

Job	Status	Duration	Start time	Finish time	Actions	Details
✓ Pre-install check	success	00:00:36	10/11/2023 12:19:37	10/11/2023 12:20:13		

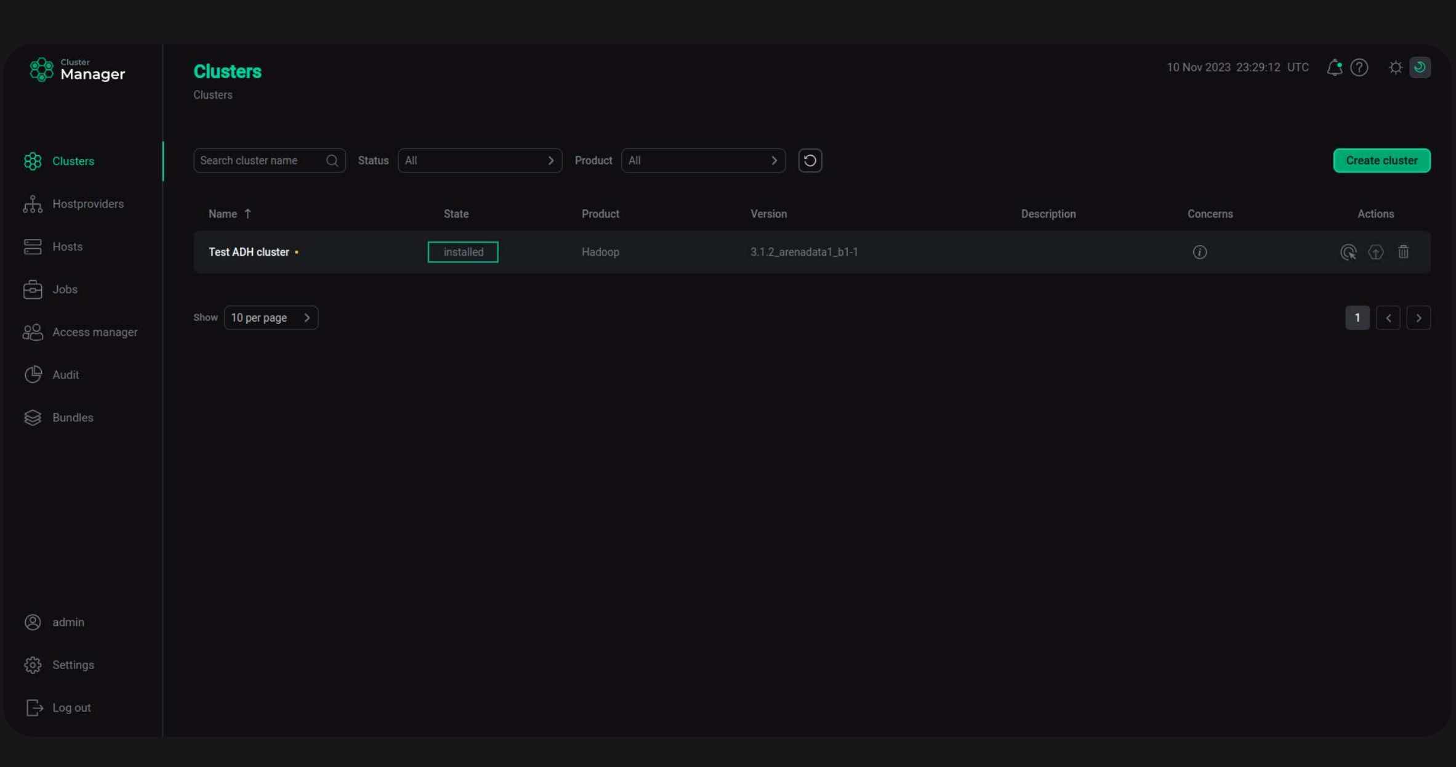
```
success Log [check]
  success Hostname match the specified name in adcm
  success HBase.main: Check presence of internal or external Zookeeper @ ees3-adh1.ru-central1.internal
  success HBase.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh1.ru-central1.internal
  success HBase.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh2.ru-central1.internal
  success HBase.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh3.ru-central1.internal
  success HDFS.main: Check presence of internal or external Zookeeper @ ees3-adh1.ru-central1.internal
  success HDFS.ZKFC: Check colocation of NameNodes and ZKFC @ ees3-adh1.ru-central1.internal
  success HDFS.NameNode: Check if /srv/hadoop-hdfs/name directory is empty or non-existent @ ees3-adh1.ru-central1.internal
  success HDFS.NameNode: Check if /srv/hadoop-hdfs/name directory is empty or non-existent @ ees3-adh3.ru-central1.internal
  success Yarn.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh1.ru-central1.internal
  success Yarn.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh2.ru-central1.internal
  success Yarn.main: Check that java tmp dir (/var/tmp) not mount as noexec @ ees3-adh3.ru-central1.internal
  success YARN.main: Enable HA ResourceManager @ ees3-adh2.ru-central1.internal
```

Job details

Step 3. Check the cluster state after the installation

As a result of successful installation, the cluster and its services change their state according to the following rules:

- The cluster state, displayed in the ***State*** field on the ***Clusters*** page, is being changed from `created` to `installed`.

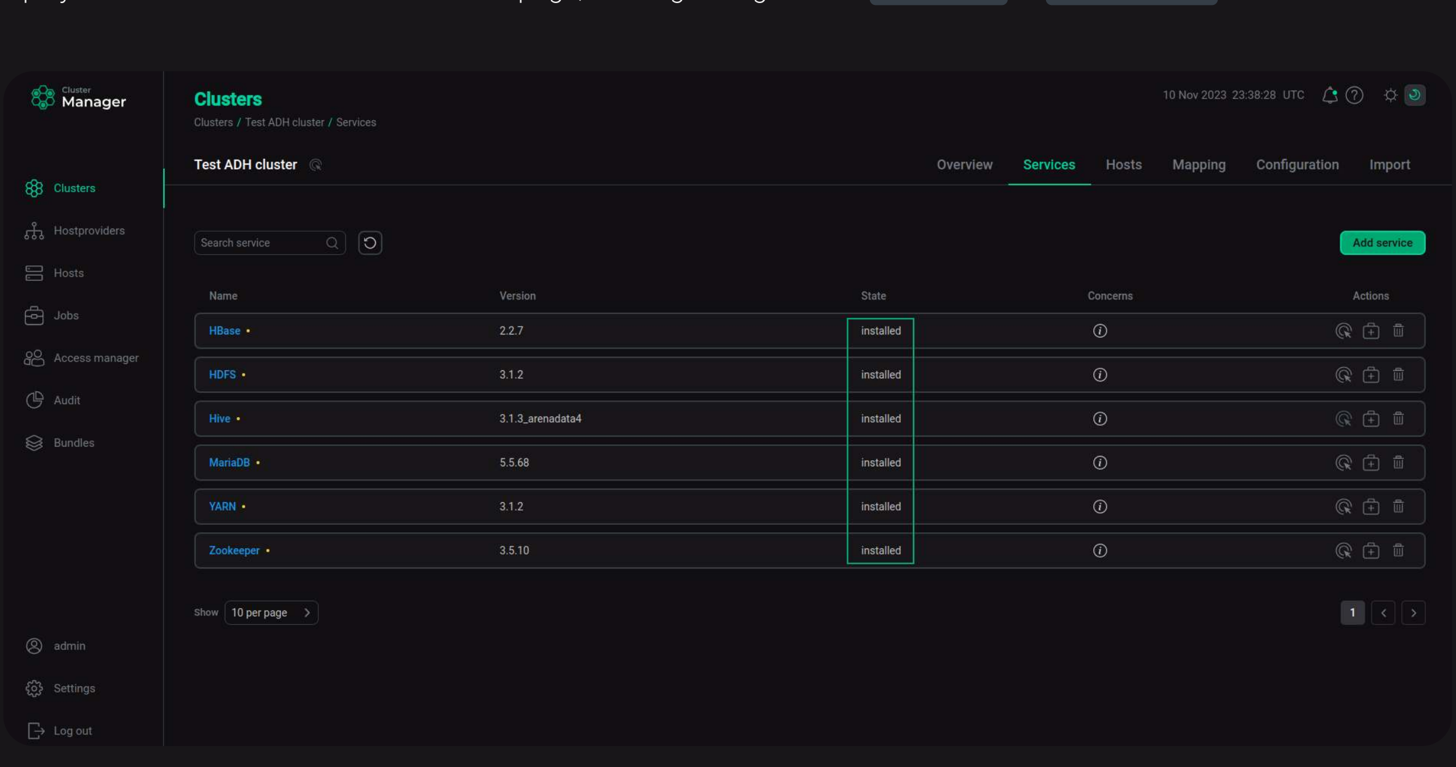


Cluster Manager
Clusters
Clusters
10 Nov 2023 23:29:12 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			

The cluster state after successful installation

- The state of cluster services, displayed on the ***Services*** tab of the cluster page, is being changed from `created` to `installed`.



Cluster Manager
Clusters
Clusters / Test ADH cluster / Services
10 Nov 2023 23:38:28 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Show 10 per page

Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

The state of cluster services after successful installation

Install monitoring

Daria Barysheva

To Table of Contents

Monitoring is an optional but strongly recommended ADH component that you can configure in one of the following ways:

- add the Monitoring service to the ADH cluster (available starting with ADH 4.0.0);
- add monitoring services to the Enterprise Tools cluster and integrate it with the ADH cluster (deprecated).

Monitoring service

Elena Dvoryadkina, Elena Kostyuchenko

Contents

To Table of Contents

Overview
Step 1. Add the service
Step 2. Add components
Step 3. Configure the service
Step 4. Install the service
Step 5. View results
- Metrics in the Prometheus format
- Prometheus web interface
- Grafana web interface

The Monitoring service deploys the Prometheus server inside ADH to collect and store cluster monitoring metrics, and also supports the ability to use the Grafana web application for visualization and analysis of information. This article describes steps required to add this service to an ADH cluster.

Overview

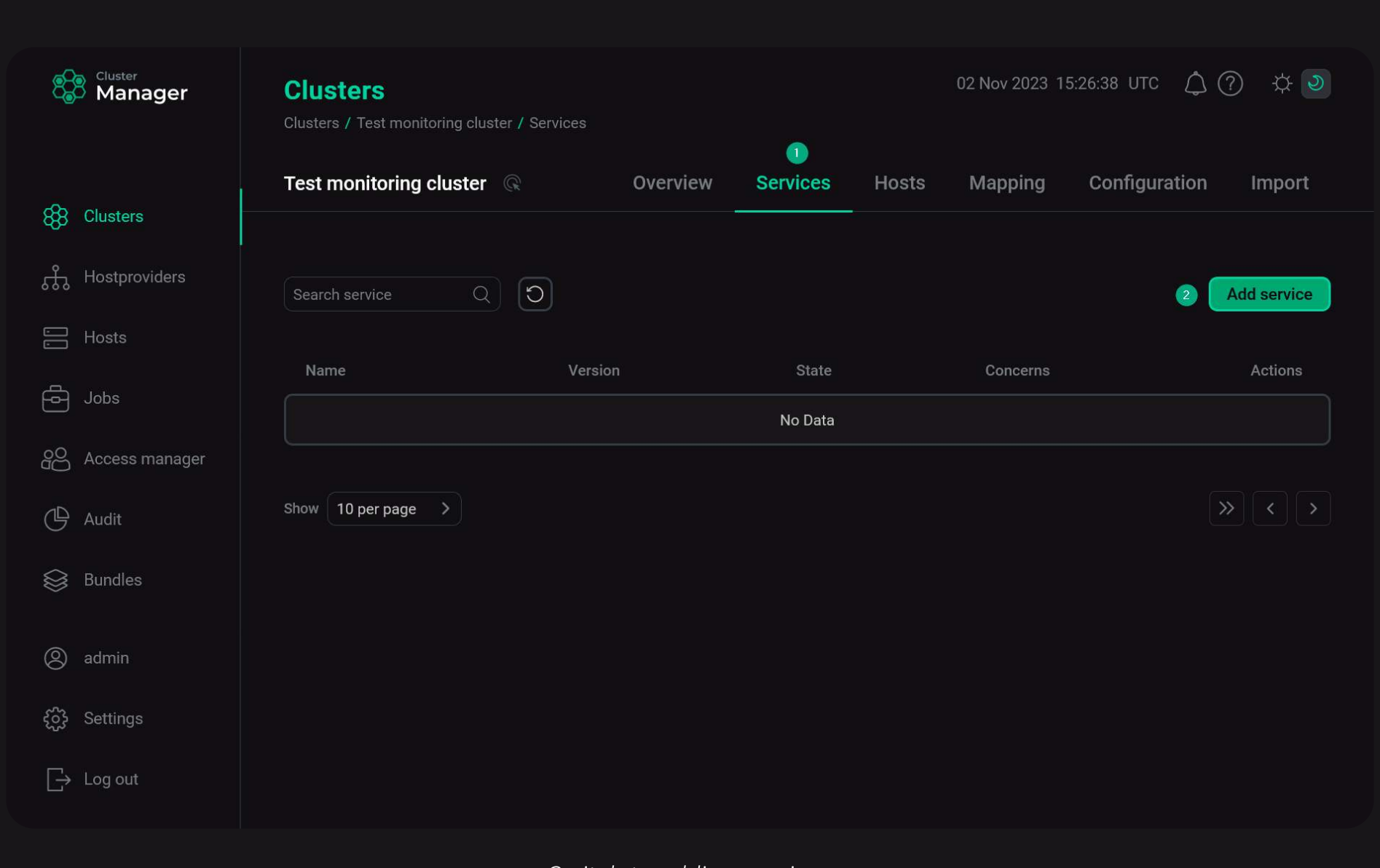
When adding the Monitoring service to ADH, you can install the Node Exporter on all or selected hosts of your cluster. It is a monitoring agent that reads host system metrics that Prometheus collects. Prometheus can also collect metrics from ADH services. These metrics will be available in the Prometheus format on the ports and endpoints specified in the service configurations. You can also use the Prometheus and Grafana web interfaces to view and analyze data that the monitoring service collects.

NOTE

- If you already have a Prometheus-compatible monitoring system set up (for example, your own Prometheus server or VictoriaMetrics), you can use it to collect ADH metrics. To do this, configure access to ADH metrics in your monitoring system using ADH's Prometheus parameters specified on the Monitoring service page.
- You can also use the Federation mechanism to migrate all metrics from the Prometheus server deployed in ADH to your Prometheus.

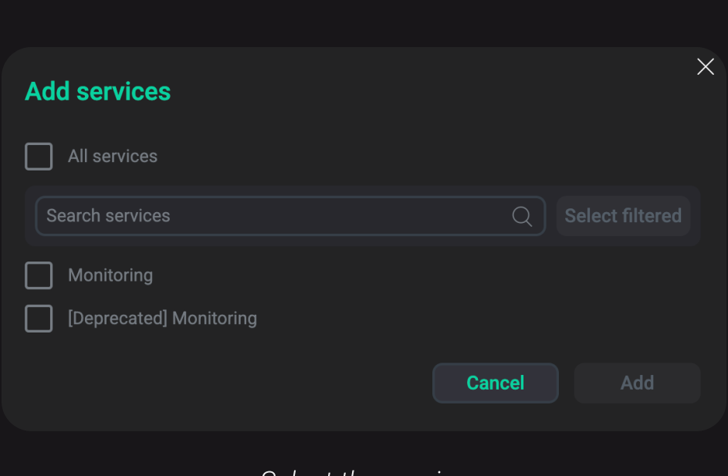
Step 1. Add the service

1. In the ADCM interface, open the ***Clusters*** page and click your ADH cluster name. On the cluster page that opens, switch to the ***Services*** tab and click ***Add services***.


Cluster Manager
Clusters
02 Nov 2023 15:26:38 UTC
Clusters / Test monitoring cluster / Services
Test monitoring cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
2
Search service
Add service
Name
Version
State
Concerns
Actions
No Data
Show
10 per page
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Switch to adding services

2. In the opened dialog, select the ***Monitoring*** service and click ***Add***.



Add services

☐ All services

Search services | Select filtered

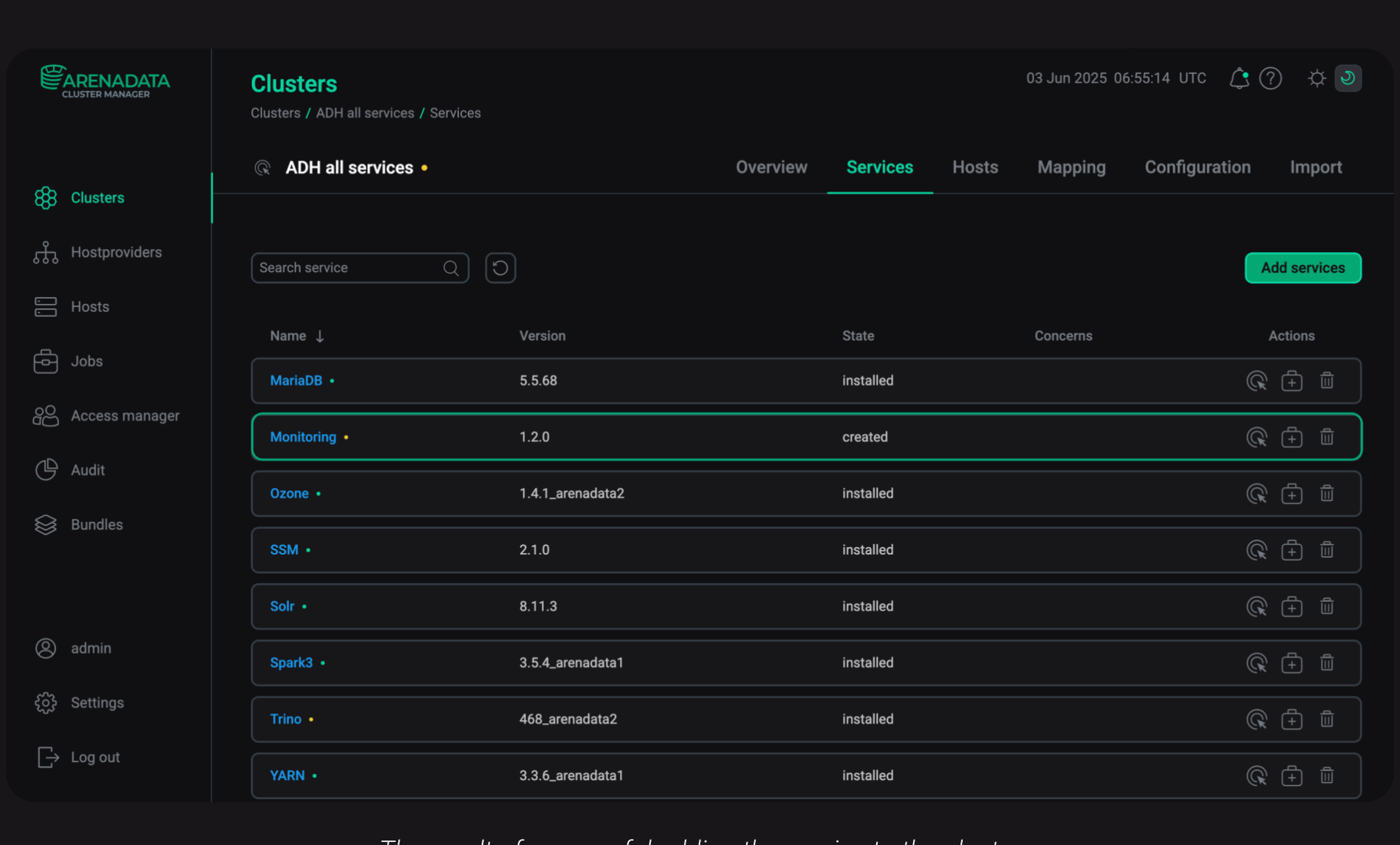
☐ Monitoring

☐ [Deprecated] Monitoring

Cancel | Add

Select the service

As a result, the added service is displayed on the ***Services*** tab.

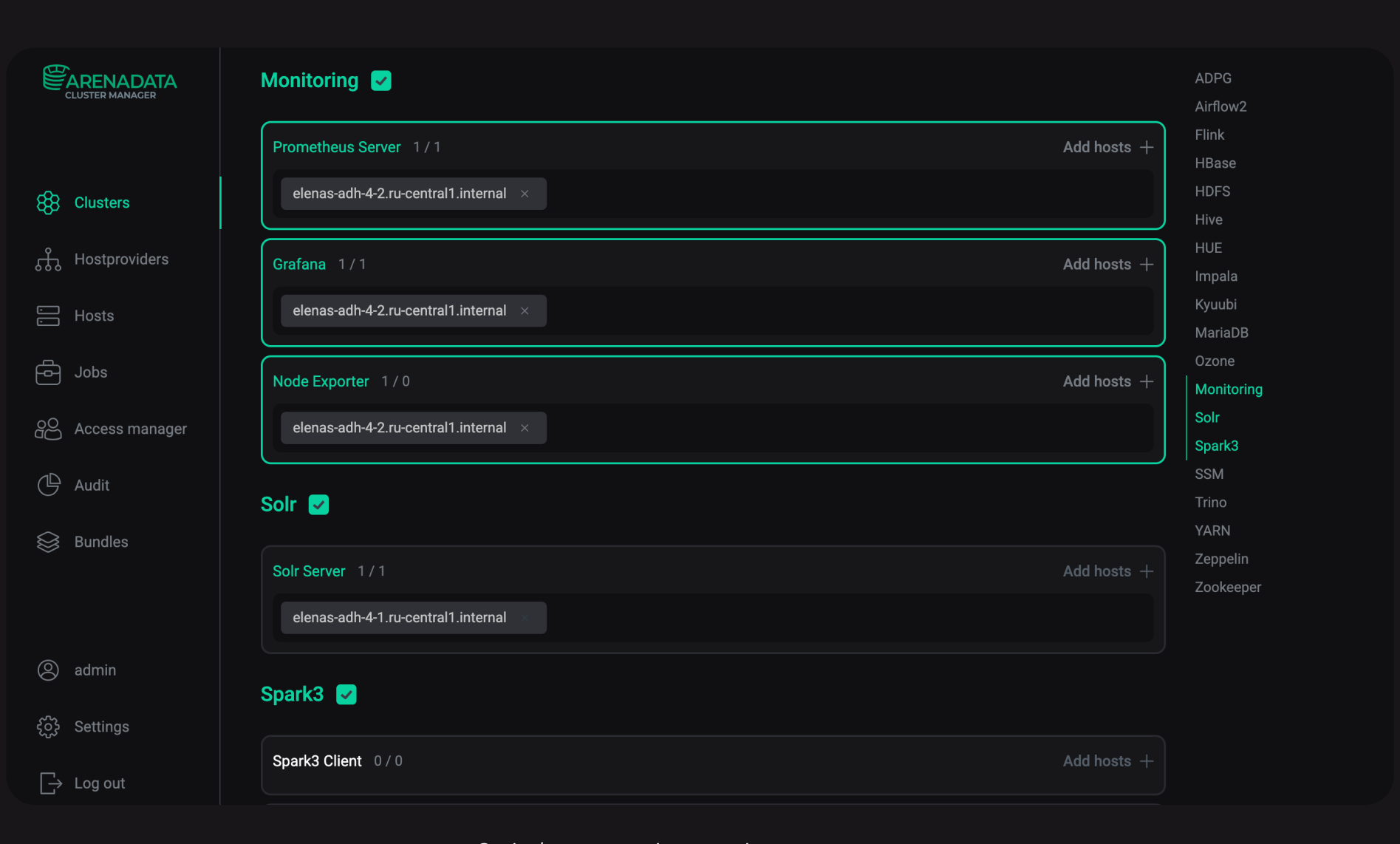

ARENADATA CLUSTER MANAGER
Clusters
03 Jun 2025 06:55:14 UTC
Clusters / ADH all services / Services
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	created		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

The result of successful adding the service to the cluster

Step 2. Add components

1. On the cluster page, open the ***Mapping*** tab to proceed to mapping service components to cluster hosts.


ARENADATA CLUSTER MANAGER
Add hosts
ADPG
Airflow2
Flink
HBase
HDFS
Hive
HUE
Impala
Kyuubi
MariaDB
Ozone
Monitoring
Solr
Spark3
SSM
Trino
YARN
Zeppelin
Zookeeper
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Monitoring ☑

Prometheus Server 1 / 1: elenas-adh-4-2.ru-central1.internal

Grafana 1 / 1: elenas-adh-4-2.ru-central1.internal

Node Exporter 1 / 0: elenas-adh-4-2.ru-central1.internal

Solr ☑

Solr Server 1 / 1: elenas-adh-4-1.ru-central1.internal

Spark3 ☑

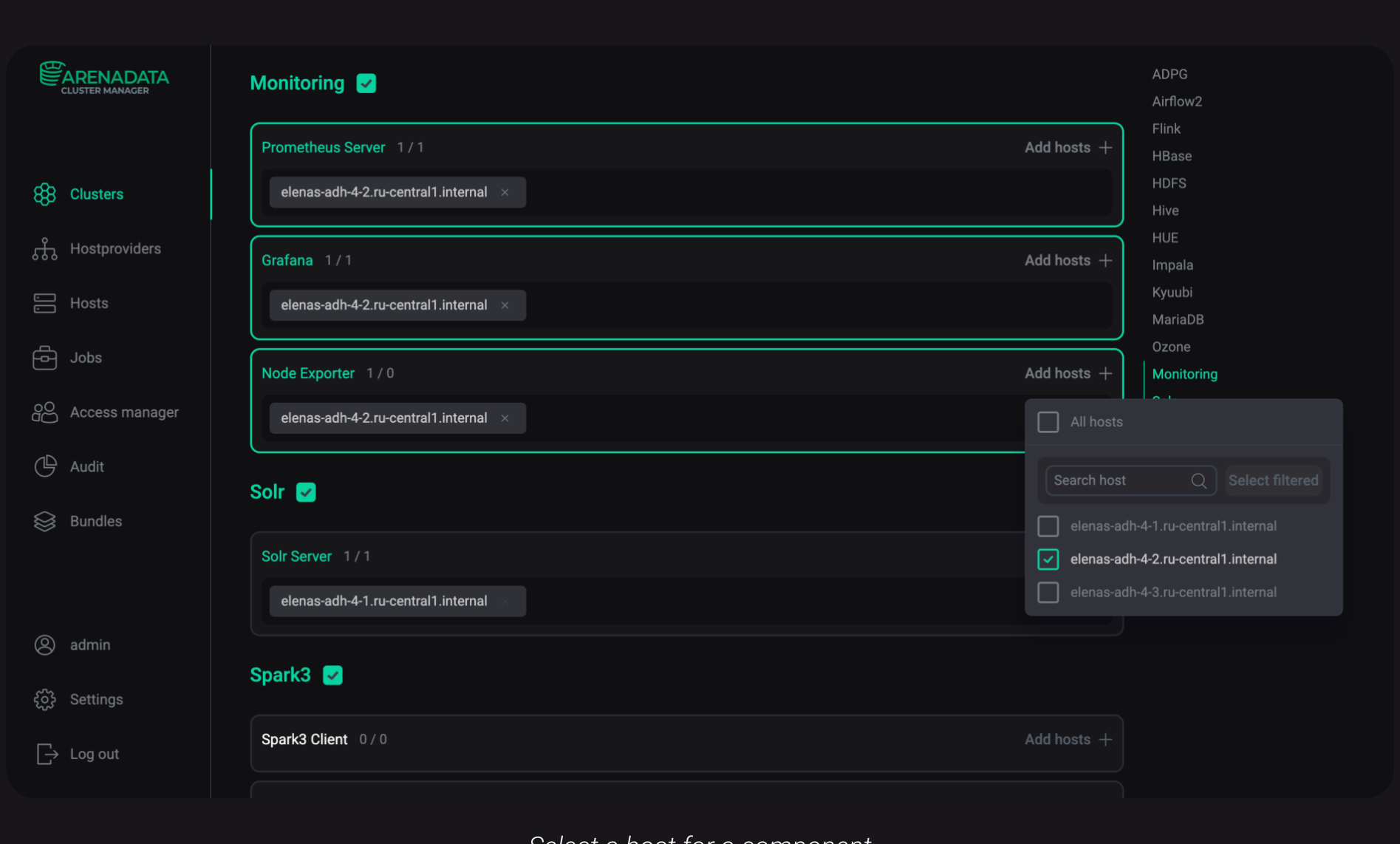
Spark3 Client 0 / 0

Switch to mapping service components

Monitoring service components

Component	Description
Prometheus Server	Stores system metrics generated by an ADH cluster and allows Grafana to request them
Grafana	Visualizes ADH metrics as graphs and charts organized into dashboards
Node Exporter	Collects system metrics and sends them to Prometheus

2. Assign hosts to components of the Monitoring service — click ***Add hosts*** and select the desired hosts in the pop-up window.


ARENADATA CLUSTER MANAGER
Add hosts
ADPG
Airflow2
Flink
HBase
HDFS
Hive
HUE
Impala
Kyuubi
MariaDB
Ozone
Monitoring
Search host
Select filtered
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

☐ All hosts

☐ elenas-adh-4-1.ru-central1.internal

☑ elenas-adh-4-2.ru-central1.internal

☐ elenas-adh-4-3.ru-central1.internal

Select a host for a component

CAUTION

It is not recommended to install the Prometheus Server on hosts with ADH components — use separate hosts for it. Otherwise, if an ADH host fails and/or the load on it is critically high, information about the corresponding problems will not be saved.

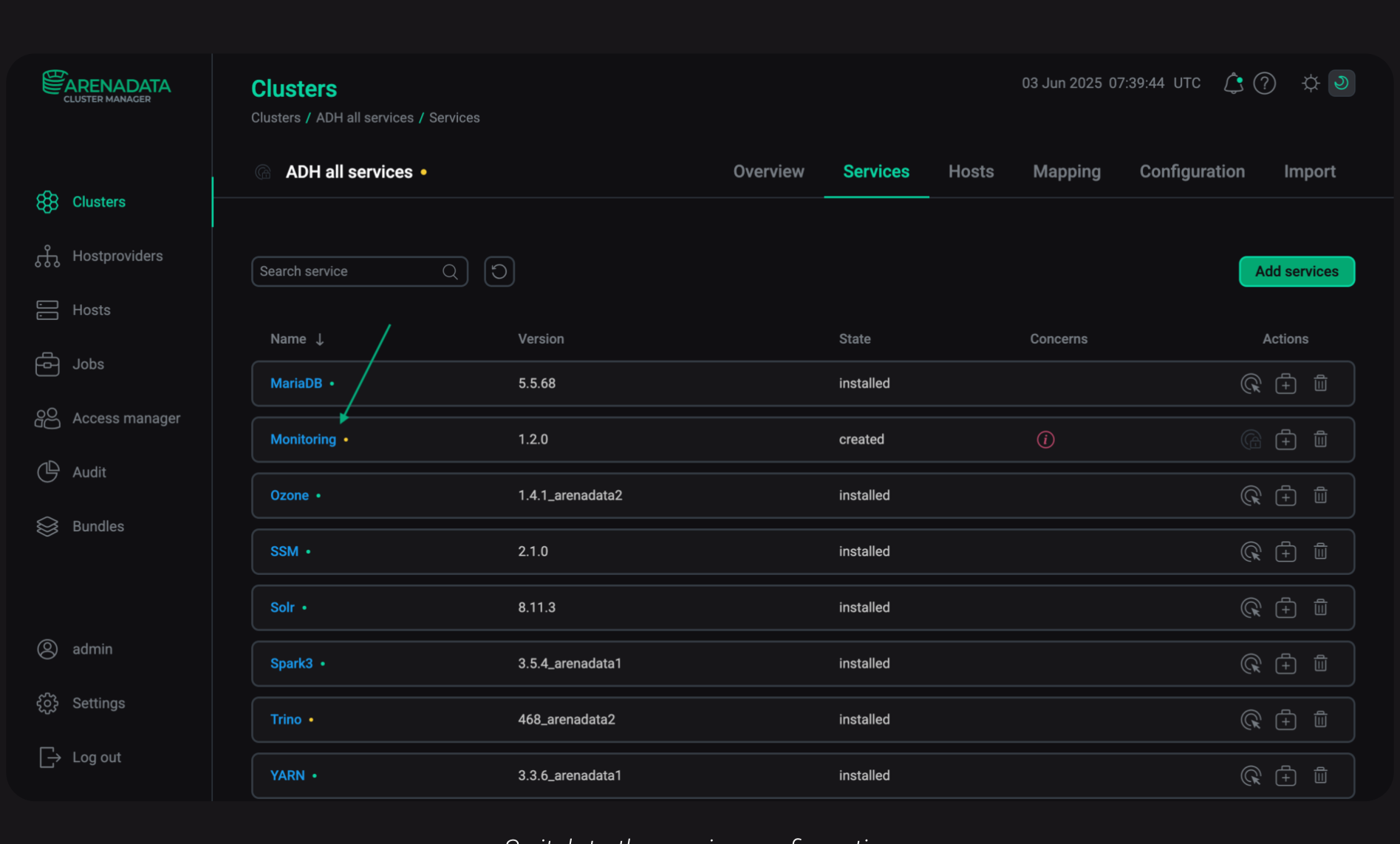
3. After the distribution of components is completed, click ***Save***.

NOTE

When changing the topology of services/configurations related to monitoring, it is necessary to restart the Prometheus component in the Monitoring service.

Step 3. Configure the service

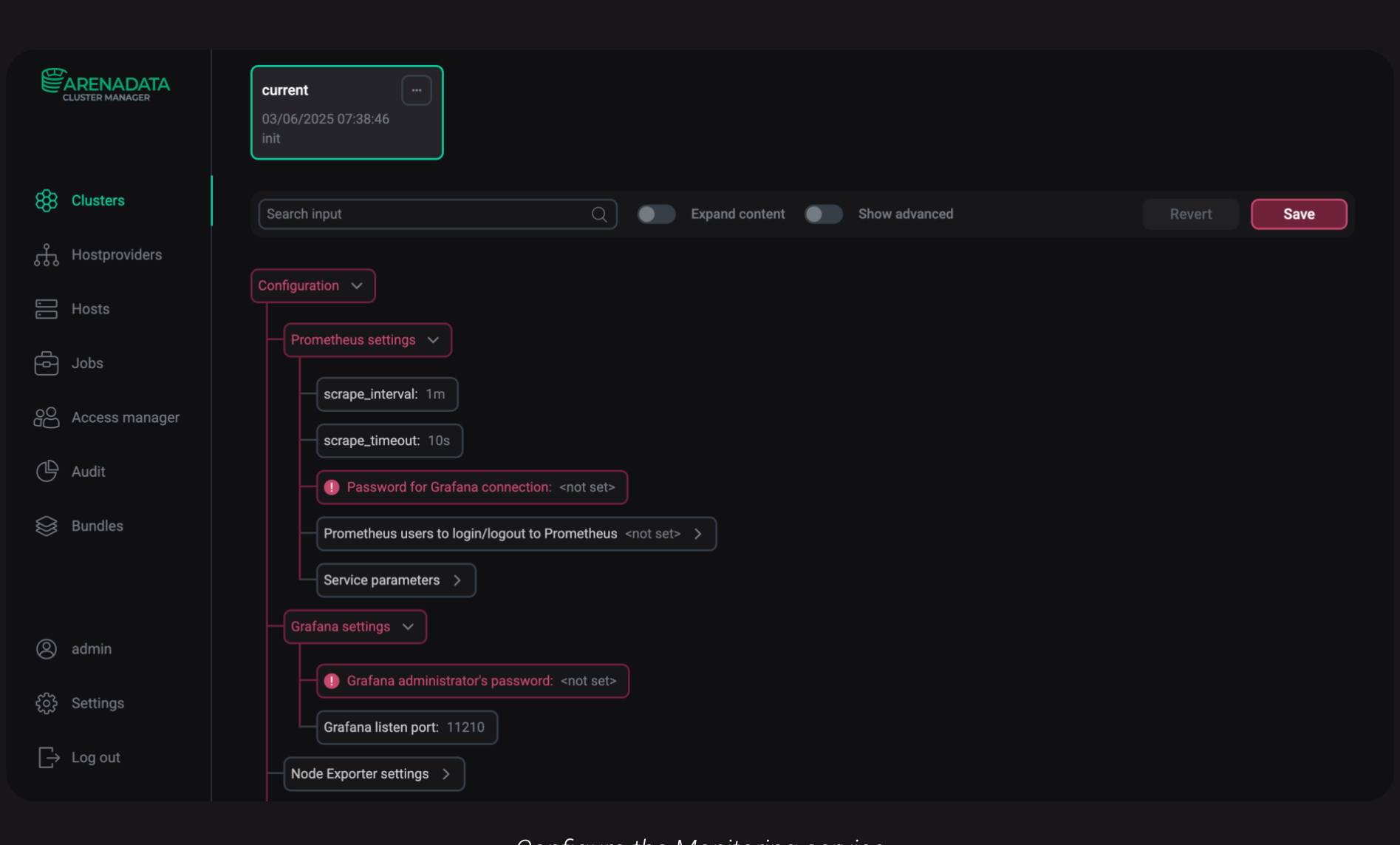
1. Open the ***Services*** tab on the cluster page and click the ***Monitoring*** service name in the ***Name*** column.


ARENADATA CLUSTER MANAGER
Clusters
03 Jun 2025 07:39:44 UTC
Clusters / ADH all services / Services
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	created		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

Switch to the service configuration

2. On the page that opens, fill in the service's configuration parameters — see the Monitoring section for parameter descriptions. Fields highlighted in red are required.


ARENADATA CLUSTER MANAGER
Search input
Expand content
Show advanced
Revert
Save
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

current
03/06/2025 07:38:46
init

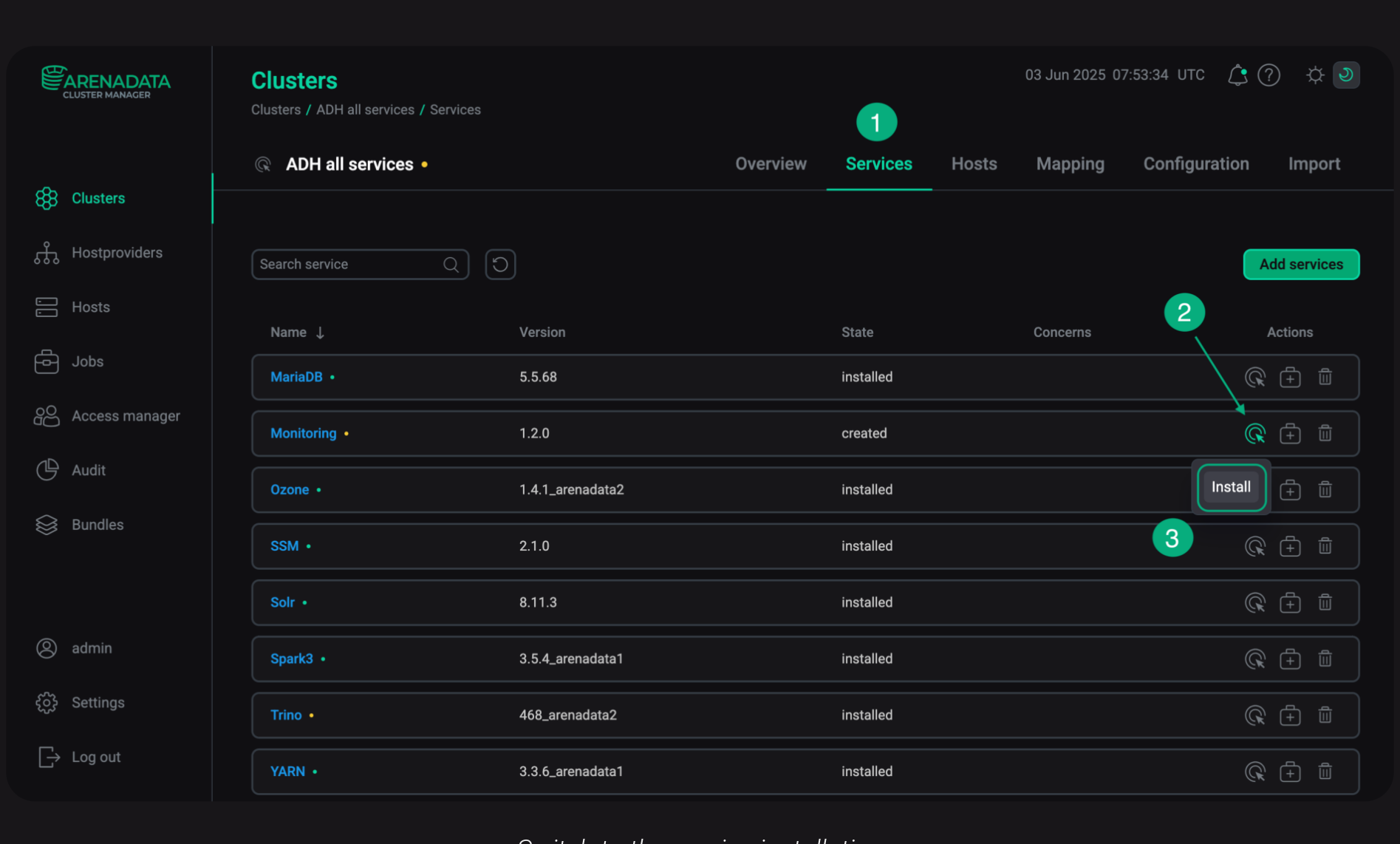
Configuration
- Prometheus settings
  - scrape_interval: 1m
  - scrape_timeout: 10s
  - Password for Grafana connection: <not set>
  - Prometheus users to login/logout to Prometheus <not set>
  - Service parameters
- Grafana settings
  - Grafana administrator's password: <not set>
  - Grafana listen port: 11210
- Node Exporter settings

Configure the Monitoring service

After specifying all necessary parameters, click ***Save***.

Step 4. Install the service

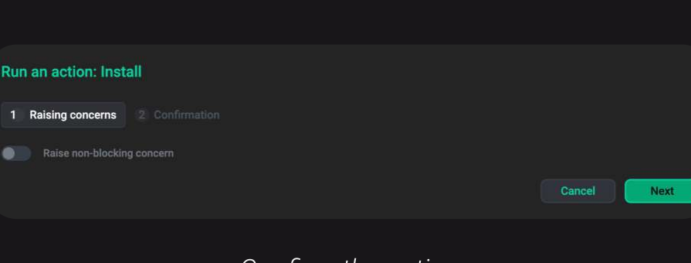
1. On the ***Services*** tab, click the icon for the Monitoring service in the ***Actions*** column and run the ***Install*** action.


ARENADATA CLUSTER MANAGER
Clusters
03 Jun 2025 07:53:34 UTC
Clusters / ADH all services / Services
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
1
2
3
Install
Search service
Add services
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	created		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

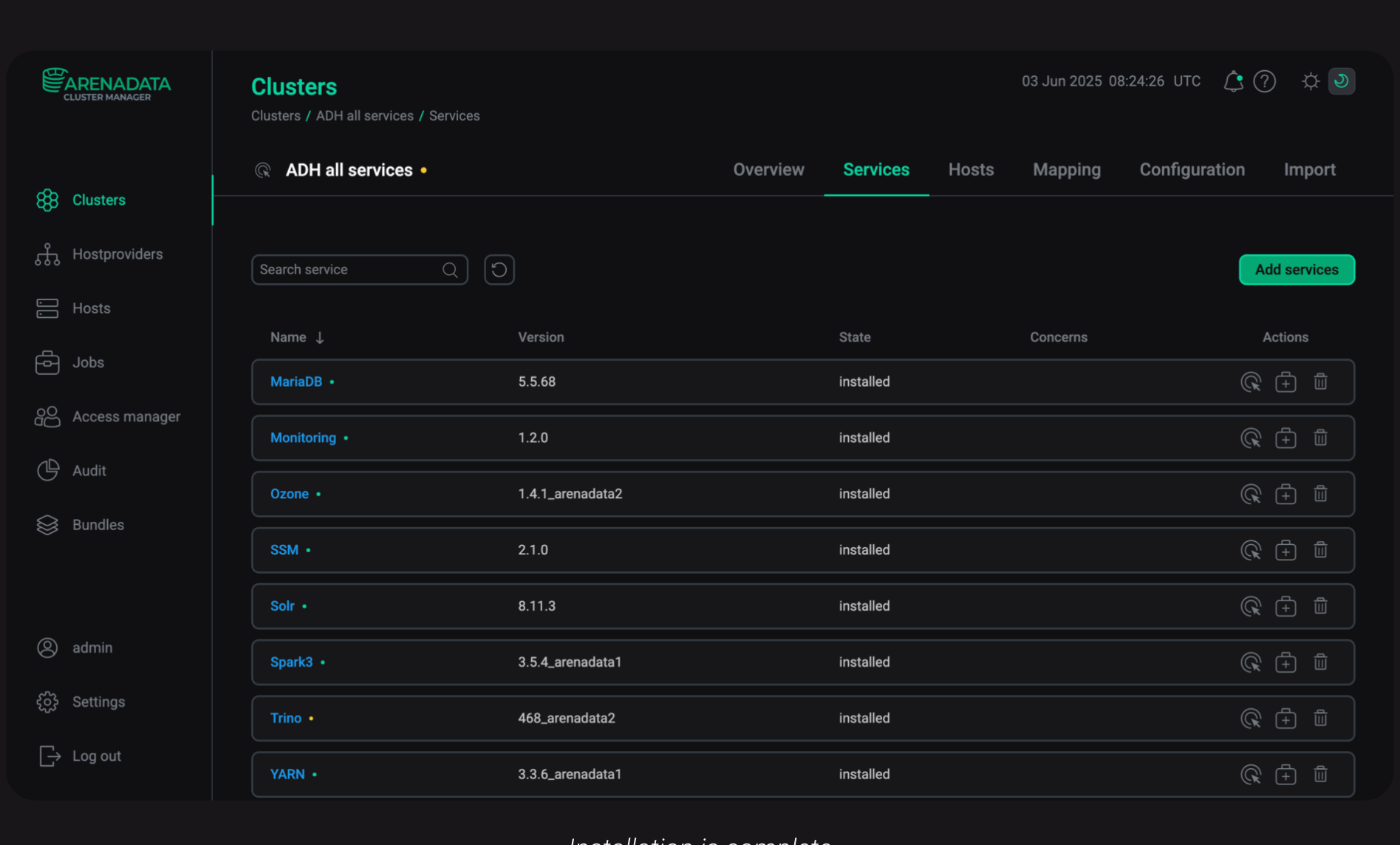
Switch to the service installation

2. Confirm the action in the opened window.


Run an action: Install
1 Raising concerns
2 Confirmation
Raise non-blocking concern
Cancel
Next

Confirm the action

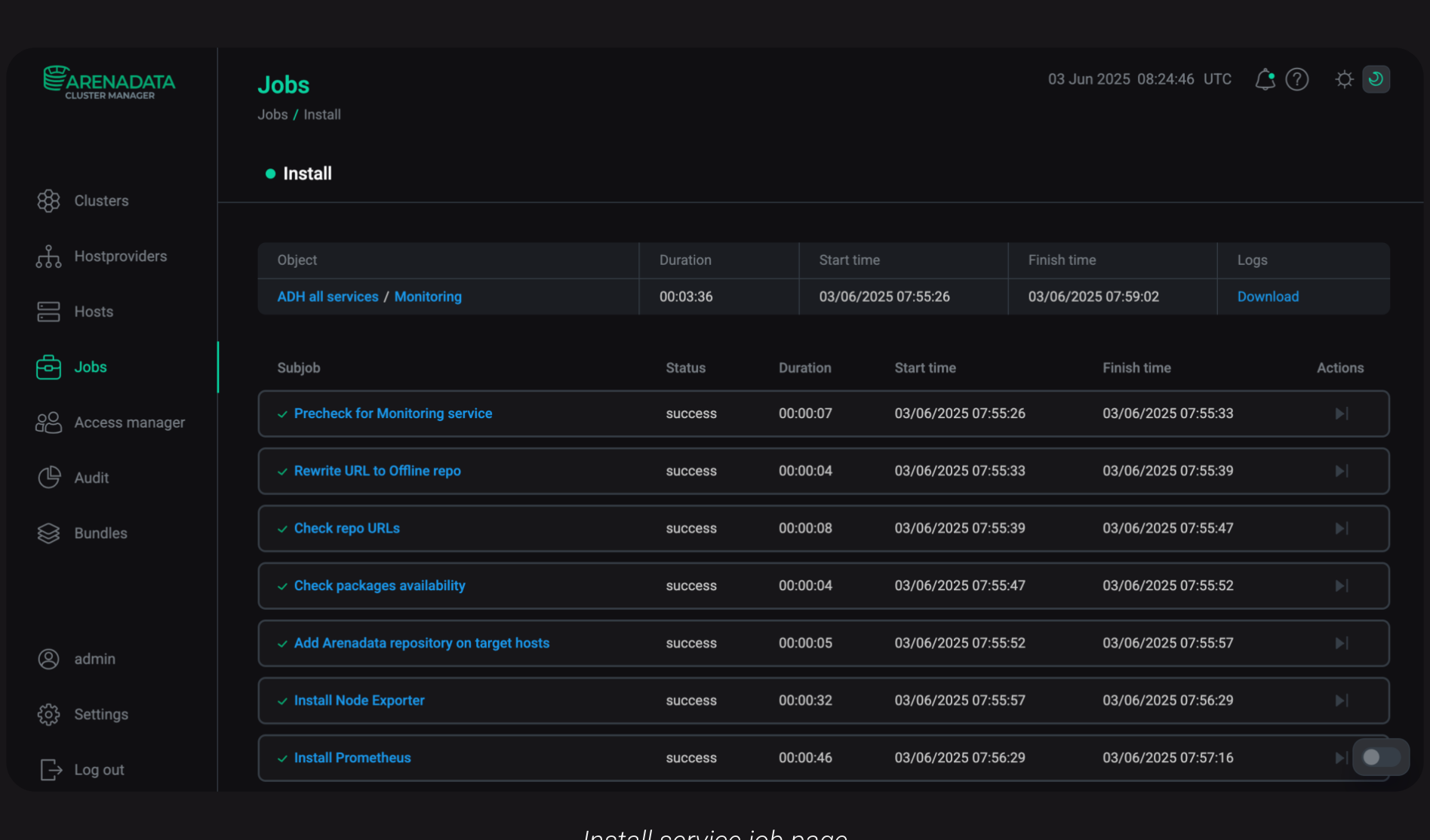
3. Wait until the installation is completed. Then check that the service state has changed from `created` to `installed`.


ARENADATA CLUSTER MANAGER
Clusters
03 Jun 2025 08:24:26 UTC
Clusters / ADH all services / Services
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Name	Version	State	Concerns	Actions
MariaDB	5.5.68	installed		
Monitoring	1.2.0	installed		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Solr	8.11.3	installed		
Spark3	3.5.4_arenadata1	installed		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

Installation is complete

To view the service installation process and analyze errors if they occur, select ***Jobs*** in the left navigation menu and click the ***Install*** job name in the ***Jobs*** list.


ARENADATA CLUSTER MANAGER
Jobs
03 Jun 2025 08:24:46 UTC
Jobs / Install
Install
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Object	Duration	Start time	Finish time	Logs
ADH all services / Monitoring	00:03:36	03/06/2025 07:55:26	03/06/2025 07:59:02	Download

Subjob	Status	Duration	Start time	Finish time	Actions
Precheck for Monitoring service	success	00:00:07	03/06/2025 07:55:26	03/06/2025 07:55:33	
Rewrite URL to Offline repo	success	00:00:04	03/06/2025 07:55:33	03/06/2025 07:55:39	
Check repo URLs	success	00:00:08	03/06/2025 07:55:39	03/06/2025 07:55:47	
Check packages availability	success	00:00:04	03/06/2025 07:55:47	03/06/2025 07:55:52	
Add Arenadata repository on target hosts	success	00:00:05	03/06/2025 07:55:52	03/06/2025 07:55:57	
Install Node Exporter	success	00:00:32	03/06/2025 07:55:57	03/06/2025 07:56:29	
Install Prometheus	success	00:00:46	03/06/2025 07:56:29	03/06/2025 07:57:16	

Install service job page

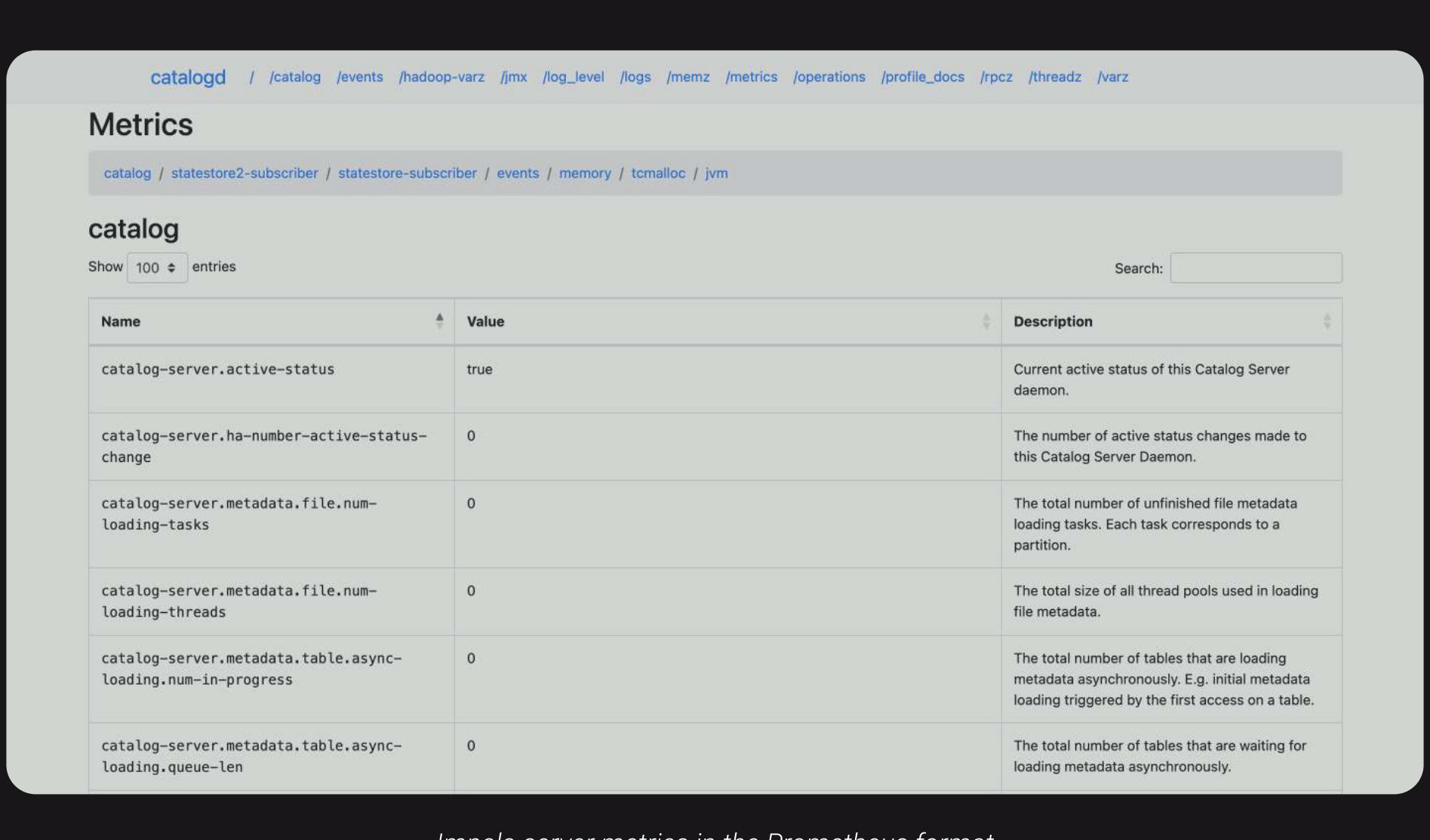
Step 5. View results

The Monitoring service starts automatically after the installation. To ensure that monitoring works correctly, check that both system and ADH service metrics are collected from the cluster hosts. To do this, view metrics in the Prometheus format in the browser or use the Prometheus or Grafana web interface.

Metrics in the Prometheus format

1. In the address bar of your browser, enter an address of an ADH host with the specified port and endpoint to listen for service or system metrics. Port numbers and endpoints are defined by the parameters located in:
   - the configuration pages of the ADH services — settings for accessing monitoring metrics of these services;
   - the ***Node Exporter settings*** section on the configuration page of the Monitoring service — settings for accessing system metrics of the ADH cluster hosts.

   For example, *http://10.92.42.57:25020/metrics* is an address to view metrics of the Impala server on a host with IP `10.92.42.57`.
2. The page that opens will display monitoring metrics from the specified host of the ADH cluster in the Prometheus format.


catalogd
/
/catalog
/events
/hadoop-varz
/jmx
/log_level
/logs
/memz
/metrics
/operations
/profile_docs
/rpcz
/threadz
/varz
catalog / statestore2-subscriber / statestore-subscriber / events / memory / tcmalloc / jvm
Show 100 entries
Search:

Metrics

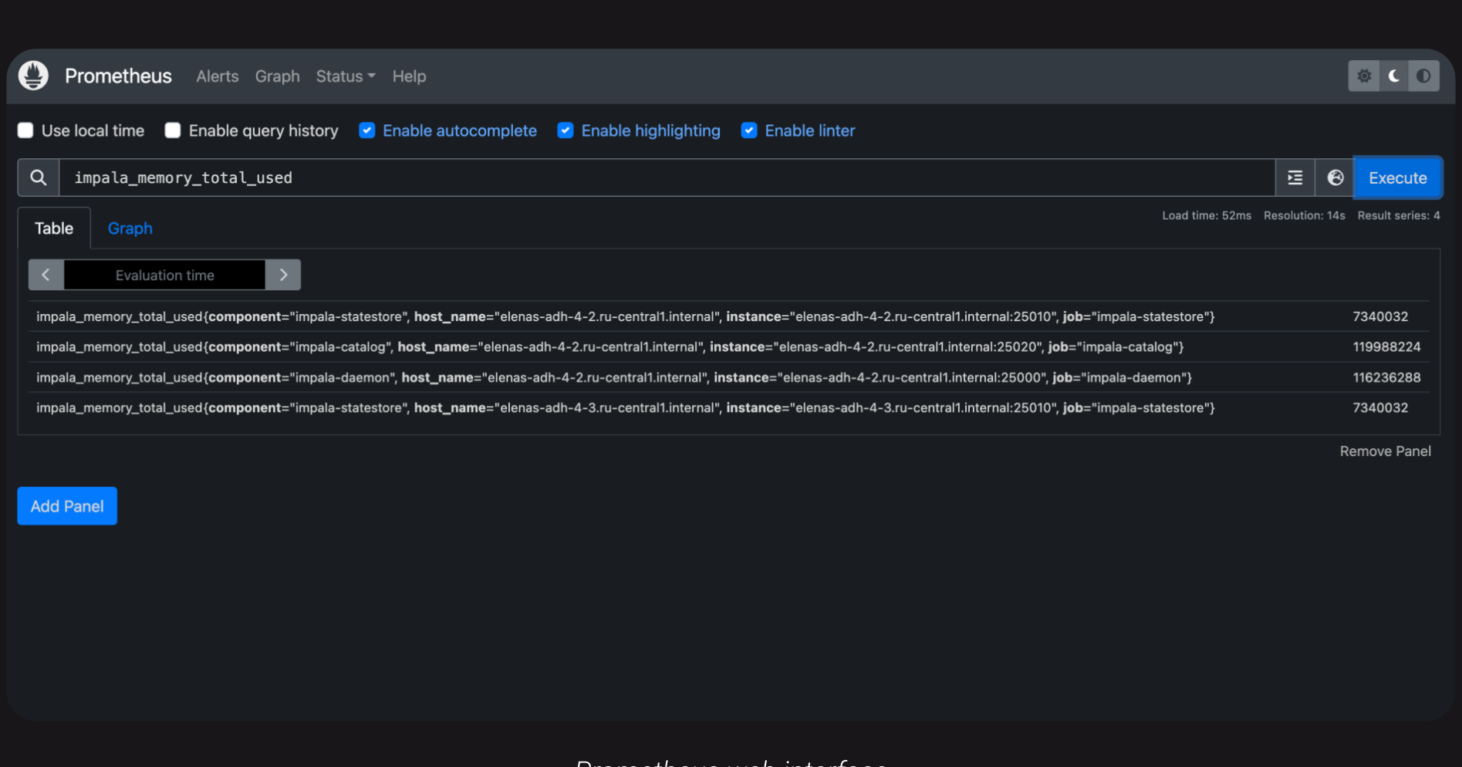
catalog

Name	Value	Description
catalog-server.active-status	true	Current active status of this Catalog Server daemon.
catalog-server.ha-number-active-status-change	0	The number of active status changes made to this Catalog Server Daemon.
catalog-server.metadata.file.num-loading-tasks	0	The total number of unfinished file metadata loading tasks. Each task corresponds to a partition.
catalog-server.metadata.file.num-loading-threads	0	The total size of all thread pools used in loading file metadata.
catalog-server.metadata.table.async-loading.num-in-progress	0	The total number of tables that are loading metadata asynchronously. E.g. initial metadata loading triggered by the first access on a table.
catalog-server.metadata.table.async-loading.queue-len	0	The total number of tables that are waiting for loading metadata asynchronously.

Impala server metrics in the Prometheus format

Prometheus web interface

1. In the address bar of your browser, enter an IP address of a host where the Prometheus Server component is installed. Specify also a port number from the ***web.listen-address*** parameter in the ***Service parameters*** group within the ***Prometheus settings*** section of the Monitoring service configuration (the default port is `11200`). For example: *http://10.92.40.107:11200*. To log into the Prometheus interface, use a username and password that are also specified in the ***Prometheus settings*** section — the ***Prometheus users to login/logout to Prometheus*** setting.
2. In the ***Expression*** field, you can enter a metric and click ***Execute*** — values of this metric on all hosts of the ADH cluster will be shown in the interface.


Prometheus
Alerts
Graph
Status
Help
Execute
Load time: 52ms
Resolution: 14s
Result series: 4
Table
Graph
Evaluation time
Remove Panel
Add Panel

☐ Use local time ☐ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

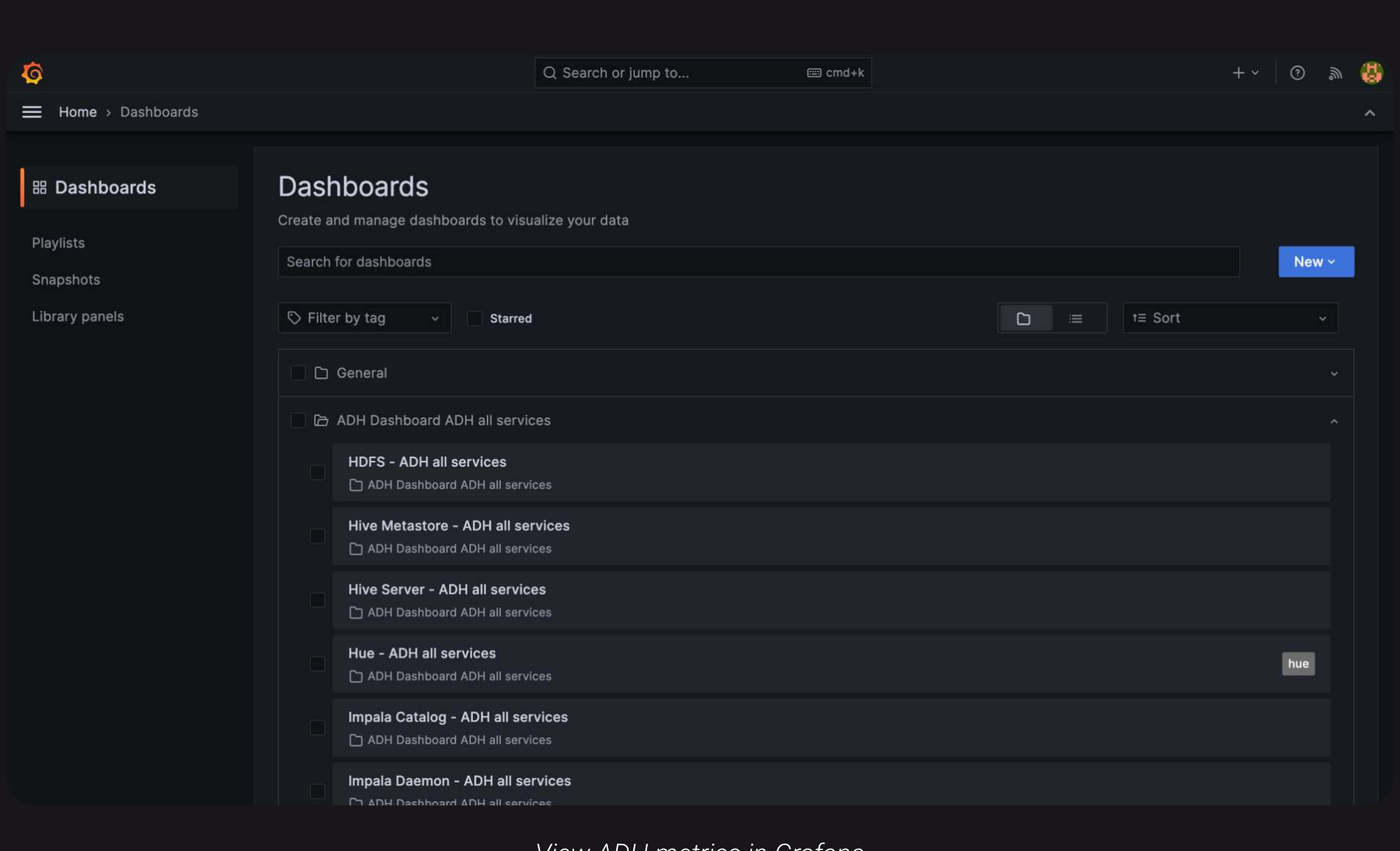
```
impala_memory_total_used
```

```
impala_memory_total_used{component="impala-statestore", host_name="elenas-adh-4-2.ru-central1.internal", instance="elenas-adh-4-2.ru-central1.internal:25010", job="impala-statestore"}   7340032
impala_memory_total_used{component="impala-catalog", host_name="elenas-adh-4-2.ru-central1.internal", instance="elenas-adh-4-2.ru-central1.internal:25020", job="impala-catalog"}   119988224
impala_memory_total_used{component="impala-daemon", host_name="elenas-adh-4-2.ru-central1.internal", instance="elenas-adh-4-2.ru-central1.internal:25000", job="impala-daemon"}   116236288
impala_memory_total_used{component="impala-statestore", host_name="elenas-adh-4-3.ru-central1.internal", instance="elenas-adh-4-3.ru-central1.internal:25010", job="impala-statestore"}   7340032
```

Prometheus web interface

Grafana web interface

1. In the address bar of your browser, enter an address of a host on which Grafana is deployed and add a port number — a value of the ***Grafana listen port*** parameter located in the ***Grafana settings*** section of the Monitoring service configuration (the default value is `11210`). For example, *http://10.92.40.107:11210*. To log in, use `admin` as a username, and the ***Grafana administrator's password*** parameter value (also found in the ***Grafana settings*** section of service configuration parameters) as a password.
2. In the window that opens, navigate to ***Home → Dashboards*** and expand the ***ADH Dashboard <ADH_cluster_name>*** section. In this section, you can select one of available dashboards to view service or system metrics coming from your ADH cluster.


Search or jump to...
cmd+k
Home › Dashboards
Dashboards
Playlists
Snapshots
Library panels
Dashboards
Create and manage dashboards to visualize your data
Search for dashboards
New
Filter by tag
Starred
Sort
General
ADH Dashboard ADH all services
HDFS - ADH all services
Hive Metastore - ADH all services
Hive Server - ADH all services
Hue - ADH all services
hue
Impala Catalog - ADH all services
Impala Daemon - ADH all services

View ADH metrics in Grafana

Monitoring cluster

Elena Dvoryadkina, Elena Kostyuchenko

To Table of Contents

To set up monitoring for ADH during **offline** installation, you can add monitoring services (Diamond, Graphite, and Grafana) to the Enterprise Tools cluster (there is no need for a separate monitoring cluster, as the Enterprise Tools cluster plays its role) and integrate it with the ADH cluster. The steps for installing monitoring in this case are listed below. You can configure the Enterprise Tools cluster both before and after the ADH cluster installation, but the Integrate with ADH cluster step can be executed only when the ADH cluster already exists.

- Configure Enterprise Tools cluster
- Integrate with ADH cluster

Configure Enterprise Tools cluster

Daria Barysheva

Contents

To Table of Contents

Step 1. Add monitoring services to the ET cluster
Step 2. Map monitoring components
Step 3. Configure monitoring services
Step 4. Install monitoring services
Step 5. Verify results
- Graphite
- Grafana

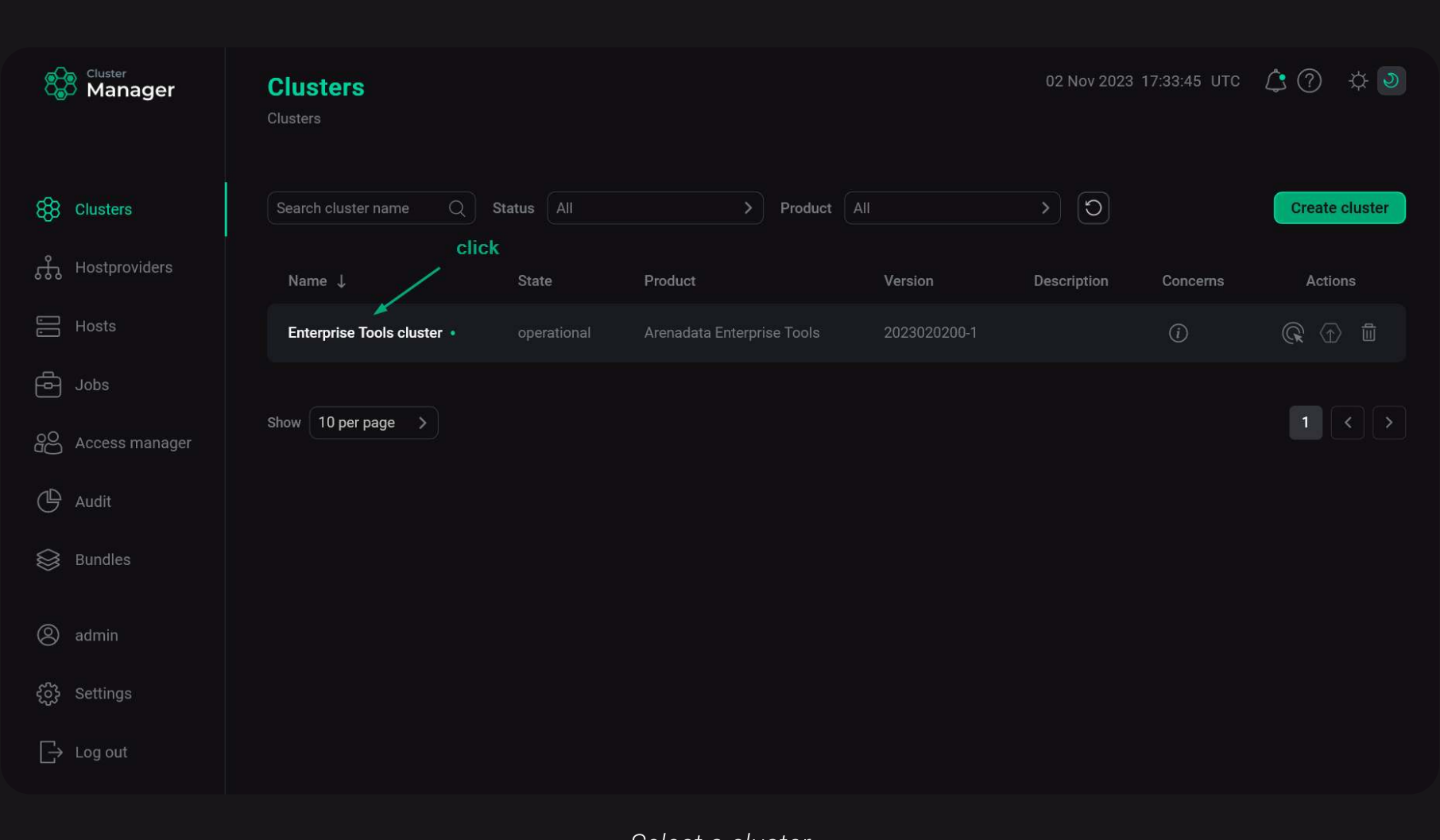
IMPORTANT

Before the subsequent monitoring configuration, make sure that the Enterprise Tools (ET) cluster is installed via ADCM. All the following actions are performed on its base.

Step 1. Add monitoring services to the ET cluster

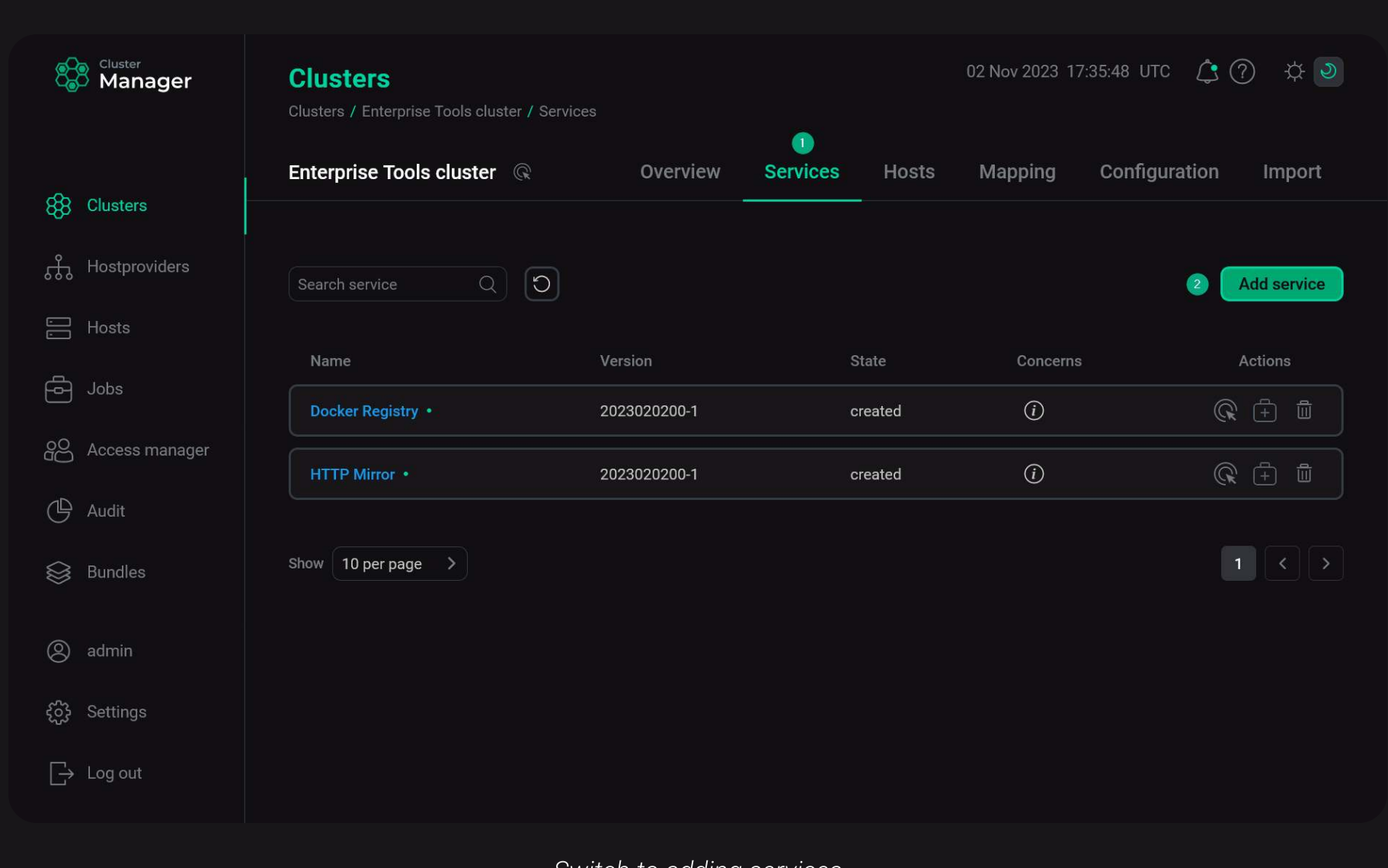
The steps for adding services to a cluster are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
02 Nov 2023 17:33:45 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
click
Name
State
Product
Version
Description
Concerns
Actions
Enterprise Tools cluster
operational
Arenadata Enterprise Tools
2023020200-1
Show 10 per page

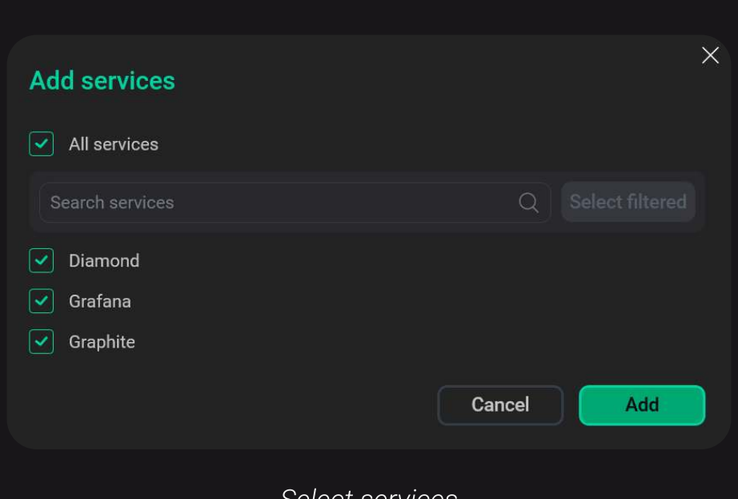
Select a cluster

2. Open the ***Services*** tab on the cluster page and click ***Add service***.


Clusters
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 17:35:48 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
2
Search service
Add service
Name
Version
State
Concerns
Actions
Docker Registry
2023020200-1
created
HTTP Mirror
2023020200-1
created
Show 10 per page

Switch to adding services

3. In the opened dialog, select services that should be added to the cluster and click ***Add***.


Add services
☑ All services
Search services
Select filtered
☑ Diamond
☑ Grafana
☑ Graphite
Cancel
Add

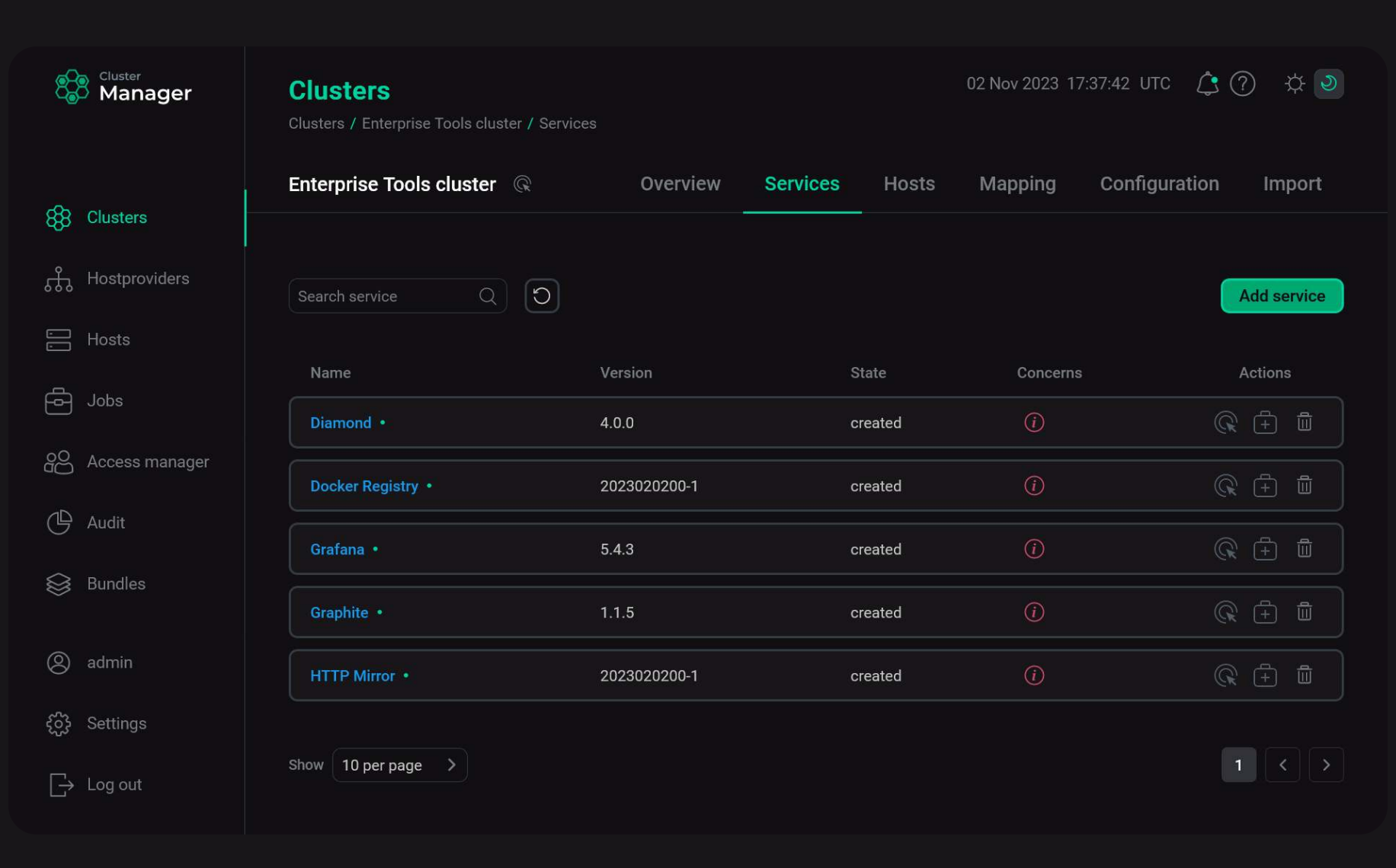
Select services

The brief description of available services is listed below.

Monitoring services that should be added to the Enterprise Tools cluster

Service	Purpose
Diamond	A Python service used for collecting system metrics from the hosts that are connected to the monitoring system. The collected metrics are processed by Carbon (the backend services of Graphite)
Graphite	Includes several components: ▪ **Carbon** — a set of services that are used to process metrics collected by Diamond. ▪ **Whisper DB** — a temporary file database where collected metrics are stored. ▪ **Web service** — allows to get and visualize metrics from Whisper DB.
Grafana	A web service that allows users to create convenient sets of graphs, alerts, and other visualization tools for metrics received from Graphite. By default, uses the port `3000`

4. As a result, the added services are displayed on the ***Services*** tab.

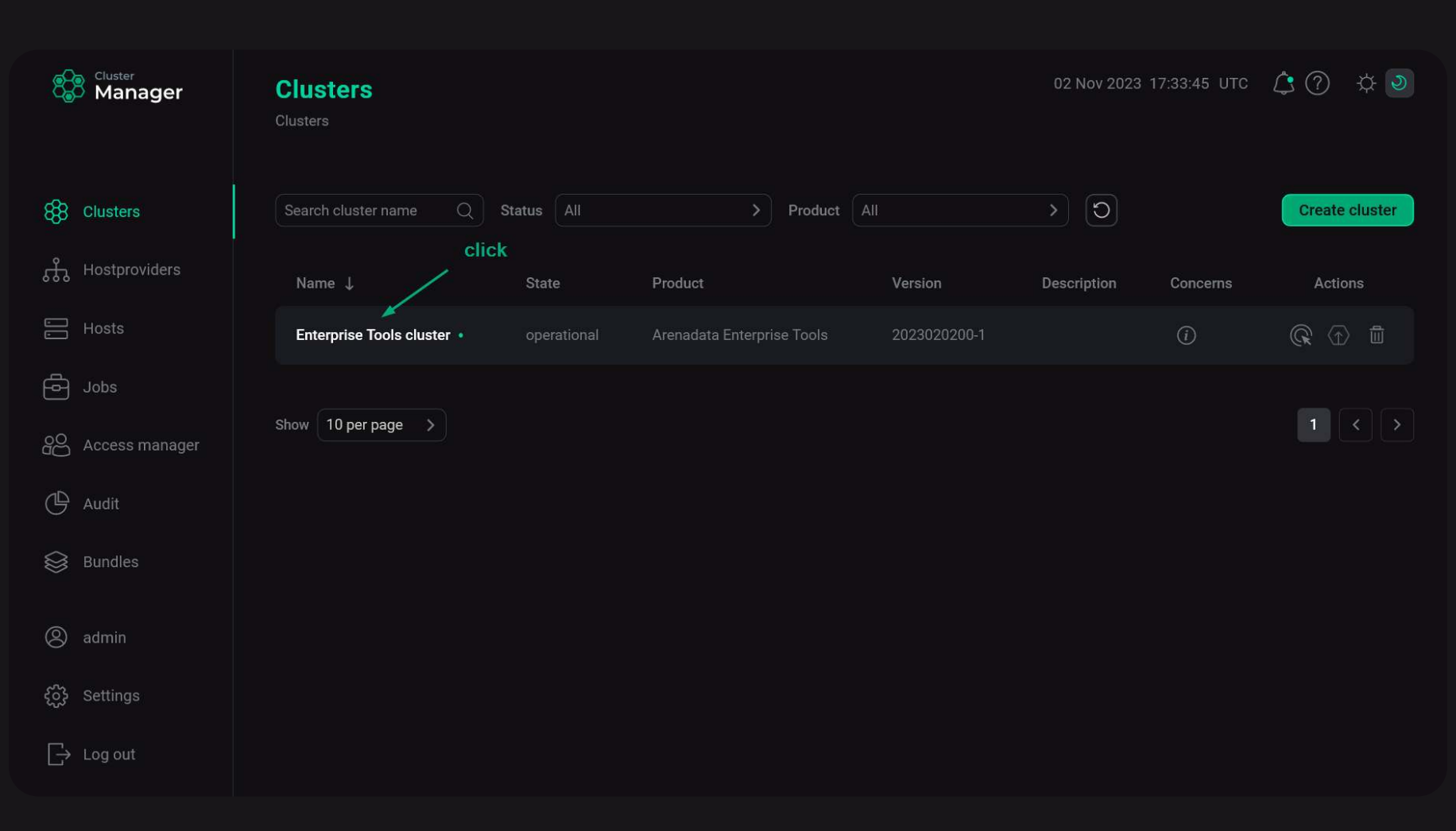

Clusters
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 17:37:42 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
Name
Version
State
Concerns
Actions
Diamond
4.0.0
created
Docker Registry
2023020200-1
created
Grafana
5.4.3
created
Graphite
1.1.5
created
HTTP Mirror
2023020200-1
created
Show 10 per page

The result of successful adding services to a cluster

Step 2. Map monitoring components

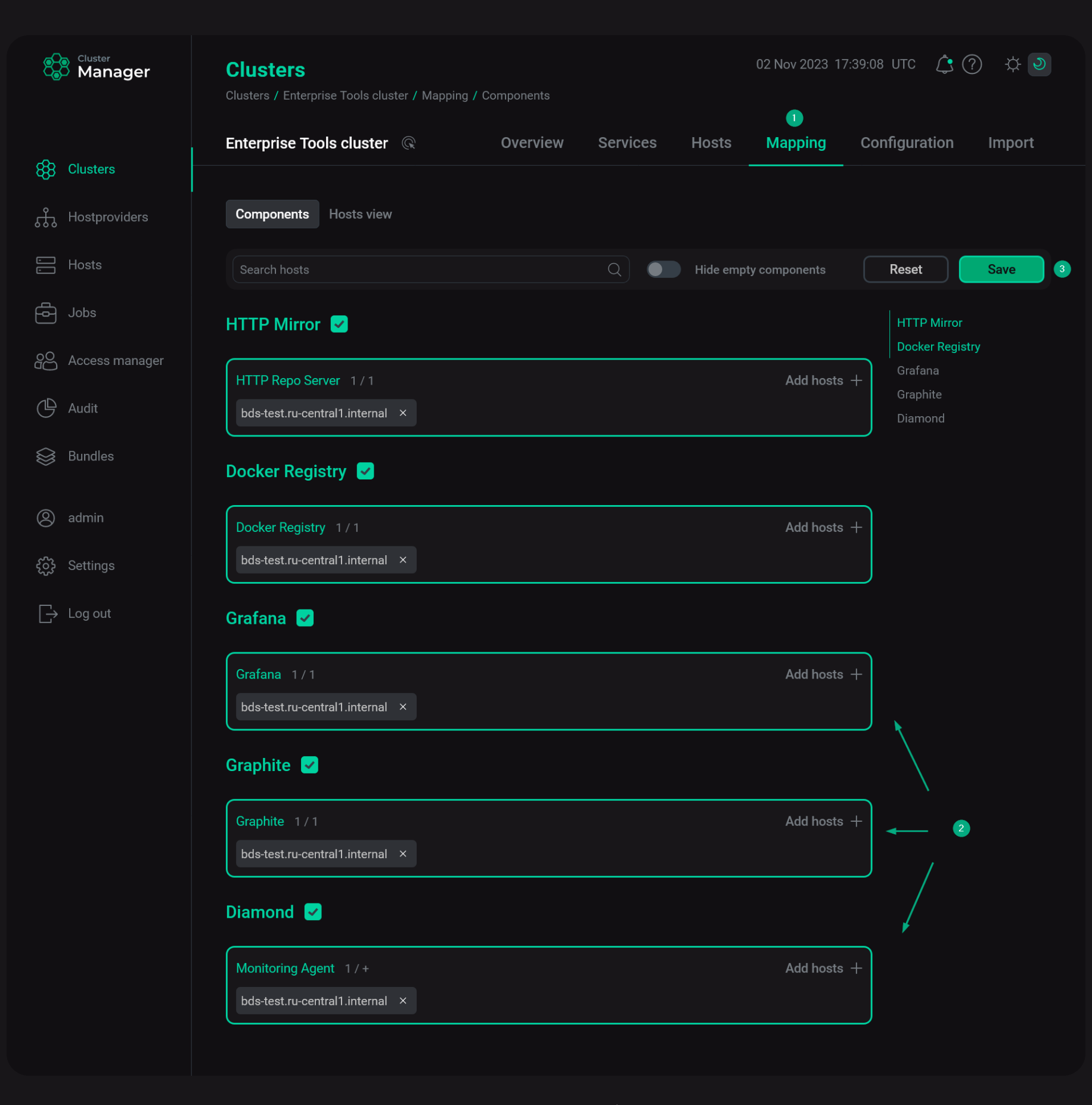
The steps for adding components to hosts are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Clusters
02 Nov 2023 17:33:45 UTC
click
Enterprise Tools cluster
operational
Arenadata Enterprise Tools
2023020200-1
Create cluster

Select a cluster

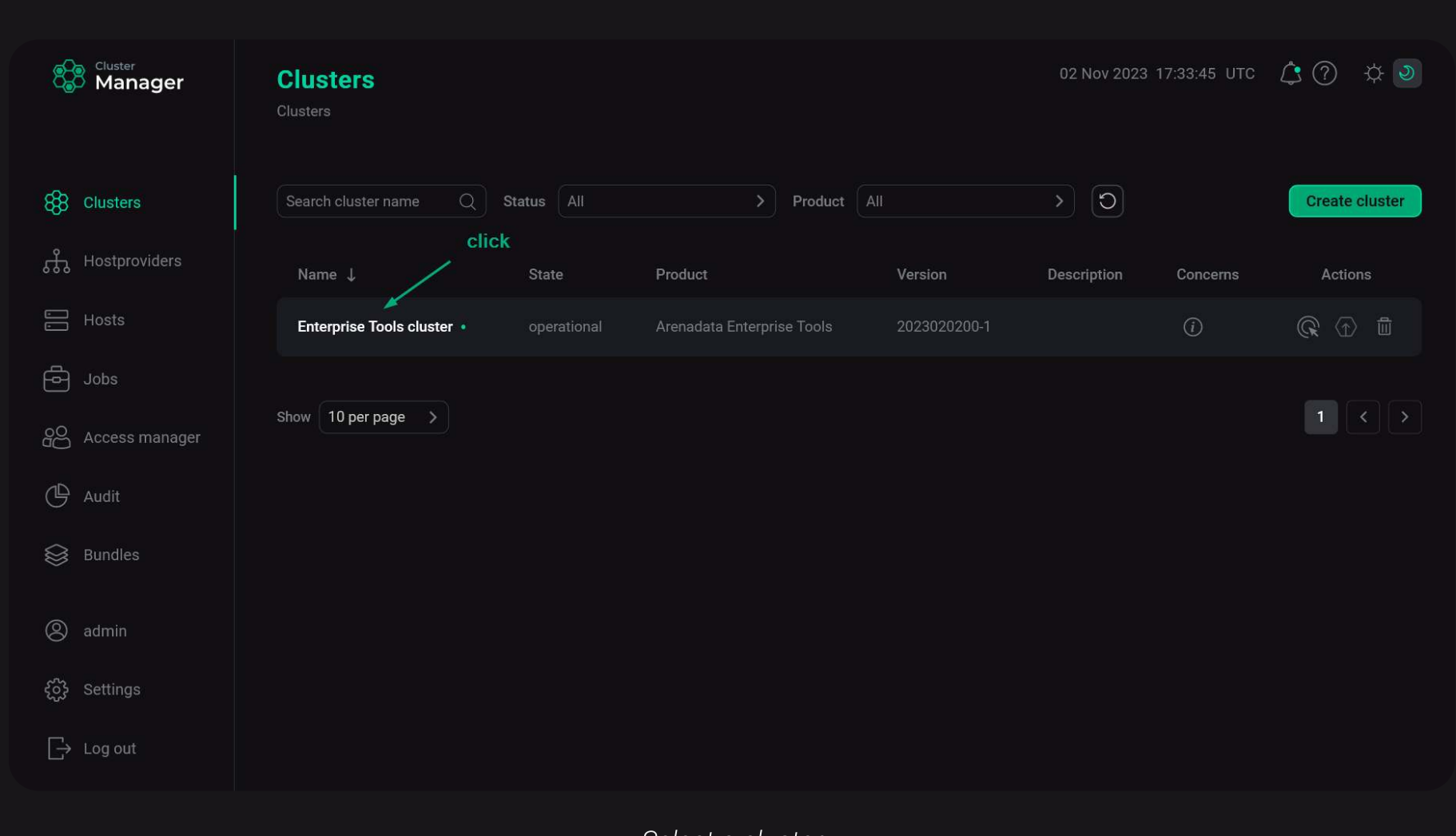
2. Open the ***Mapping*** tab on the cluster page, map monitoring components to all cluster hosts, and click ***Save***.


Clusters
Clusters / Enterprise Tools cluster / Mapping / Components
02 Nov 2023 17:39:08 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
3
HTTP Mirror
HTTP Repo Server 1 / 1
bds-test.ru-central1.internal
Docker Registry
Docker Registry 1 / 1
bds-test.ru-central1.internal
Grafana
Grafana 1 / 1
bds-test.ru-central1.internal
Graphite
Graphite 1 / 1
bds-test.ru-central1.internal
Diamond
Monitoring Agent 1 / +
bds-test.ru-central1.internal
Add hosts
2
HTTP Mirror
Docker Registry
Grafana
Graphite
Diamond

Map components to hosts

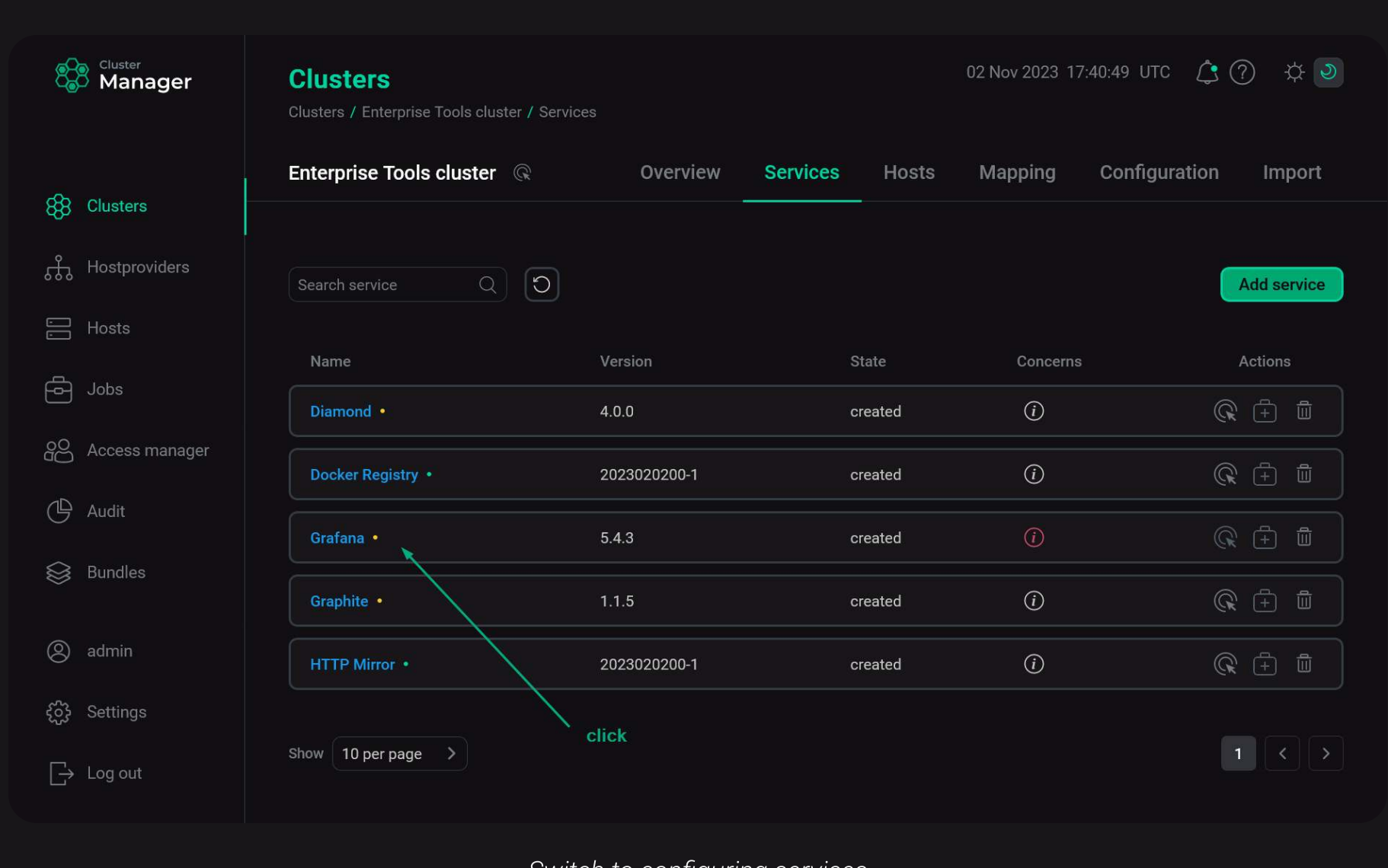
Step 3. Configure monitoring services

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Clusters
02 Nov 2023 17:33:45 UTC
click
Enterprise Tools cluster
operational
Arenadata Enterprise Tools
2023020200-1
Create cluster

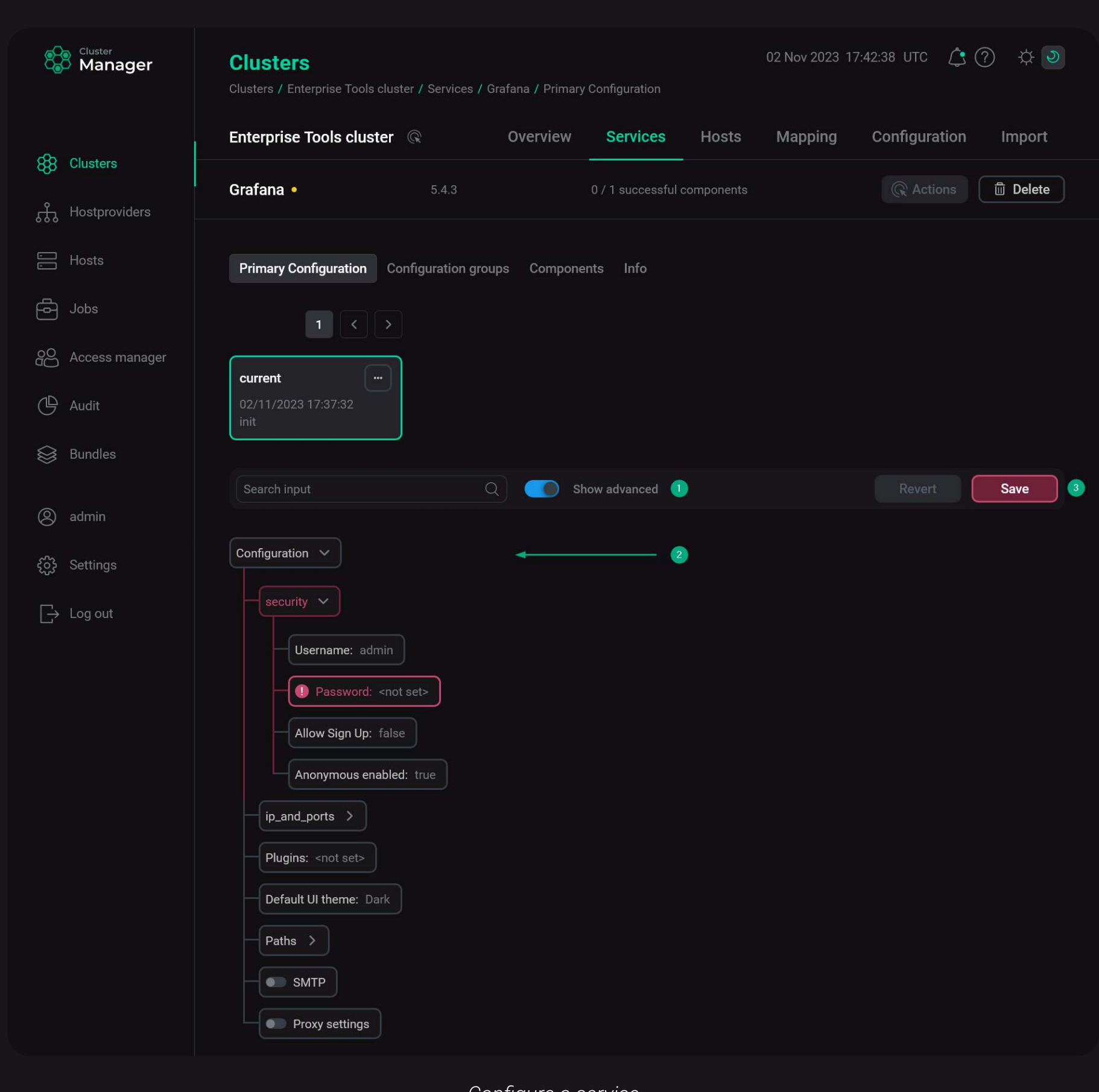
Select a cluster

2. Open the ***Services*** tab on the cluster page, find the row that contains the desired service, and click its name in the ***Name*** column.


Clusters
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 17:40:49 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Add service
Diamond
4.0.0
created
Docker Registry
2023020200-1
created
Grafana
5.4.3
created
Graphite
1.1.5
created
HTTP Mirror
2023020200-1
created
click

Switch to configuring services

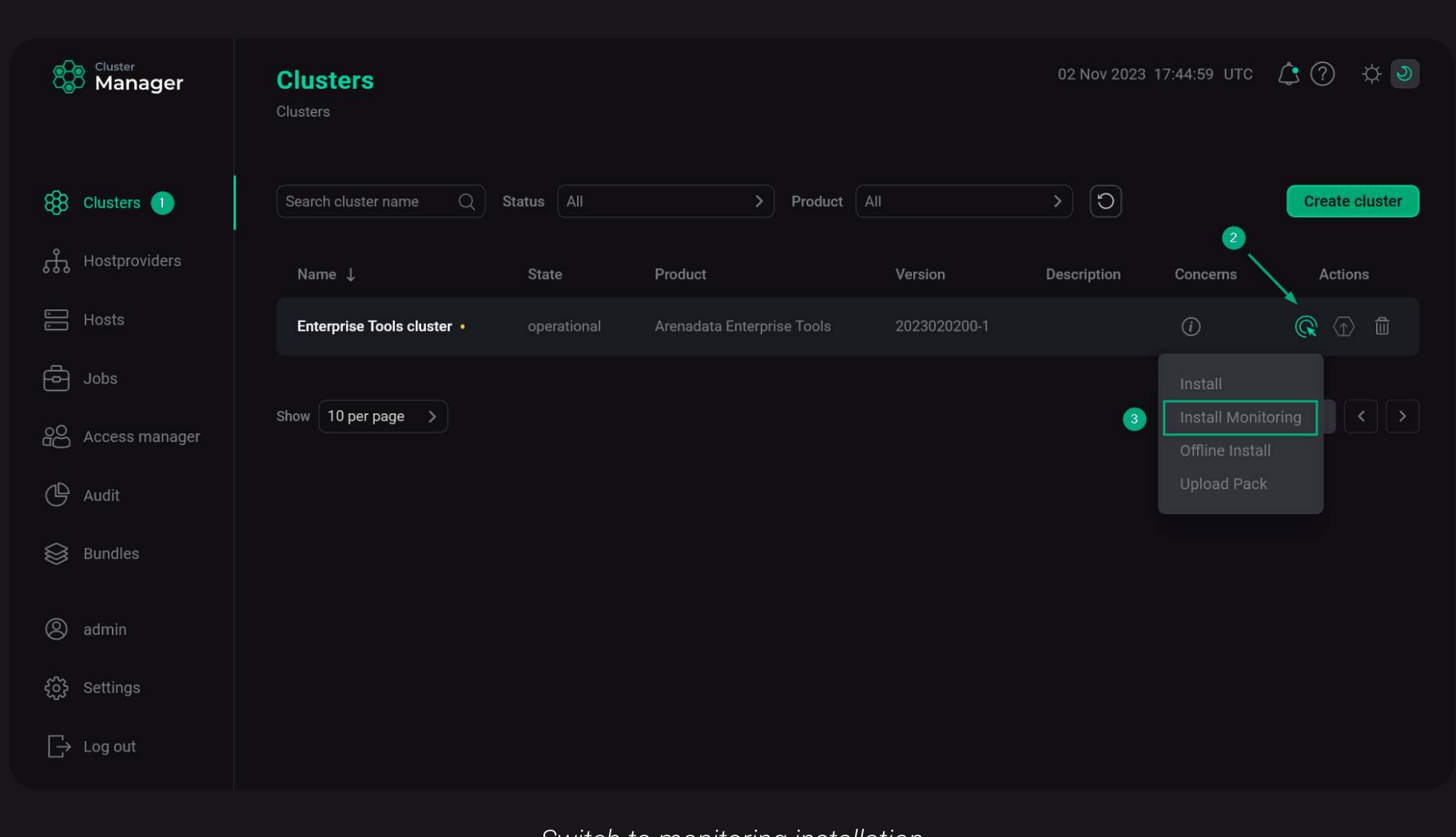
3. In the window that opens:
   - Open the ***Primary Configuration*** tab.
   - Switch on the ***Show advanced*** toggle.
   - Fill in configuration parameters of the selected service. The fields highlighted in red are mandatory.
   - Click ***Save***.


Clusters
Clusters / Enterprise Tools cluster / Services / Grafana / Primary Configuration
02 Nov 2023 17:42:38 UTC
Enterprise Tools cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Grafana
5.4.3
0 / 1 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
current
02/11/2023 17:37:32
init
Search input
Show advanced
1
Revert
Save
3
2
Configuration
security
Username: admin
Password: <not set>
Allow Sign Up: false
Anonymous enabled: true
ip_and_ports
Plugins: <not set>
Default UI theme: Dark
Paths
SMTP
Proxy settings

Configure a service

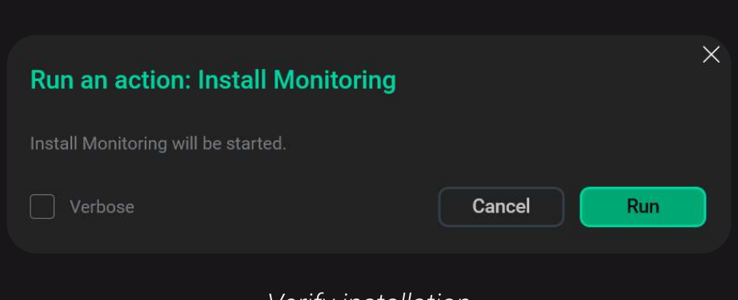
Step 4. Install monitoring services

1. Apply the ***Install Monitoring*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.


Clusters
02 Nov 2023 17:44:59 UTC
Clusters 1
Enterprise Tools cluster
operational
Arenadata Enterprise Tools
2023020200-1
Create cluster
2
3
Install
Install Monitoring
Offline Install
Upload Pack

Switch to monitoring installation

2. In the opened window, click ***Run***.


Run an action: Install Monitoring
Install Monitoring will be started.
☐ Verbose
Cancel
Run

Verify installation

3. Wait until the installation process is completed.

NOTE

In case of errors, you can find logs on the ***Jobs*** page.

Step 5. Verify results

Graphite

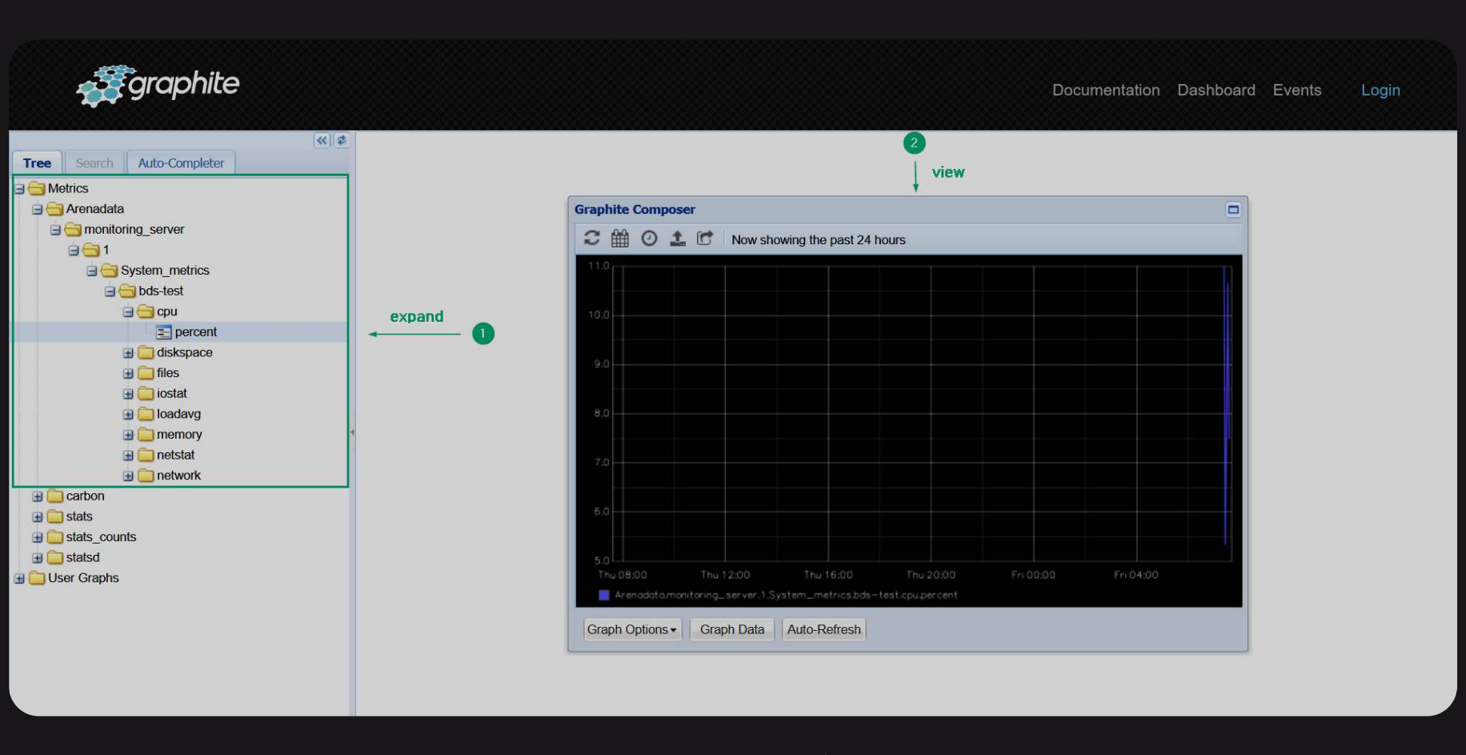
To check availability of the Graphite web-interface, you can do:

1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Web-interface TCP port*** parameter from the ***ip_and_ports*** section defined when configuring the Graphite service during the monitoring cluster installation. The value `80` (used by default) can be omitted.

IMPORTANT

In case of offline installation, this address matches the address of the host where the Enterprise Tools cluster is deployed.

2. Expand a tree at the left side of the opened form. System metrics are displayed there, grouped by hosts. You can select some of these metrics to view their change dynamics in the form of a graph at the right side of the screen.


graphite
Documentation
Dashboard
Events
Login
Tree
Search
Auto-Completer
Metrics
Arenadata
monitoring_server
1
System_metrics
bds-test
cpu
percent
diskspace
files
iostat
loadavg
memory
netstat
network
carbon
stats
stats_counts
statsd
User Graphs
expand
1
2
view
Graphite Composer
Now showing the past 24 hours
Graph Options
Graph Data
Auto-Refresh

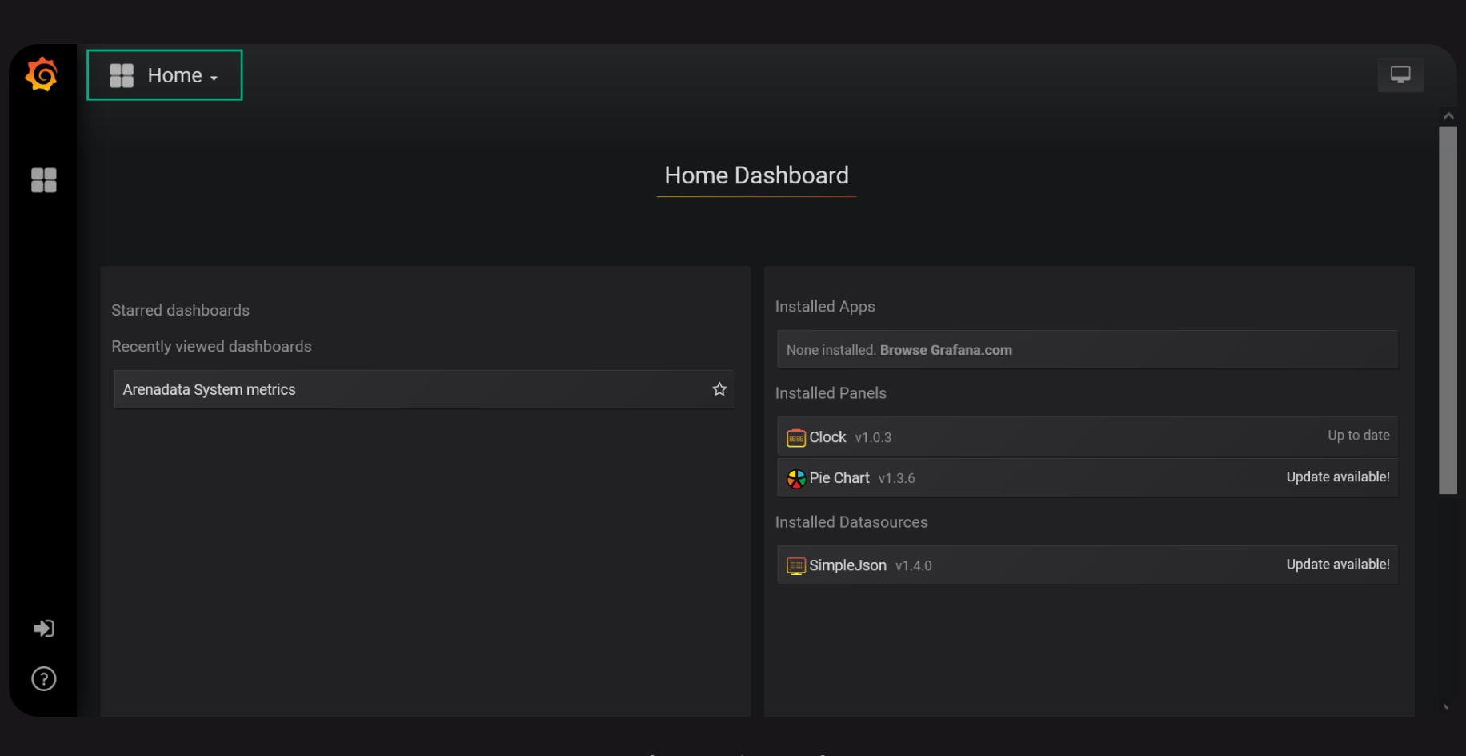
Graphite web-interface

Grafana

To check availability of the Grafana web-interface, you can do:

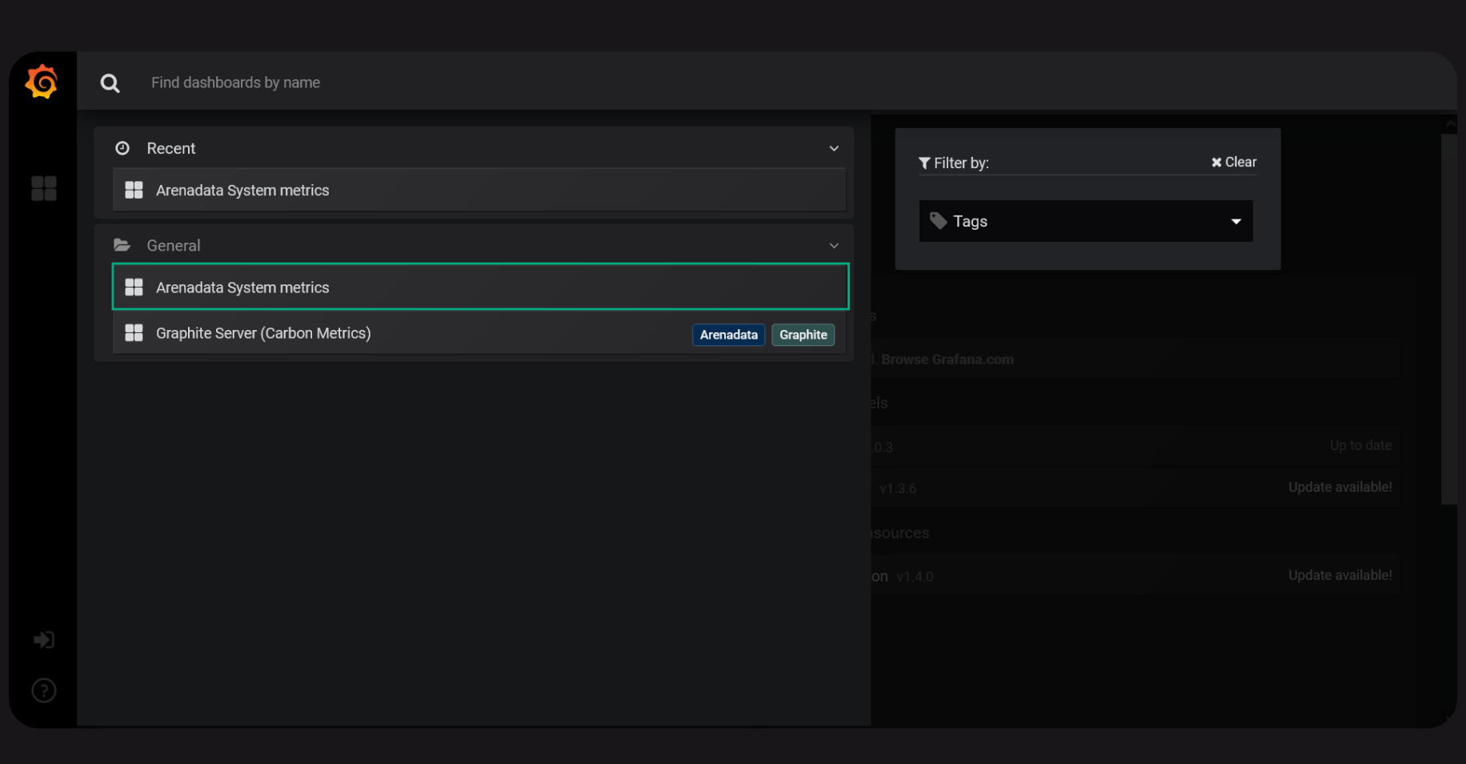
1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Port*** parameter from the ***ip_and_ports*** section defined when configuring the Grafana service during the monitoring cluster installation (by default, `3000`).

2. Select the ***Home*** menu item in the opened form.


Home
Home Dashboard
Starred dashboards
Recently viewed dashboards
Arenadata System metrics
Installed Apps
None installed. Browse Grafana.com
Installed Panels
Clock v1.0.3
Up to date
Pie Chart v1.3.6
Update available!
Installed Datasources
SimpleJson v1.4.0
Update available!

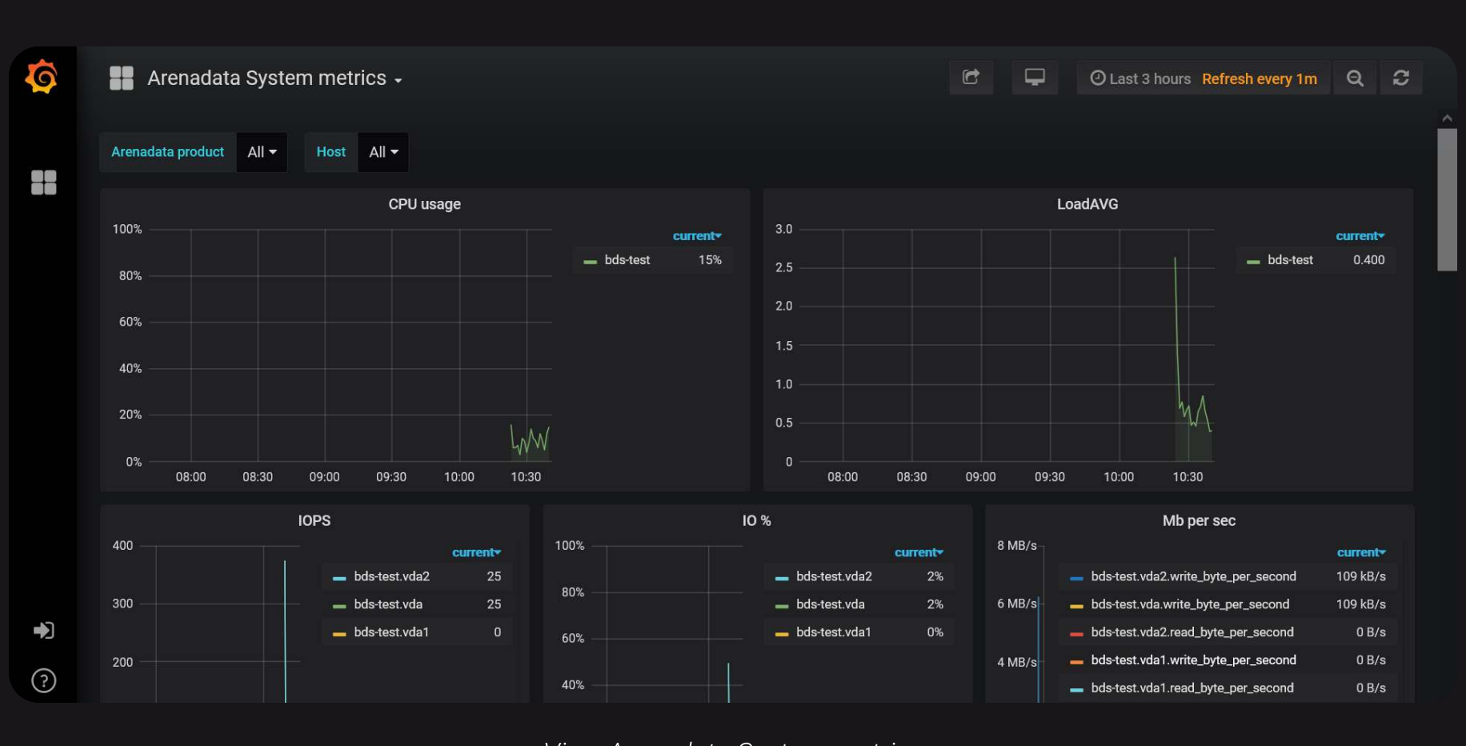
Grafana web-interface

3. Open one of available dashboards. For example, ***Arenadata System metrics***.


Find dashboards by name
Recent
Arenadata System metrics
General
Arenadata System metrics
Graphite Server (Carbon Metrics)
Arenadata
Graphite
Filter by:
Clear
Tags

Switch to Arenadata System metrics

4. View available graphs.


Arenadata System metrics
Last 3 hours
Refresh every 1m
Arenadata product
All
Host
All
CPU usage
LoadAVG
IOPS
IO %
Mb per sec
bds-test
bds-test.vda2
bds-test.vda
bds-test.vda1

View Arenadata System metrics

Integrate with ADH cluster

Eugenia Kuzina, Daria Barysheva

Contents

To Table of Contents

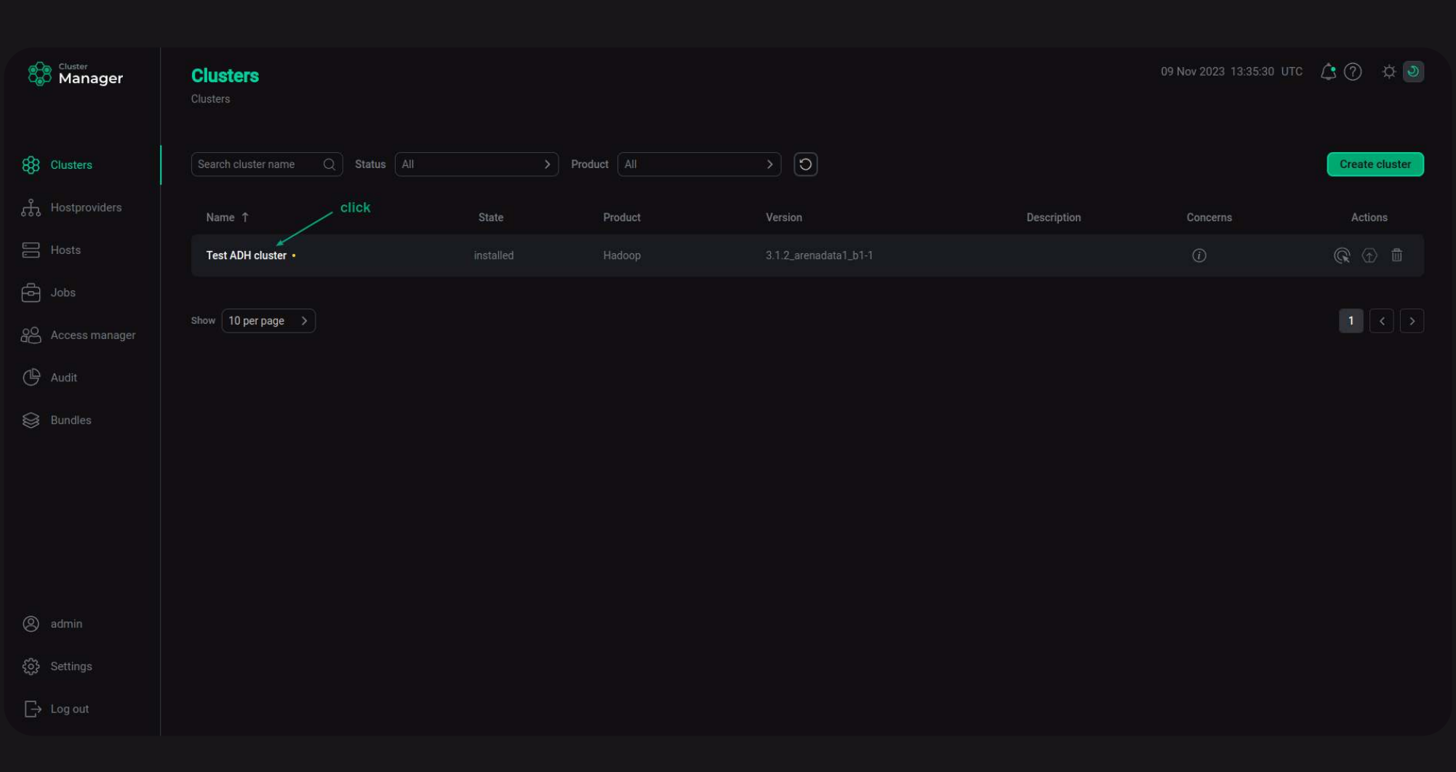
Step 1. Import monitoring settings
Step 2. Add a monitoring service
Step 3. Map monitoring components
Step 4. Install a monitoring service
Step 5. Verify results
- Graphite
- Grafana

In order to determine that a new cluster should be monitored, it is necessary to integrate it with the previously created monitoring cluster and to add a separate Monitoring service into it.

The steps for building integration between a monitoring cluster and ADH cluster are described below. All actions should be performed on the base of ADH cluster.

Step 1. Import monitoring settings

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.

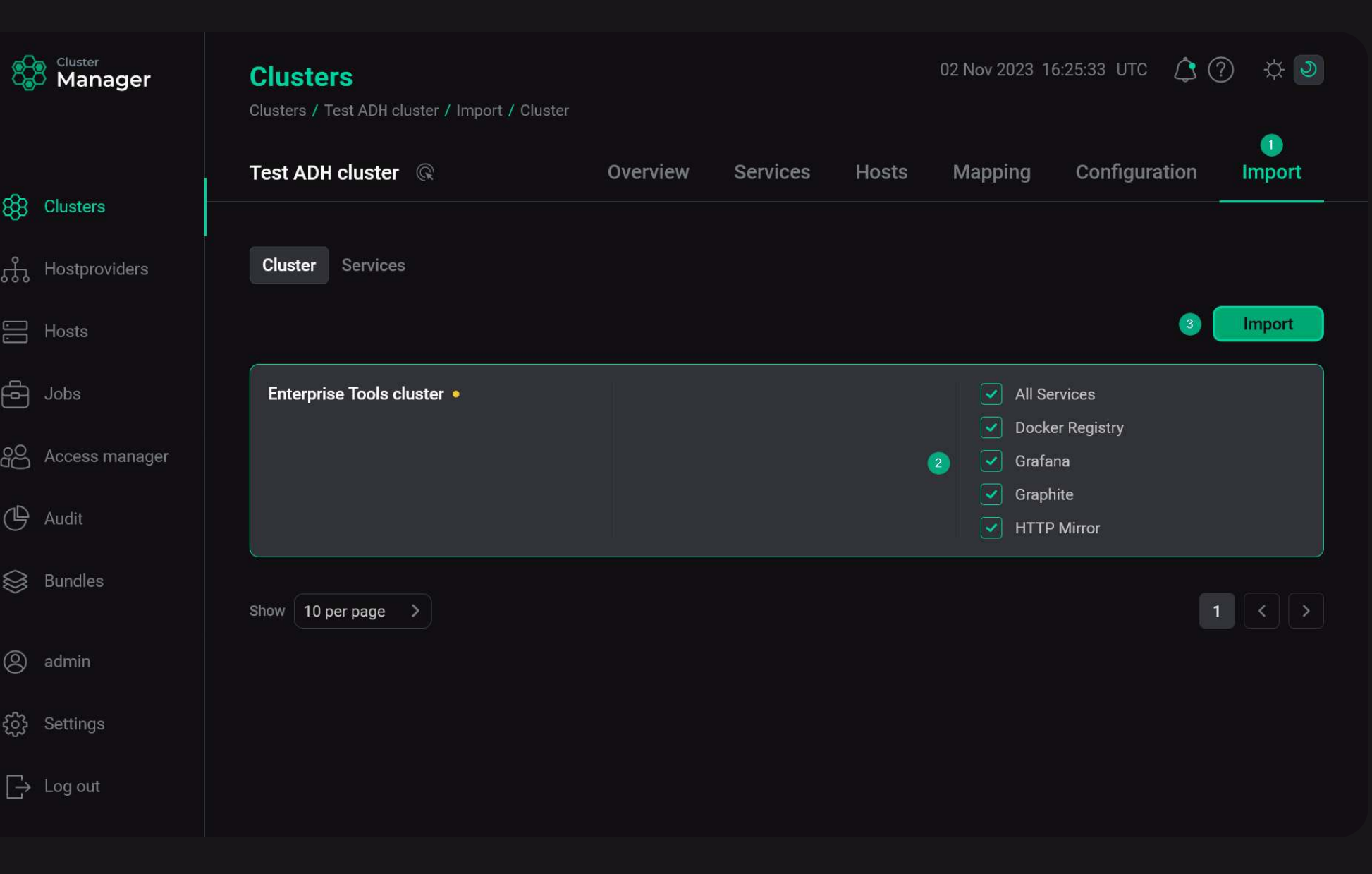


Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			

Cluster Manager
Clusters
09 Nov 2023 13:35:30 UTC
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
click
Show
10 per page
1

Select a cluster

2. Open the ***Import*** tab on the cluster page.
3. Select ***Graphite*** and ***Grafana*** services of the previously created monitoring cluster.
4. Click ***Import***.



Enterprise Tools cluster
☑ All Services
☑ Docker Registry
☑ Grafana
☑ Graphite
☑ HTTP Mirror

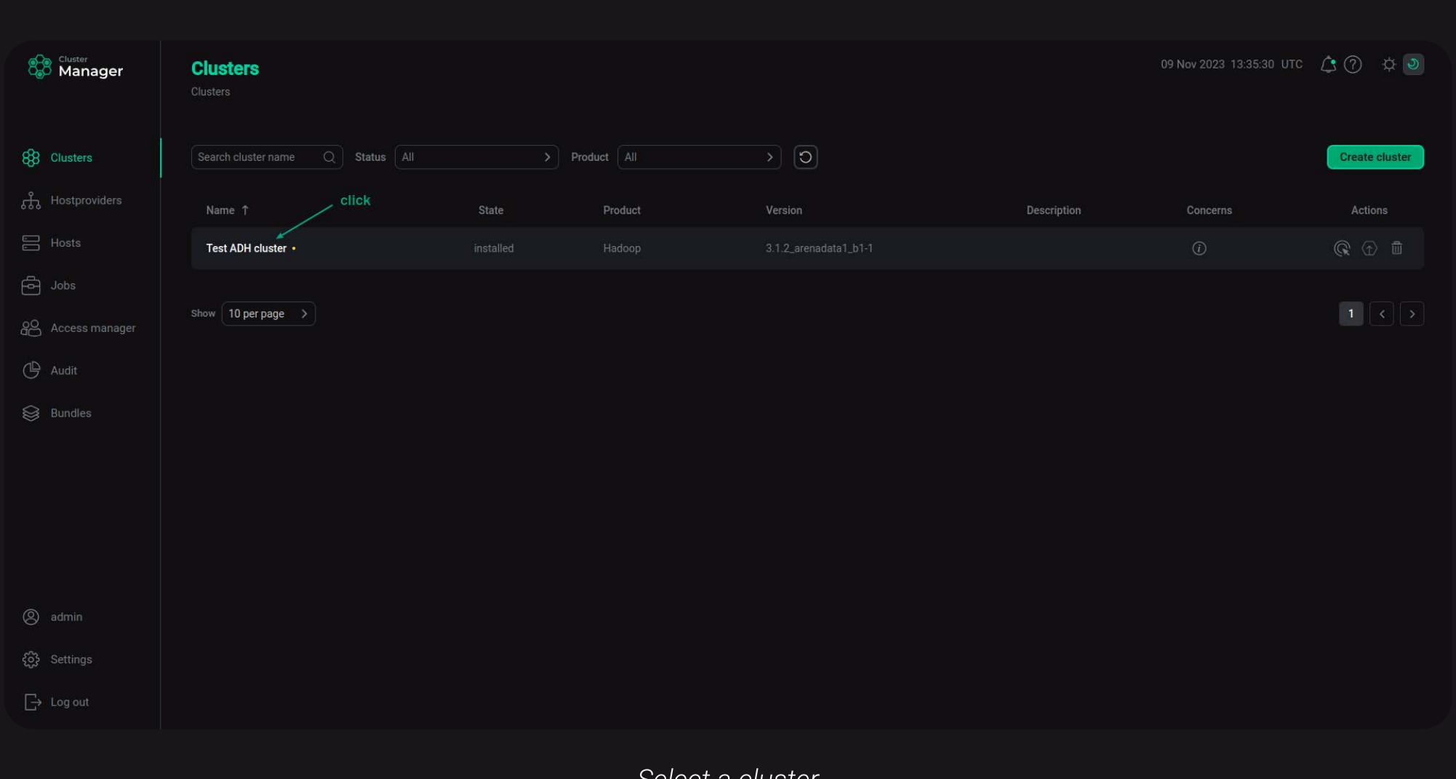
Cluster Manager
Clusters
02 Nov 2023 16:25:33 UTC
Clusters / Test ADH cluster / Import / Cluster
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
1
Import
Cluster
Services
3
Import
2
Show
10 per page
1

Integration with a monitoring cluster

Step 2. Add a monitoring service

The steps for adding services to a cluster are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.

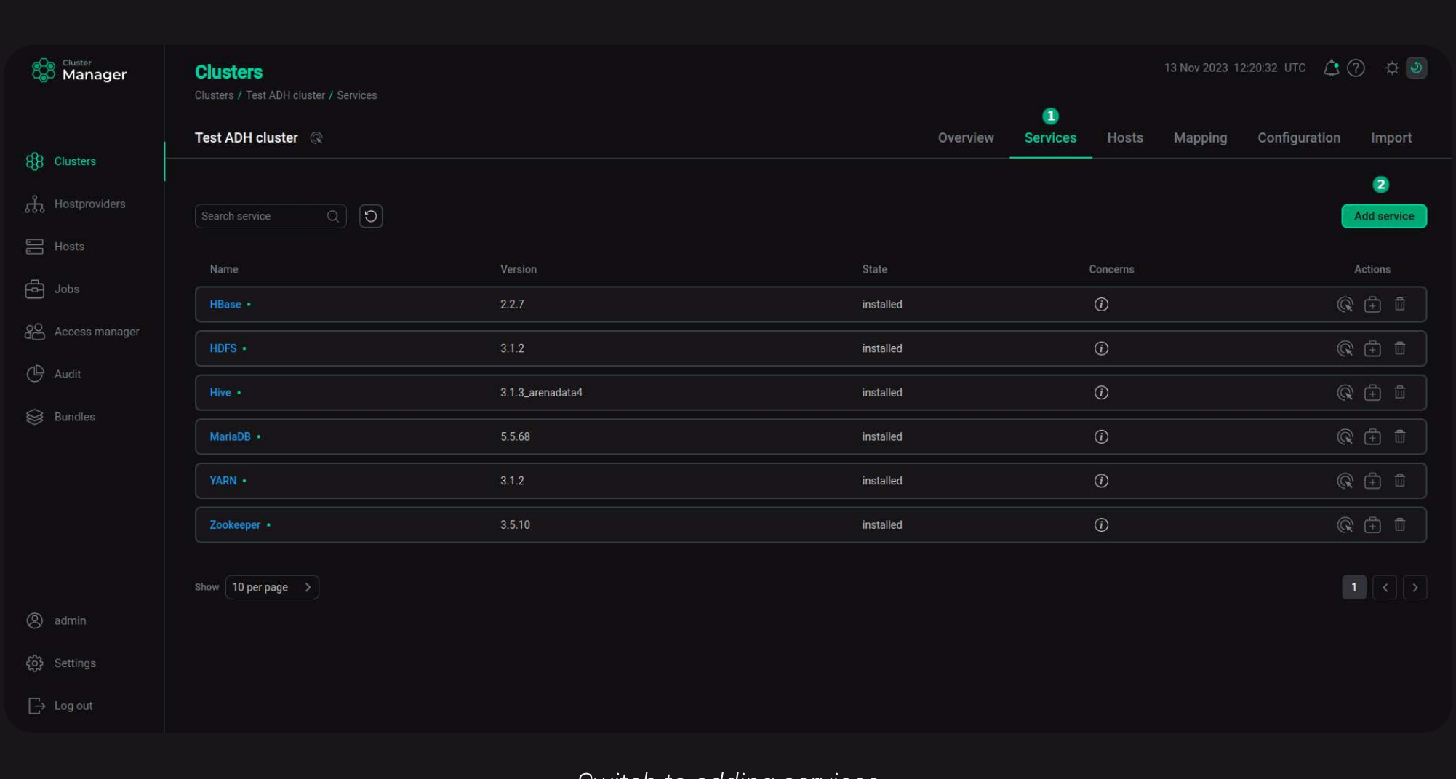


Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			

Cluster Manager
Clusters
09 Nov 2023 13:35:30 UTC
click
Create cluster

Select a cluster

2. Open the ***Services*** tab on the cluster page and click ***Add service***.

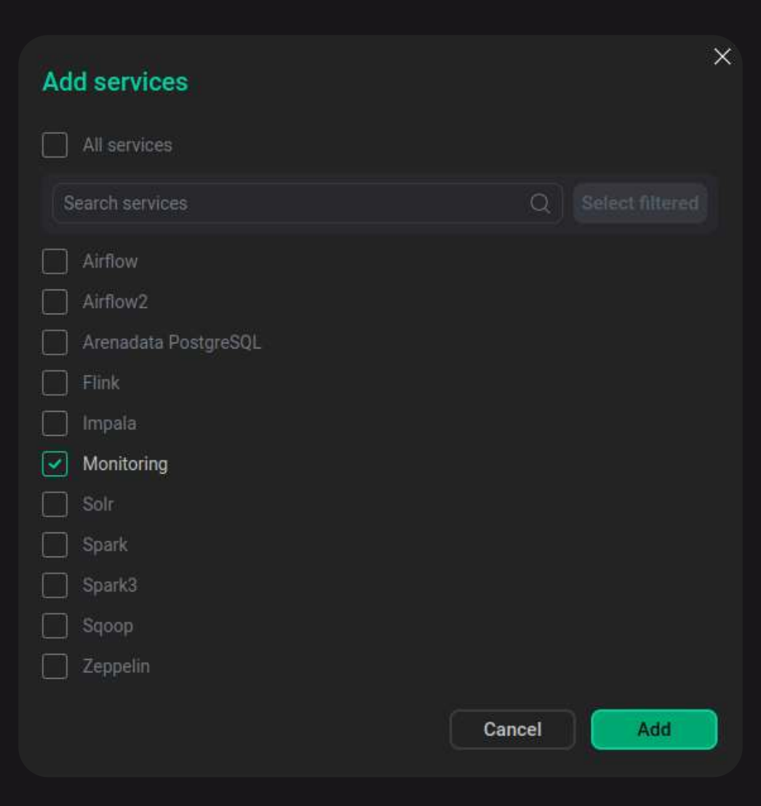


Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

Clusters
13 Nov 2023 12:20:32 UTC
Clusters / Test ADH cluster / Services
Test ADH cluster
Overview
1
Services
Hosts
Mapping
Configuration
Import
2
Add service
Search service

Switch to adding services

3. Select the service Monitoring in the opened dialog and click ***Add***.

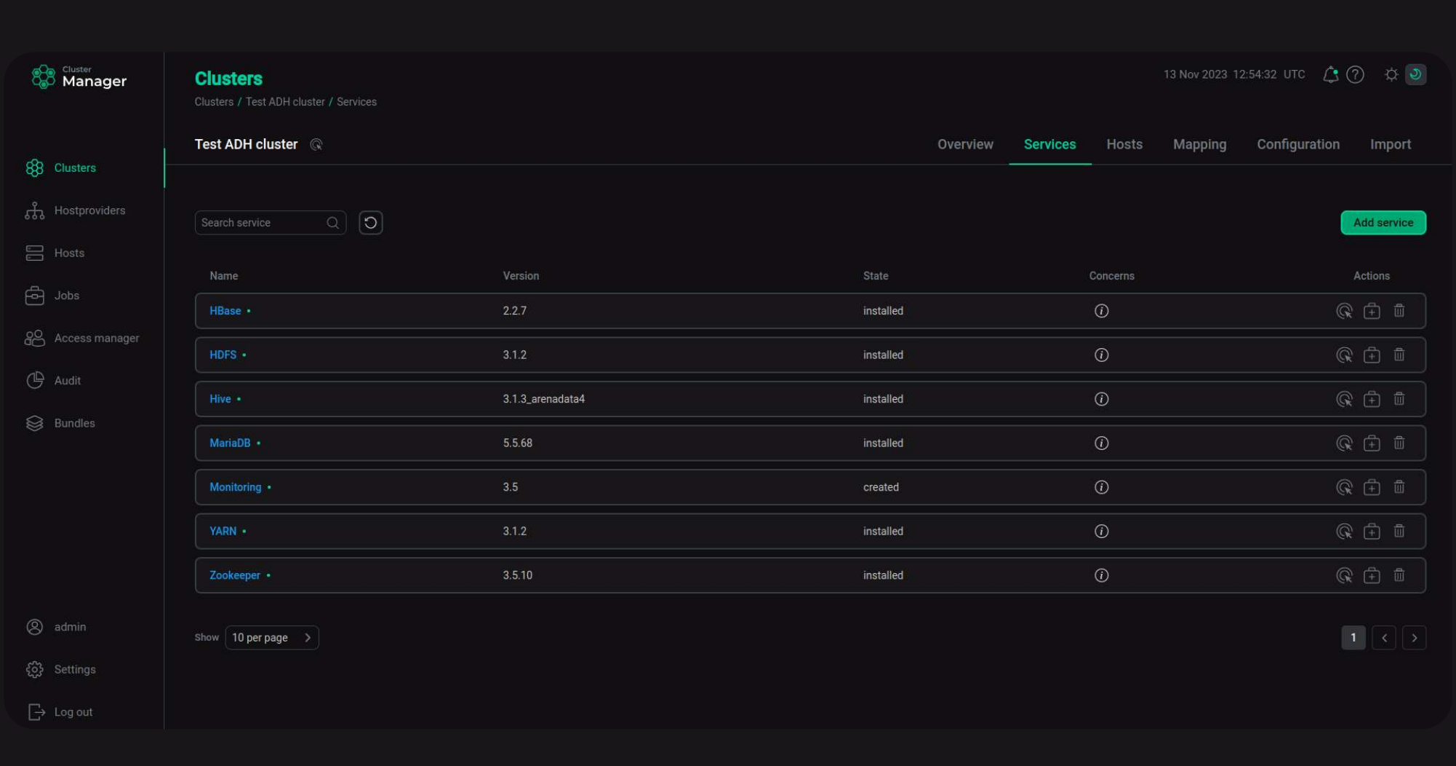


Add services
☐ All services
Search services
Select filtered
☐ Airflow
☐ Airflow2
☐ Arenadata PostgreSQL
☐ Flink
☐ Impala
☑ Monitoring
☐ Solr
☐ Spark
☐ Spark3
☐ Sqoop
☐ Zeppelin

Cancel Add

Select services

4. As a result, the added services are displayed on the ***Services*** tab.



Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
Monitoring	3.5	created		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

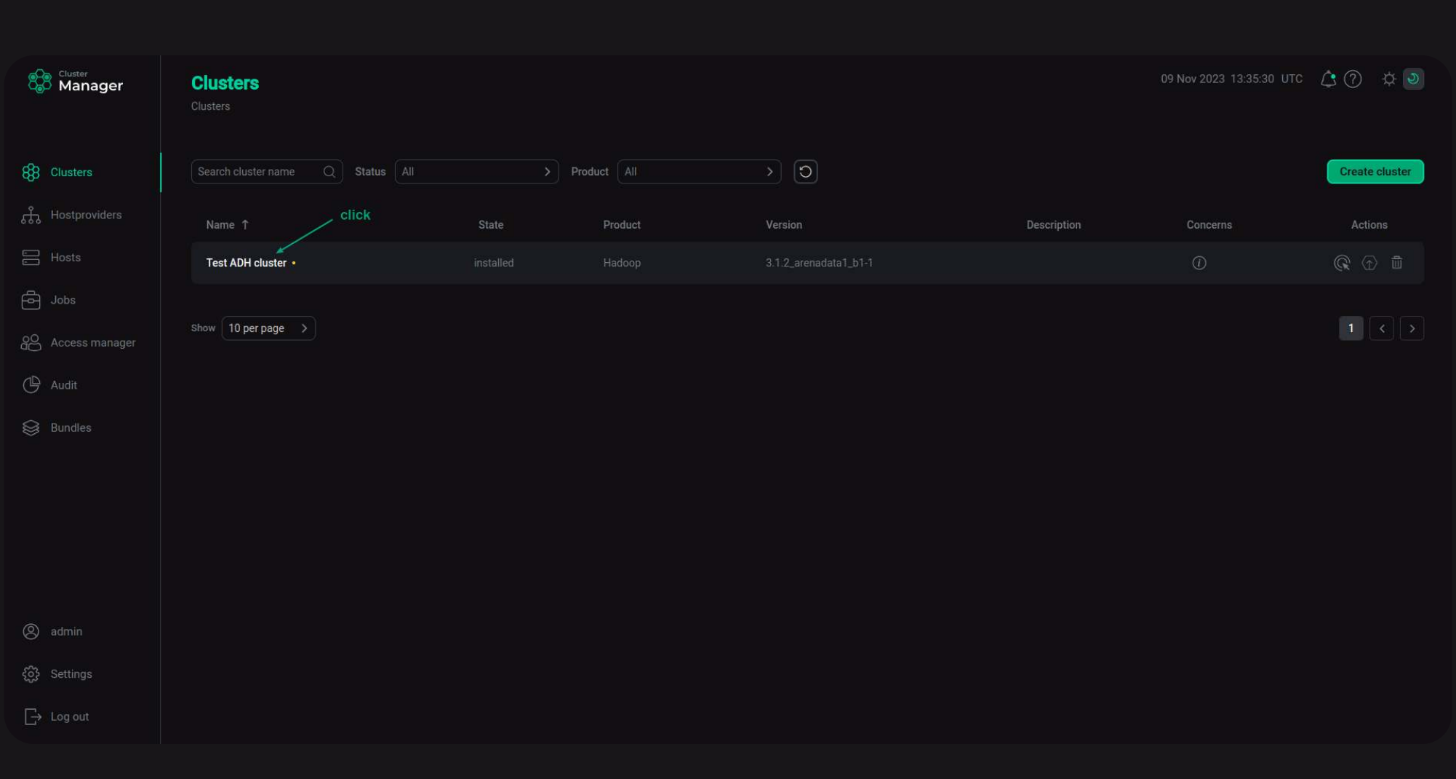
13 Nov 2023 12:54:32 UTC
Add service

The result of successful adding services to a cluster

Step 3. Map monitoring components

The steps for adding components to hosts are listed below:

1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.

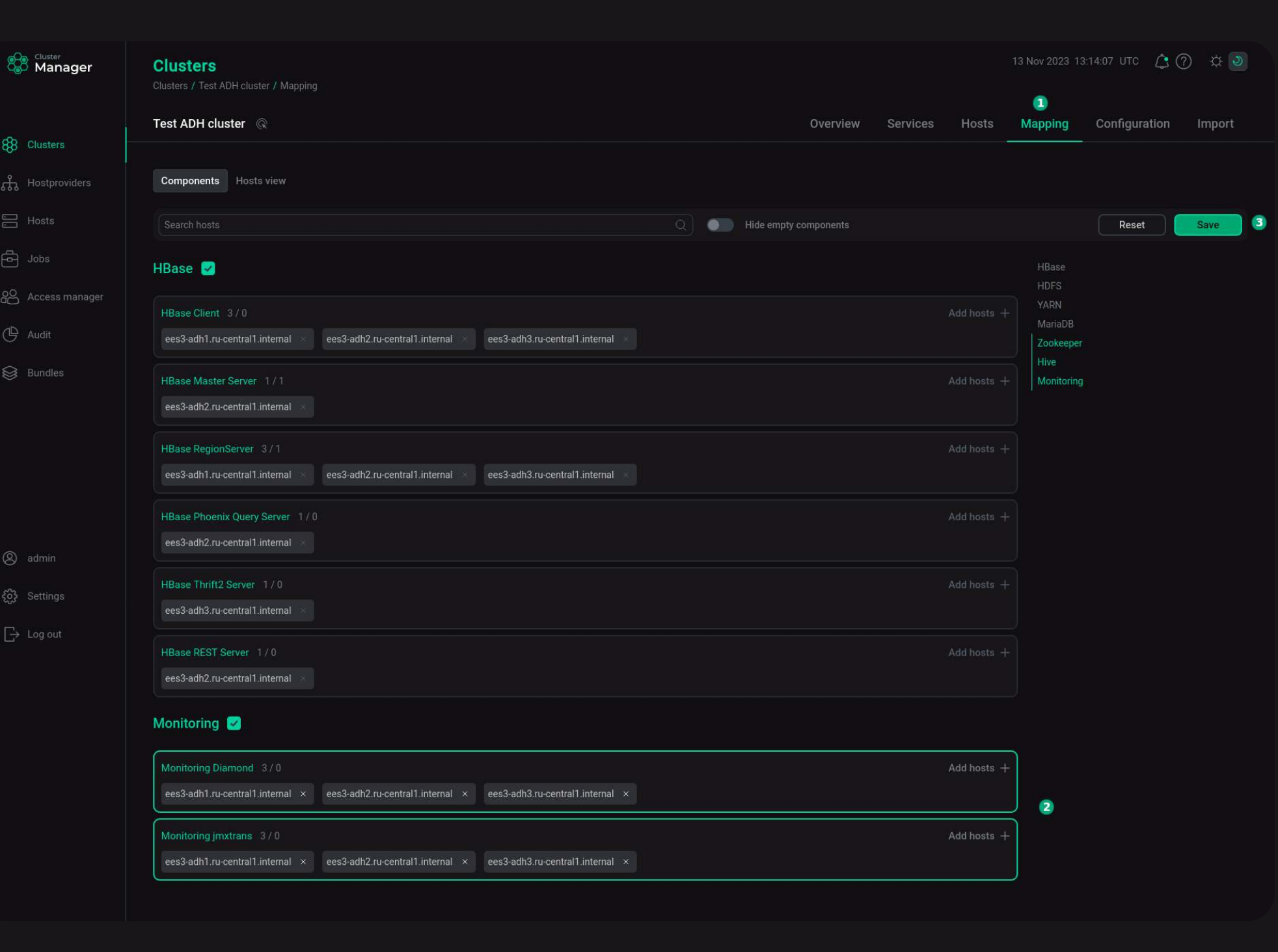


Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			

click
Create cluster

Select a cluster

2. Open the ***Mapping*** tab on the cluster page, map monitoring components to all cluster hosts, and click ***Save***.



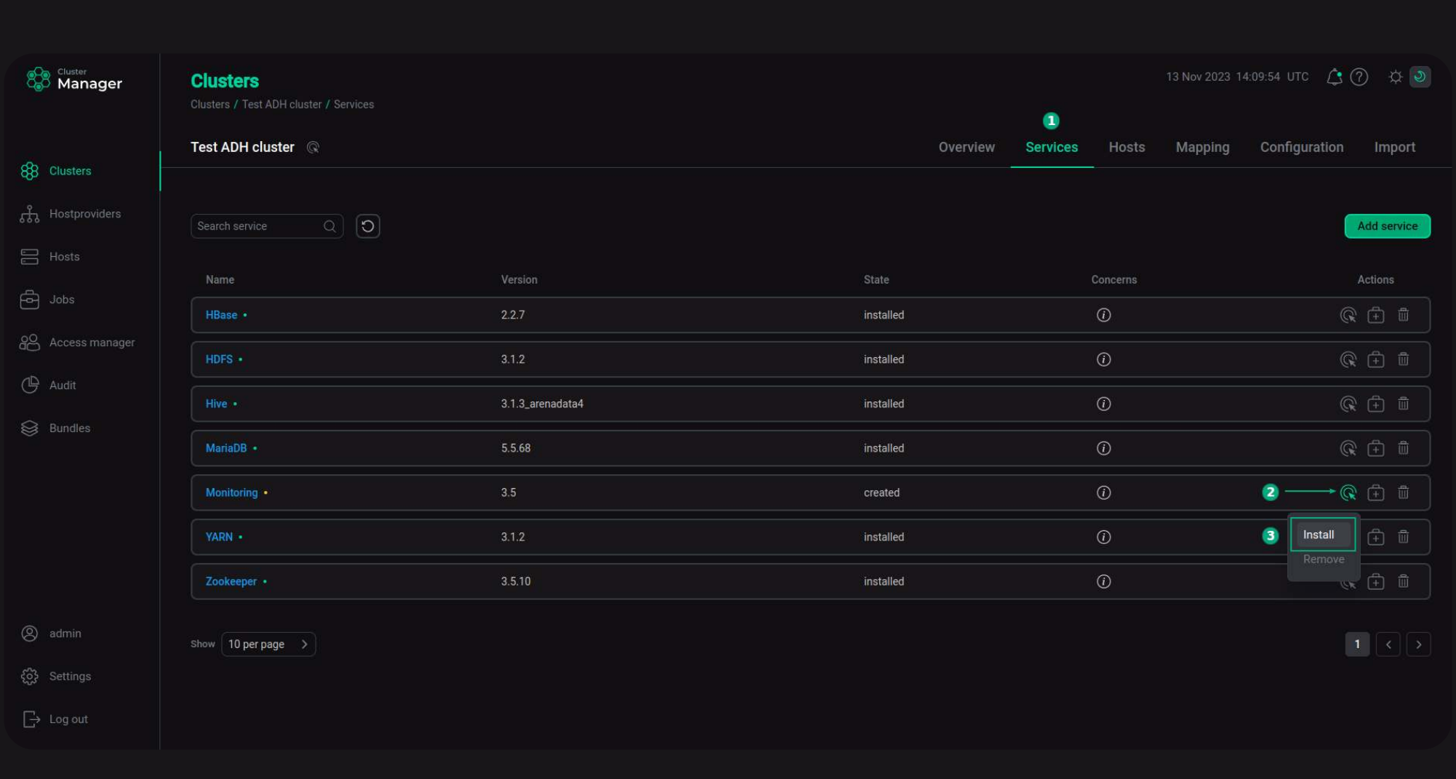
HBase ☑
HBase Client 3 / 0: ees3-adh1.ru-central1.internal, ees3-adh2.ru-central1.internal, ees3-adh3.ru-central1.internal
HBase Master Server 1 / 1: ees3-adh2.ru-central1.internal
HBase RegionServer 3 / 1: ees3-adh1.ru-central1.internal, ees3-adh2.ru-central1.internal, ees3-adh3.ru-central1.internal
HBase Phoenix Query Server 1 / 0: ees3-adh2.ru-central1.internal
HBase Thrift2 Server 1 / 0: ees3-adh3.ru-central1.internal
HBase REST Server 1 / 0: ees3-adh2.ru-central1.internal
Monitoring ☑
Monitoring Diamond 3 / 0: ees3-adh1.ru-central1.internal, ees3-adh2.ru-central1.internal, ees3-adh3.ru-central1.internal
Monitoring jmxtrans 3 / 0: ees3-adh1.ru-central1.internal, ees3-adh2.ru-central1.internal, ees3-adh3.ru-central1.internal

13 Nov 2023 13:14:07 UTC
Clusters / Test ADH cluster / Mapping
1
Mapping
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
3
Add hosts
HBase
HDFS
YARN
MariaDB
Zookeeper
Hive
Monitoring
2

Map components to hosts

Step 4. Install a monitoring service

1. Open the ***Services*** tab in the cluster menu. Click the icon in the ***Actions*** column and select the ***Install*** action in the row that refers to the Monitoring service. Verify the action in the opened window.



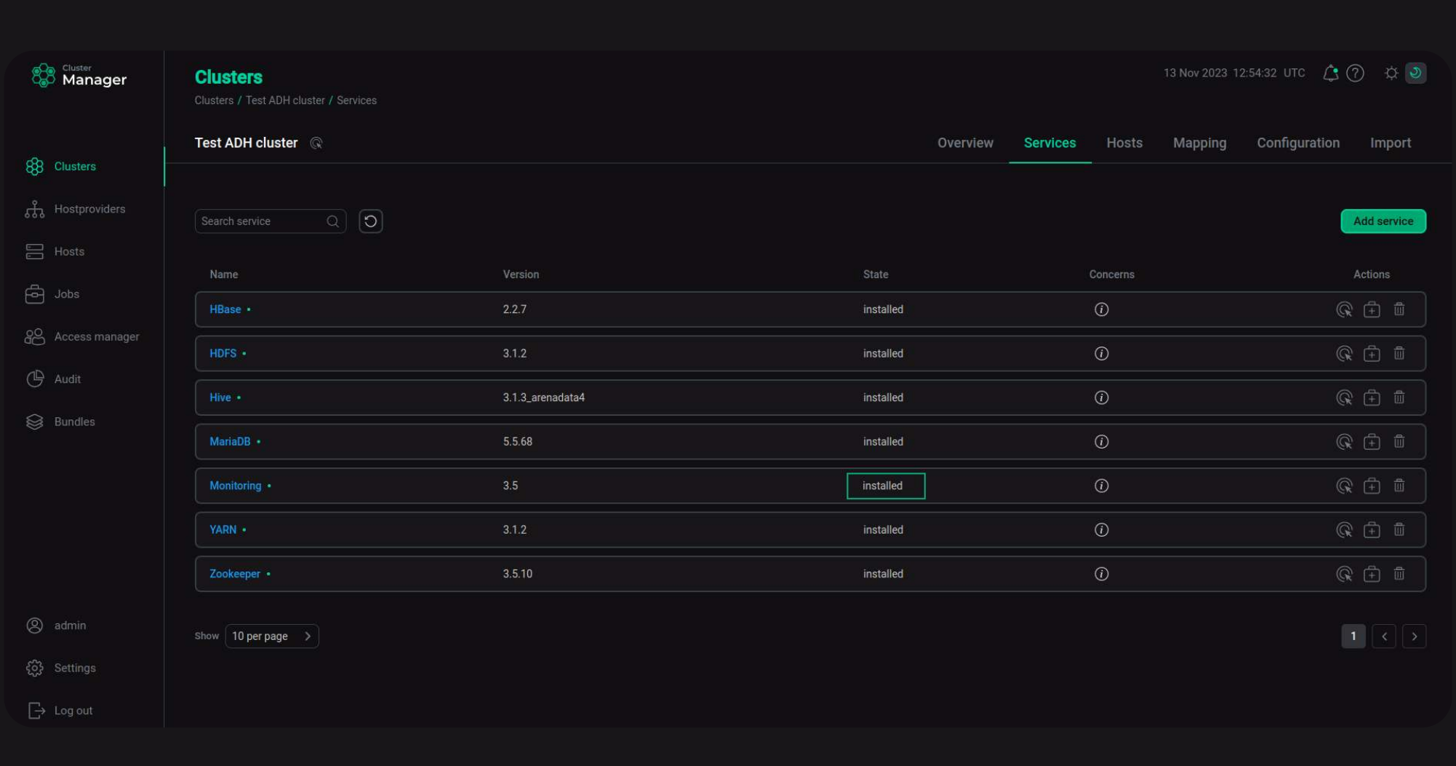
Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
Monitoring	3.5	created		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

13 Nov 2023 14:09:54 UTC
1
Services
2
3
Install
Remove

Switch to monitoring service installation

2. Wait until the job is completed. After successful installation, the monitoring service changes its state from `created` to `installed`.

> **NOTE**
>
> In case of errors, you can find logs on the ***Jobs*** page.



Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
Monitoring	3.5	installed		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

13 Nov 2023 12:54:32 UTC

Monitoring installation is completed

> **CAUTION**
>
> After you install the monitoring cluster, you need to restart the ADH cluster to view metrics.

Step 5. Verify results

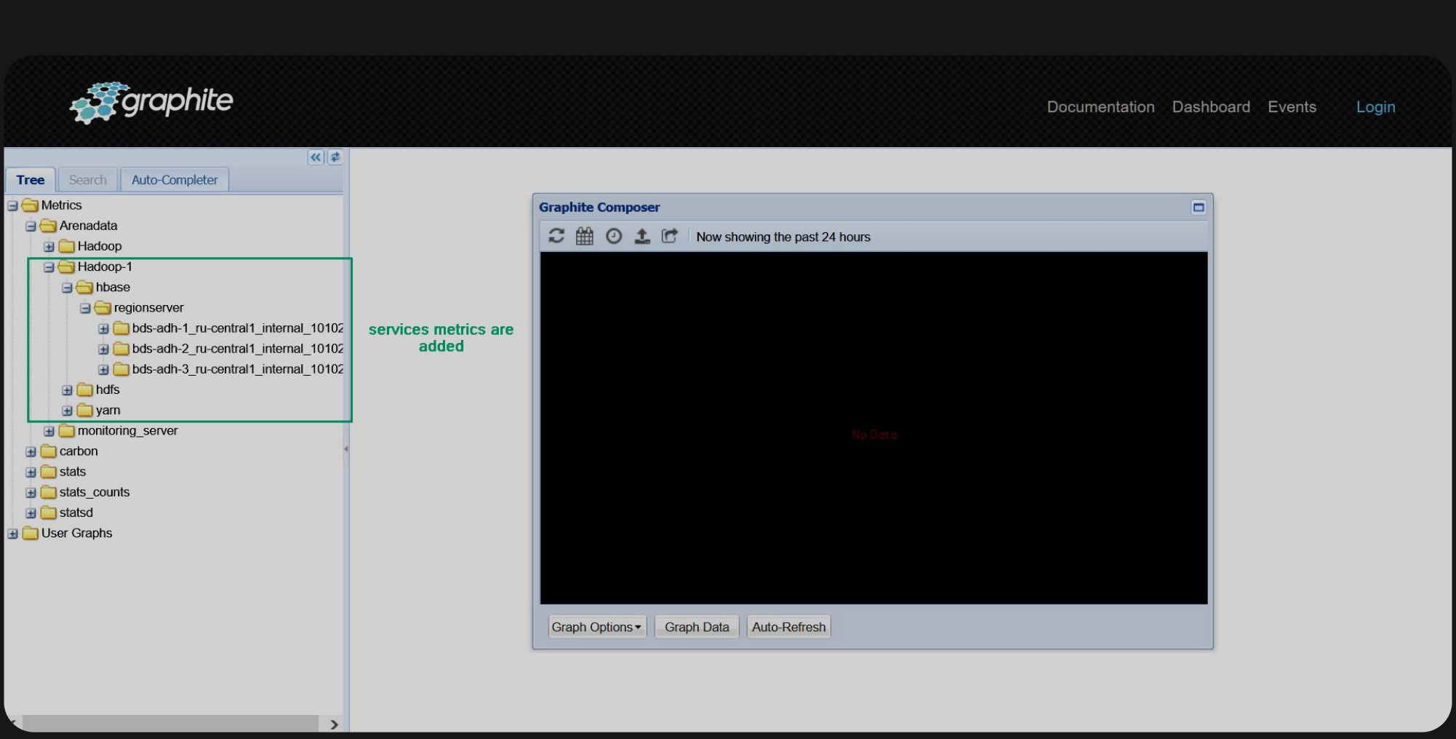
To ensure that installation process is completed successfully, check the following:

- System metrics are collected from the hosts of your cluster, not only from the host where the monitoring cluster has been deployed earlier.
- Along with system metrics, service metrics are collected from the hosts of your cluster. For example, ADH cluster provides metrics for such services as HDFS, YARN, HBase, Spark3.

You can check it both in the Graphite and Grafana web-interface.

Graphite

1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Web-interface TCP port*** parameter from the ***ip_and_ports*** section defined when configuring the Graphite service during the monitoring cluster installation. The value `80` (used by default) can be omitted.
2. Expand a tree at the left side of the opened form. System and service metrics are displayed there, grouped by hosts. You can select some of these metrics to view their change dynamics in the form of a graph at the right side of the screen.

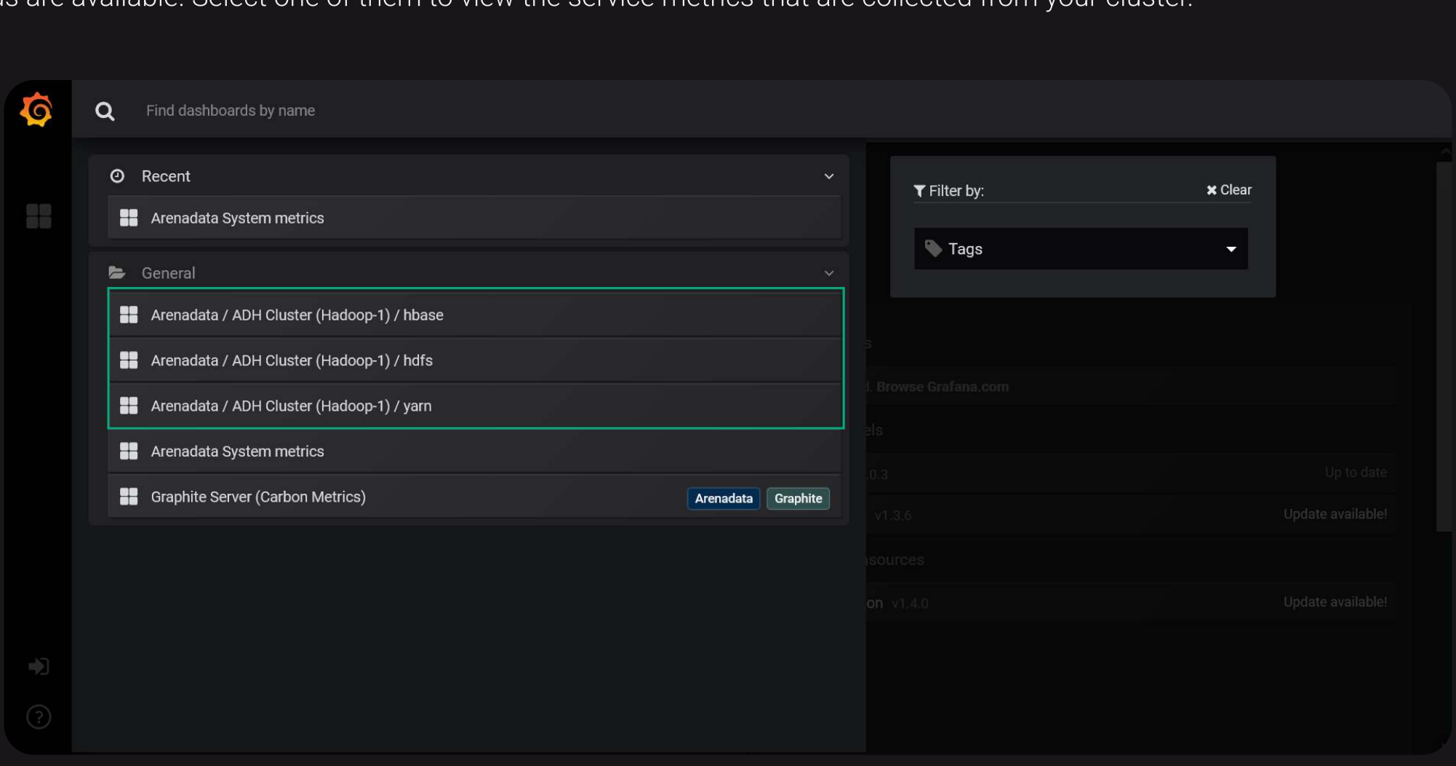


graphite
Documentation
Dashboard
Events
Login
Tree
Search
Auto-Completer
Metrics
Arenadata
Hadoop
Hadoop-1
hbase
regionserver
bds-adh-1_ru-central1_internal_10102
bds-adh-2_ru-central1_internal_10102
bds-adh-3_ru-central1_internal_10102
hdfs
yarn
monitoring_server
carbon
stats
stats_counts
statsd
User Graphs
services metrics are added
Graphite Composer
Now showing the past 24 hours
Graph Options
Graph Data
Auto-Refresh

New hosts and service metrics are available in Graphite

Grafana

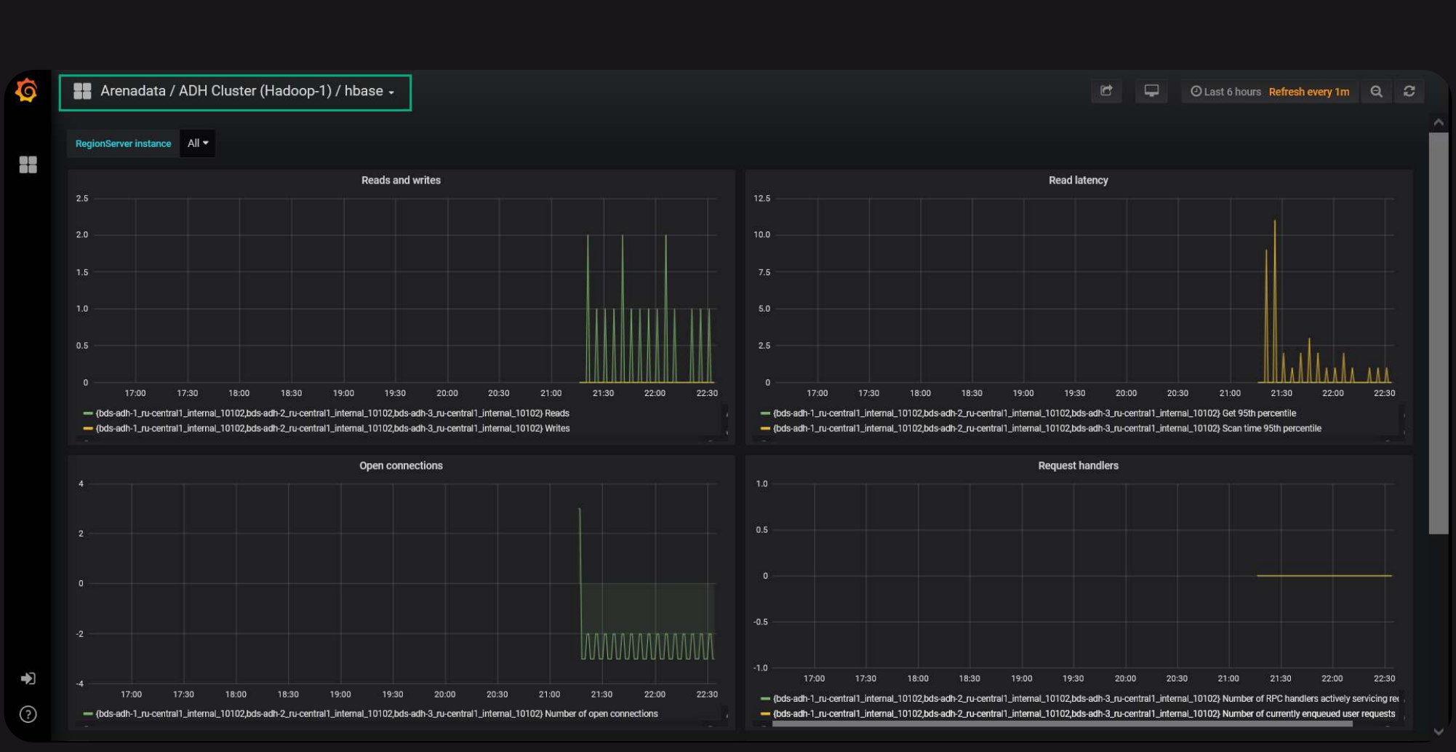
1. In the browser bar, enter an address of the host where the monitoring cluster is deployed. To specify the port number, use the value of the ***Port*** parameter from the ***ip_and_ports*** section defined when configuring the Grafana service during the monitoring cluster installation (by default, `3000`).
2. Select the ***Home*** menu item in the opened form.
3. Notice that new dashboards are available. Select one of them to view the service metrics that are collected from your cluster.



Find dashboards by name
Recent
Arenadata System metrics
General
Arenadata / ADH Cluster (Hadoop-1) / hbase
Arenadata / ADH Cluster (Hadoop-1) / hdfs
Arenadata / ADH Cluster (Hadoop-1) / yarn
Arenadata System metrics
Graphite Server (Carbon Metrics)
Arenadata
Graphite
Filter by:
Clear
Tags

Switch to service metrics

4. View available graphs.



Arenadata / ADH Cluster (Hadoop-1) / hbase
Last 6 hours
Refresh every 1m
RegionServer instance
All
Reads and writes
Read latency

View service metrics

How to

To Table of Contents

Airflow

Flink

HBase

HDFS

Hive

HUE

Impala

Kyuubi

MapReduce

Ozone

Phoenix

Solr

Spark

SSM

Trino

YARN

Zeppelin

ZooKeeper

Connect to Airflow via CLI

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Usage examples

Overview

Airflow provides a command line interface implemented as the *bin/airflow* script located in the */opt/airflow/bin* directory.

While it is recommended to manage Airflow via ADCM or the Airflow web UI, the CLI lets you access an extended number of additional options for working with Airflow.

To interact with Airflow via CLI, connect to a cluster host with an Airflow component via SSH and execute the script by running:

```
$ sudo /opt/airflow/bin/airflow
```

The output looks like this:

```
Usage: airflow [-h] GROUP_OR_COMMAND ...

Positional Arguments:
  GROUP_OR_COMMAND

    Groups:
      celery         Celery components
      config         View configuration
      connections    Manage connections
      dags           Manage DAGs
      db             Database operations
      jobs           Manage jobs
      kubernetes     Tools to help run the KubernetesExecutor
      pools          Manage pools
      providers      Display providers
      roles          Manage roles
      tasks          Manage tasks
      users          Manage users
      variables      Manage variables

    Commands:
      cheat-sheet    Display cheat sheet
      dag-processor  Start a standalone Dag Processor instance
      info           Show information about current Airflow and environment
      kerberos       Start a kerberos ticket renewer
      plugins        Dump information about loaded plugins
      rotate-fernet-key
                     Rotate encrypted connection credentials and variables
      scheduler      Start a scheduler instance
      standalone     Run an all-in-one copy of Airflow
      sync-perm      Update permissions for existing roles and optionally DAGs
      triggerer      Start a triggerer instance
      version        Show the version
      webserver      Start a Airflow webserver instance

Options:
  -h, --help         show this help message and exit
```

The detailed description of each command is available in the Airflow2 CLI command reference.

For example, check the version by running:

```
$ sudo /opt/airflow/bin/airflow version
```

The output:

```
2.6.3
```

You can display help for a specific command by using the `--help` option, for example:

```
$ sudo /opt/airflow/bin/airflow dags --help
```

The commands have the following syntax:

```
$ airflow [GROUP_OR_COMMAND] [SUB_COMMAND] [OPTIONS]
```

Where:

- `GROUP_OR_COMMAND` — an Airflow CLI command or a group of commands like `dags`, `jobs`, etc.
- `SUB_COMMAND` — a command related to a group, for example, `dags list`.
- `OPTIONS` — the selected command's options.

Usage examples

Airflow provides several tools, one of which is the `dags` command, which allows you to manage DAGs using the CLI.

For example, to get a list of existing DAGs, run:

```
$ sudo /opt/airflow/bin/airflow dags list
```

Example output:

```
dag_id                                                   | filepath
| owner   | paused
=========================================================+========================================================================
============+========+=======
dataset_consumes_1                                       | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_consumes_1_and_2                                 | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_consumes_1_never_scheduled                       | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_consumes_unknown_never_scheduled                 | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_produces_1                                       | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
dataset_produces_2                                       | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_datasets.py | airflow | None
example_bash_operator                                    | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_bash_operat | airflow | None
                                                         | or.py
|         |
example_branch_datetime_operator                         | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_branch_date | airflow | None
                                                         | time_operator.py
|         |
example_branch_datetime_operator_2                       | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_branch_date | airflow | None
                                                         | time_operator.py
|         |
example_branch_datetime_operator_3                       | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_branch_date | airflow | None
                                                         | time_operator.py
|         |
example_kubernetes_executor                              | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_kubernetes_ | airflow | None
                                                         | executor.py
|         |
example_local_kubernetes_executor                        | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_local_kuber | airflow | None
                                                         | netes_executor.py
|         |
example_nested_branch_dag                                | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_nested_bran | airflow | None
                                                         | ch_dag.py
|         |
example_params_trigger_ui                                | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/example_params_trig | airflow | None
                                                         | ger_ui.py
|         |
tutorial                                                 | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/tutorial.py         | airflow | None
tutorial_dag                                             | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/tutorial_dag.py     | airflow | None
tutorial_taskflow_api                                    | /opt/airflow/lib/python3.10/site-
packages/airflow/example_dags/tutorial_taskflow_a | airflow | None
                                                         | pi.py
|         |
```

To export the list of DAGs to a file, use the `--output` option and provide one of the following formats as the argument: `json`, `yaml`, `plain`, or `table`.

To run a DAG, use the `trigger` command:

```
$ sudo /opt/airflow/bin/airflow dags trigger tutorial
```

Here `tutorial` is the ID of the DAG to run.

Connect to Airflow via REST API

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Example requests
  - List pools
  - Check import issues
  - Get information about a DAG
  - Trigger a DAG run
  - Get tasks

Overview

Along with the web interfaces, Airflow provides access to its REST API. This article contains a few examples of how to work with DAGs via the API using curl. The full list of endpoints is available in the Airflow REST API specification.

A URL for Airflow API requests has the following format:

http://<server-address>:<port>/api/v1/<endpoint>

Where:

- `<server-address>` — the IP address or the host name of the Airflow server;
- `<port>` — the port of the Airflow server;
- `<endpoint>` — the endpoint of the desired method.

Most endpoints accept JSON as input and return JSON in response. Add the following header when making requests:

```
Content-Type: application/json
```

The example request:

```
$ curl -L 'http://localhost:8080/api/v1/dags' -H 'Content-Type: application/json' --user "admin:admin"
```

This command requests information about existing DAGs in the JSON format.

Output sample:

```
{
      "dag_id": "example_params_ui_tutorial",
      "default_view": "grid",
      "description": "DAG demonstrating various options for a trigger form generated by DAG params",
      "file_token": ".
eJw9yjEOgCAMAMC_sEs1PohURWwE29AS5fdujJecAxYDpHpmfiHTBtLt4mf1ywxKFifB_cYUdaT4YZEcw4FJBwQrFg2NgjXjSpi9dPcDbOcjlA.
9dagyc8E47YZAC_P1pUK1AnuBgA",
      "fileloc": "/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_params_ui_tutorial.py",
      "has_import_errors": false,
      "has_task_concurrency_limits": false,
      "is_active": true,
      "is_paused": true,
      "is_subdag": false,
      "last_expired": null,
      "last_parsed_time": "2024-08-29T13:43:21.591051+00:00",
      "last_pickled": null,
      "max_active_runs": 16,
      "max_active_tasks": 16,
      "next_dagrun": null,
      "next_dagrun_create_after": null,
      "next_dagrun_data_interval_end": null,
      "next_dagrun_data_interval_start": null,
      "owners": [
        "airflow"
      ],
      "pickle_id": null,
      "root_dag_id": null,
      "schedule_interval": null,
      "scheduler_lock": null,
      "tags": [
        {
          "name": "example_ui"
        }
      ],
      "timetable_description": "Never, external triggers only"
    },
```

Example requests

Here are the examples of Airflow REST API requests. The examples that contain the `<dag-id>` placeholder require a DAG ID. Provide the ID of the required DAG instead of the placeholder.

For authorization, examples use the default credentials: the `admin` user with the password `admin`.

List pools

To list the existing Airflow pools, execute a `GET` request to the `pools` endpoint.

For example:

```
$ curl -X GET 'http://localhost:8080/api/v1/pools' \
     -H 'Content-Type: application/json' --user "admin:admin"
```

The output:

```
{
  "pools": [
    {
      "description": "Default pool",
      "name": "default_pool",
      "occupied_slots": 0,
      "open_slots": 128,
      "queued_slots": 0,
      "running_slots": 0,
      "scheduled_slots": 7,
      "slots": 128
    }
  ],
  "total_entries": 1
}
```

Check import issues

To check if there are any DAG import issues, send a request to the `importErrors` endpoint.

For example:

```
$ curl --verbose 'http://localhost:8080/api/v1/importErrors' -H 'Content-Type: application/json' --user "admin:admin"
```

The output:

```
{
  "import_errors": [
    {
      "filename": "/opt/airflow/dags/second_example.py",
      "import_error_id": 7,
      "stack_trace": "Traceback (most recent call last):\n  File \"<frozen importlib._bootstrap_external>\", line 947, in
source_to_code\n  File \"<frozen importlib._bootstrap>\", line 241, in _call_with_frames_removed\n  File
\"/opt/airflow/dags/second_example.py\", line 1\n    mport pandas as pd\n          ^^^^^^\nSyntaxError: invalid syntax\n",
      "timestamp": "2024-09-03T13:24:12.808983+00:00"
    }
  ],
  "total_entries": 1
}
```

Get information about a DAG

To retrieve information about a specific DAG, send a `GET` request with the DAG's ID to the `details` endpoint.

For example:

```
$ curl -v -X GET http://localhost:8080/api/v1/dags/<dag-id>/details \
  -H 'Content-Type: application/json' \
  --user "admin:admin"
```

The output:

```
{
  "catchup": false,
  "concurrency": 16,
  "dag_id": "tutorial_dag",
  "dag_run_timeout": null,
  "default_view": "grid",
  "description": "DAG tutorial",
  "doc_md": "\n### DAG Tutorial Documentation\nThis DAG is demonstrating an Extract -> Transform -> Load pipeline\n",
  "end_date": null,
  "file_token": ".eJw9ysEOQDAMANB_2d1KfJAUtTVKm63C_l5cHF_yAqg5IJdN9AbhGax51nOMQw-VnTrDZcdE9U_04GFC04qpgl-uhVE-RWvhBZv_HbQ.
xzpxxi3Ra3GD87EJmfl6rDYC8ME",
  "fileloc": "/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/tutorial_dag.py",
  "is_active": true,
  "is_paused": true,
  "is_paused_upon_creation": null,
  "is_subdag": false,
  "last_parsed": "2024-09-03T13:20:29.979033+00:00",
  "max_active_runs": 16,
  "max_active_tasks": 16,
  "orientation": "LR",
  "owners": [
    "airflow"
  ],
  "params": {},
  "pickle_id": null,
  "render_template_as_native_obj": false,
  "schedule_interval": null,
  "start_date": "2021-01-01T00:00:00+00:00",
  "tags": [
    {
      "name": "example"
    }
  ],
  "timezone": "Timezone('UTC')"
}
```

Trigger a DAG run

To trigger a DAG, send a `POST` request that contains the DAG run parameters to the `dagRuns` endpoint.

For example:

```
$ curl -v -X POST http://localhost:8080/api/v1/dags/<dag-id>/dagRuns \
  -H 'Content-Type: application/json' \
  -d '<json>' \
  --user "admin:admin"
```

Where `<json>` is the parameters of the DAG run in the JSON format.

For example:

```
{
  "dag_run_id": "string",
  "logical_date": "2019-08-24T14:15:22Z",
  "execution_date": "2019-08-24T14:15:22Z",
  "conf": {},
  "note": "string"
}
```

To set a specific execution date (`logical_date`) for your DAG, pass a timestamp in the `YYYY-MM-DDTHH:mm:S` format.

The output:

```
{
  "conf": {},
  "dag_id": "tutorial_dag",
  "dag_run_id": "string",
  "data_interval_end": "2019-08-24T14:15:22+00:00",
  "data_interval_start": "2019-08-24T14:15:22+00:00",
  "end_date": null,
  "execution_date": "2019-08-24T14:15:22+00:00",
  "external_trigger": true,
  "last_scheduling_decision": null,
  "logical_date": "2019-08-24T14:15:22+00:00",
  "note": "string",
  "run_type": "manual",
  "start_date": null,
  "state": "queued"
}
```

Get tasks

To see the list of tasks for a DAG, send a `GET` request to the DAG's `tasks` endpoint.

The example request:

```
$ curl -v -X GET http://localhost:8080/api/v1/dags/<dag-id>/tasks  \
      -H 'Content-Type: application/json'  \
      --user "admin:admin"
```

The output

```
{
  "tasks": [
    {
      "class_ref": {
        "class_name": "PythonOperator",
        "module_path": "airflow.operators.python"
      },
      "depends_on_past": false,
      "downstream_task_ids": [
        "transform_people"
      ],
      "end_date": null,
      "execution_timeout": null,
      "extra_links": [],
      "is_mapped": false,
      "operator_name": "PythonOperator",
      "owner": "airflow",
      "params": {},
      "pool": "default_pool",
      "pool_slots": 1.0,
      "priority_weight": 1.0,
      "queue": "default",
      "retries": 0.0,
      "retry_delay": {
        "__type": "TimeDelta",
        "days": 0,
        "microseconds": 0,
        "seconds": 300
      },
      "retry_exponential_backoff": false,
      "start_date": "2021-12-17T00:00:00+00:00",
      "task_id": "extract_people",
      "template_fields": [
        "templates_dict",
        "op_args",
        "op_kwargs"
      ],
      "trigger_rule": "all_success",
      "ui_color": "#ffefeb",
      "ui_fgcolor": "#000",
      "wait_for_downstream": false,
      "weight_rule": "downstream"
    },
    {
      "class_ref": {
        "class_name": "PythonOperator",
        "module_path": "airflow.operators.python"
      },
      "depends_on_past": false,
      "downstream_task_ids": [],
      "end_date": null,
      "execution_timeout": null,
      "extra_links": [],
      "is_mapped": false,
      "operator_name": "PythonOperator",
      "owner": "airflow",
      "params": {},
      "pool": "default_pool",
      "pool_slots": 1.0,
      "priority_weight": 1.0,
      "queue": "default",
      "retries": 0.0,
      "retry_delay": {
        "__type": "TimeDelta",
        "days": 0,
        "microseconds": 0,
        "seconds": 300
      },
      "retry_exponential_backoff": false,
      "start_date": "2021-12-17T00:00:00+00:00",
      "task_id": "load_people",
      "template_fields": [
        "templates_dict",
        "op_args",
        "op_kwargs"
      ],
      "trigger_rule": "all_success",
      "ui_color": "#ffefeb",
      "ui_fgcolor": "#000",
      "wait_for_downstream": false,
      "weight_rule": "downstream"
    },
    {
      "class_ref": {
        "class_name": "PythonOperator",
        "module_path": "airflow.operators.python"
      },
      "depends_on_past": false,
      "downstream_task_ids": [
        "load_people"
      ],
      "end_date": null,
      "execution_timeout": null,
      "extra_links": [],
      "is_mapped": false,
      "operator_name": "PythonOperator",
      "owner": "airflow",
      "params": {},
      "pool": "default_pool",
      "pool_slots": 1.0,
      "priority_weight": 1.0,
      "queue": "default",
      "retries": 0.0,
      "retry_delay": {
        "__type": "TimeDelta",
        "days": 0,
        "microseconds": 0,
        "seconds": 300
      },
      "retry_exponential_backoff": false,
      "start_date": "2021-12-17T00:00:00+00:00",
      "task_id": "transform_people",
      "template_fields": [
        "templates_dict",
        "op_args",
        "op_kwargs"
      ],
      "trigger_rule": "all_success",
      "ui_color": "#ffefeb",
      "ui_fgcolor": "#000",
      "wait_for_downstream": false,
      "weight_rule": "downstream"
    }
  ],
  "total_entries": 3
}
```

Airflow UI overview

Daria Barysheva

Contents

To Table of Contents

- Airflow UI
  - DAGs
  - Datasets
  - Security
  - Browse
  - Admin
  - Docs
- Flower UI

By default, Airflow provides two web-based user interface forms:

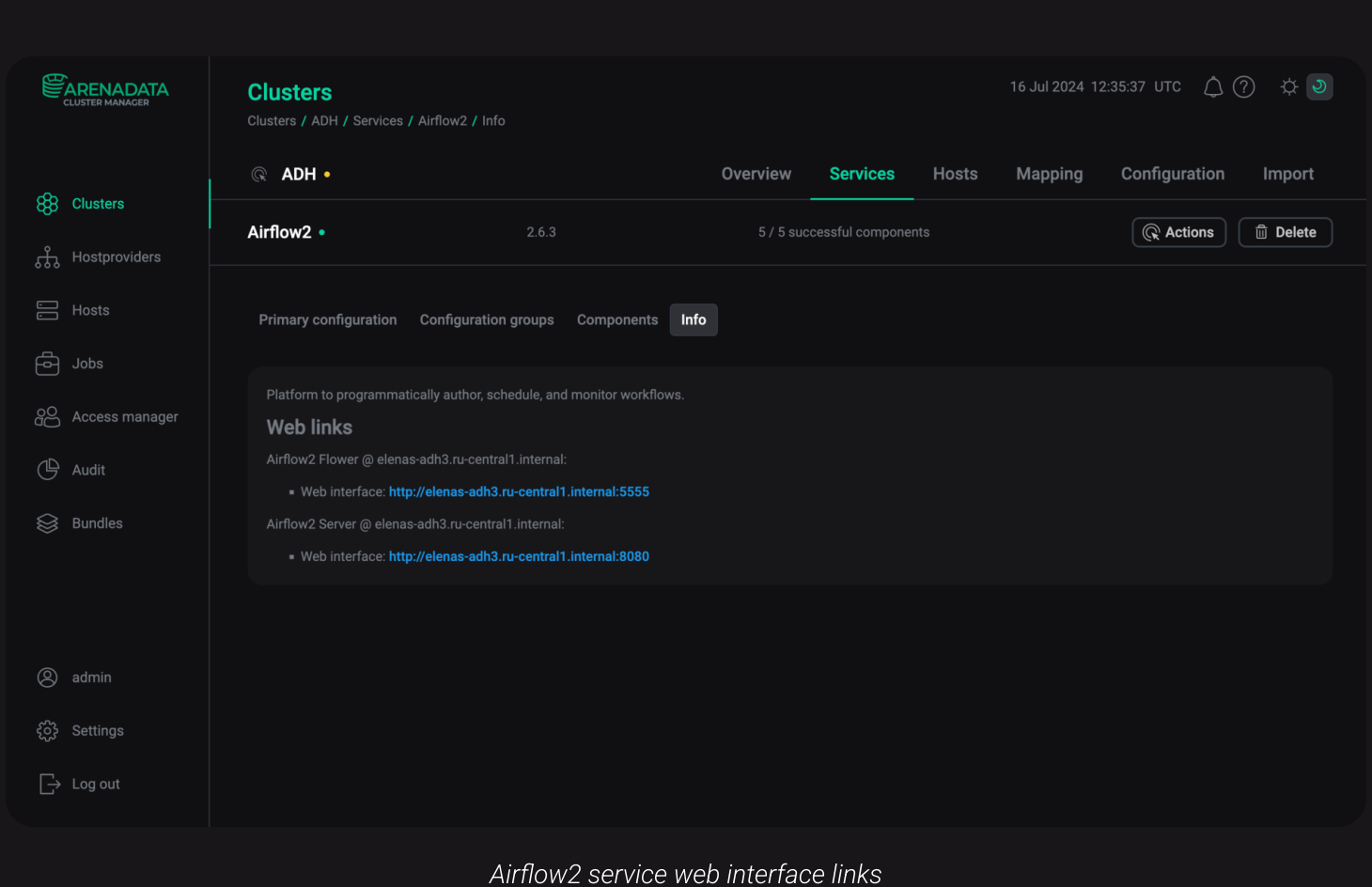
- **Airflow Web UI**. It allows you to monitor DAGs, work with datasets, change security settings, configure the metadata database options (for example, connections to external sources), and so on.
- **Flower**. It is a separate web tool for monitoring and administrating Celery clusters.

Web UI URLs

System	URL	Documentation
Airflow Web UI	http://<Airflow Server IP>:8080	Link
Flower	http://<Airflow Server IP>:5555	Link

If you have an SSL certificate configured, use `https://` instead of `http://`.

The links to both web interfaces are available on the ***Info*** tab, on the service's page in ADCM.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
16 Jul 2024 12:35:37 UTC
Overview
Services
Hosts
Mapping
Configuration
Import
Actions
Delete
Primary configuration
Configuration groups
Components
Info

Clusters

Clusters / ADH / Services / Airflow2 / Info

ADH

Airflow2 2.6.3 5 / 5 successful components

Platform to programmatically author, schedule, and monitor workflows.

Web links

Airflow2 Flower @ elenas-adh3.ru-central1.internal:
- Web interface: http://elenas-adh3.ru-central1.internal:5555

Airflow2 Server @ elenas-adh3.ru-central1.internal:
- Web interface: http://elenas-adh3.ru-central1.internal:8080

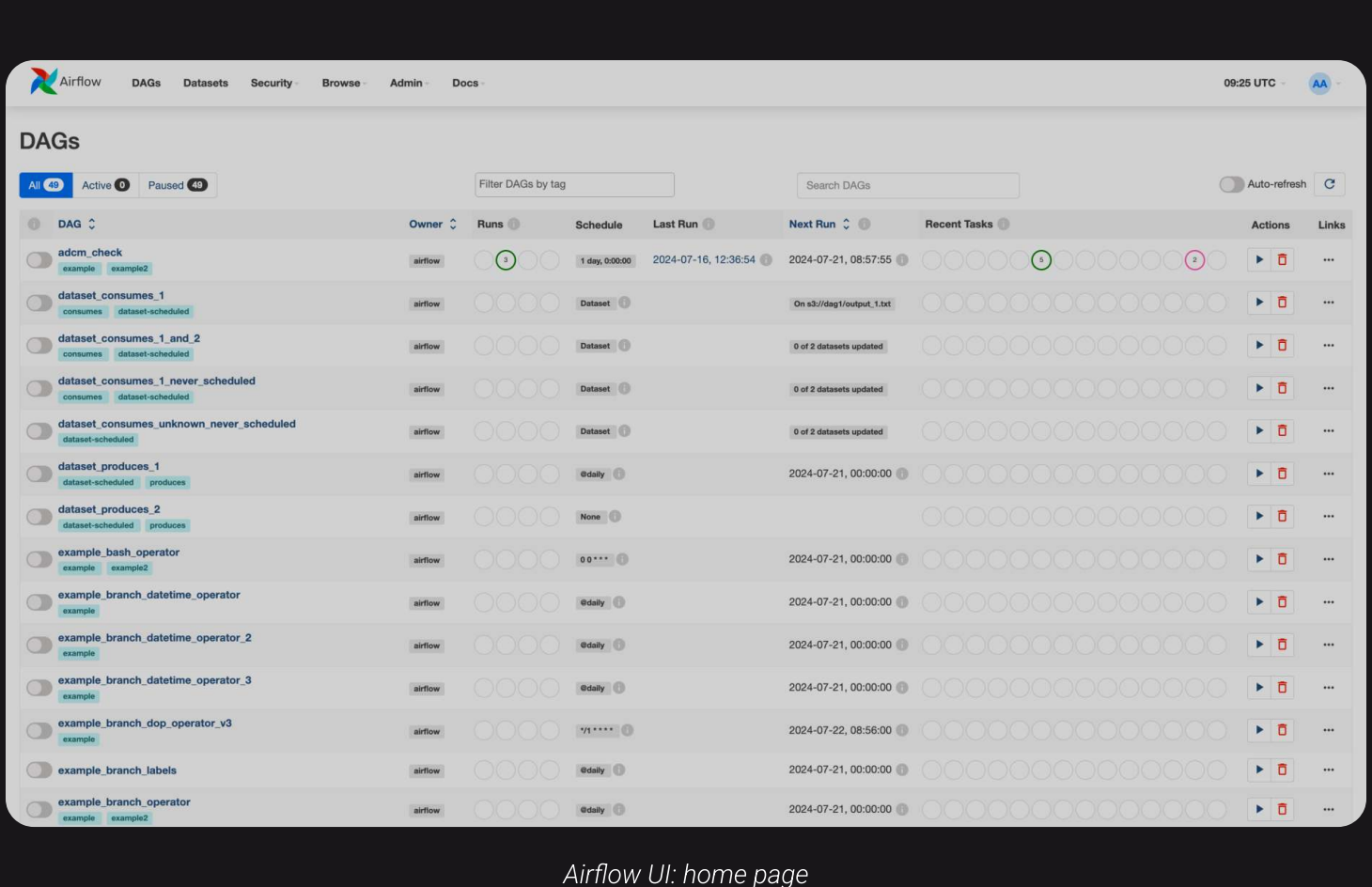
Airflow2 service web interface links

Airflow UI

DAGs

The ***DAGs*** page is the home page of the Airflow UI. It displays a list of data pipelines with information about them, several actions, and shortcuts to useful pages:

- ***DAG*** — the unique DAG identifier name. You can use the toggle button to pause the data pipeline. By clicking on the DAG's name, you can navigate to that DAG's page.
- ***Owner*** — the owner of the DAG as defined in the DAG's code. By clicking on the owner's name, you can filter DAGs to display only those that belong to that owner.
- ***Runs*** — the statuses of all previous DAG runs: queued, success, running, failed.
- ***Schedule*** — indicates how often a DAG runs. Some DAGs are run manually and some on a schedule.
- ***Last Run*** — the time and date of the latest DAG run. You can click on the date to see more information about that DAG.
- ***Next Run*** — the time and date of the next DAG run. For dataset-triggered DAGs, it shows the number of datasets that have been updated since the last DAG run.
- ***Recent Tasks*** — the statuses of the current or latest DAG runs.
- ***Action*** — using the actions menu, you can trigger a DAG, trigger a DAG with a specific configuration, or delete a DAG. The delete action doesn't actually delete the DAG, only the related metadata.
- ***Links*** — in the link menu, there are shortcuts to specific tabs on the DAG page.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
09:25 UTC
AA
All 48
Active 0
Paused 48
Filter DAGs by tag
Search DAGs
Auto-refresh

DAGs

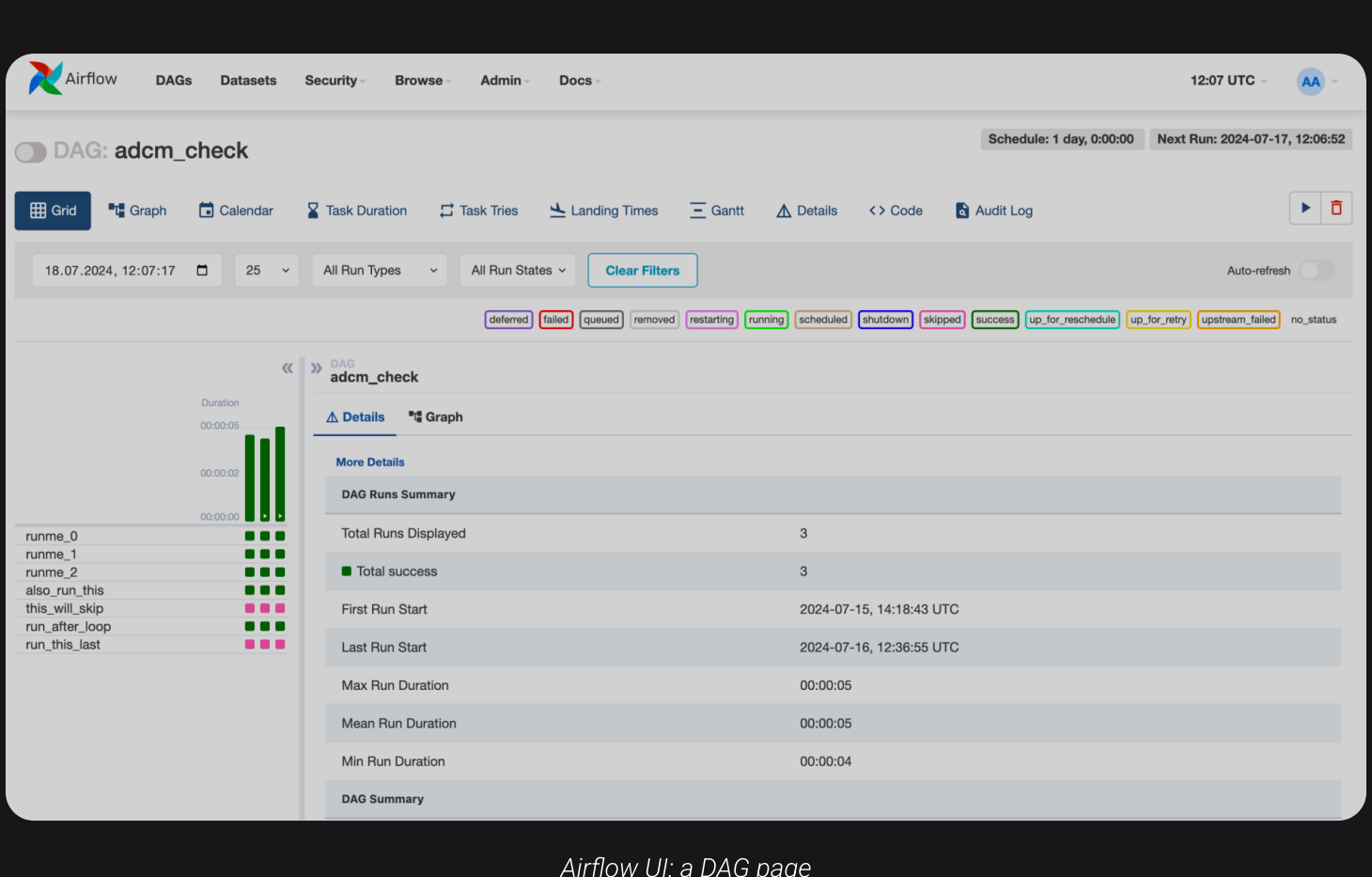
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
adcm_check example example2	airflow	3	1 day, 0:00:00	2024-07-16, 12:36:54	2024-07-21, 08:57:55	5 2		
dataset_consumes_1 consumes dataset-scheduled	airflow		Dataset		On s3://dag1/output_1.txt			
dataset_consumes_1_and_2 consumes dataset-scheduled	airflow		Dataset		0 of 2 datasets updated			
dataset_consumes_1_never_scheduled consumes dataset-scheduled	airflow		Dataset		0 of 2 datasets updated			
dataset_consumes_unknown_never_scheduled dataset-scheduled	airflow		Dataset		0 of 2 datasets updated			
dataset_produces_1 dataset-scheduled produces	airflow		@daily		2024-07-21, 00:00:00			
dataset_produces_2 dataset-scheduled produces	airflow		None					
example_bash_operator example example2	airflow		0 0 * * *		2024-07-21, 00:00:00			
example_branch_datetime_operator example	airflow		@daily		2024-07-21, 00:00:00			
example_branch_datetime_operator_2 example	airflow		@daily		2024-07-21, 00:00:00			
example_branch_datetime_operator_3 example	airflow		@daily		2024-07-21, 00:00:00			
example_branch_dop_operator_v3 example	airflow		*/1 * * * *		2024-07-22, 08:56:00			
example_branch_labels	airflow		@daily		2024-07-21, 00:00:00			
example_branch_operator example example2	airflow		@daily		2024-07-21, 00:00:00			

Airflow UI: home page

Additionally, you can use search or filter the list of DAGs by a tag.

On a DAG page, you can find several representations of the DAG's information:

- ***Grid*** — a chart of DAG runs by duration, task instances, and runs summary. The tasks are represented as squares and the DAGs as bars. To see information for a specific task, click on the corresponding square. There you can find information about Xcoms, logs, and actions related to the task.
- ***Graph*** — a graph of tasks with information about runs.
- ***Calendar*** — a calendar of running and planned DAGs. Each date in the calendar has an indicator of DAG runs' success rate.
- ***Task Duration*** — a graph of the duration for each task.
- ***Task Tries*** — a graph of run attempts for each task.
- ***Landing Times*** — the representation of finish times for each task related to the scheduled finish times.
- ***Gantt*** — the view of planned and actual finish times for tasks.
- ***Details*** — the detailed information on a specific DAG.
- ***Code*** — the DAG's code.
- ***Audit Log*** — the view displays selected events and operations that have been taken on this DAG. The included and excluded events are set in the Airflow configuration.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:07 UTC
AA
Grid
Graph
Calendar
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Audit Log
18.07.2024, 12:07:17
25
All Run Types
All Run States
Clear Filters
Auto-refresh
deferred
failed
queued
removed
restarting
running
scheduled
shutdown
skipped
success
up_for_reschedule
up_for_retry
upstream_failed
no_status
Duration
runme_0
runme_1
runme_2
also_run_this
this_will_skip
run_after_loop
run_this_last
Details
Graph
More Details

DAG: adcm_check

Schedule: 1 day, 0:00:00

Next Run: 2024-07-17, 12:06:52

adcm_check

DAG Runs Summary

Total Runs Displayed	3
Total success	3
First Run Start	2024-07-15, 14:18:43 UTC
Last Run Start	2024-07-16, 12:36:55 UTC
Max Run Duration	00:00:05
Mean Run Duration	00:00:05
Min Run Duration	00:00:04

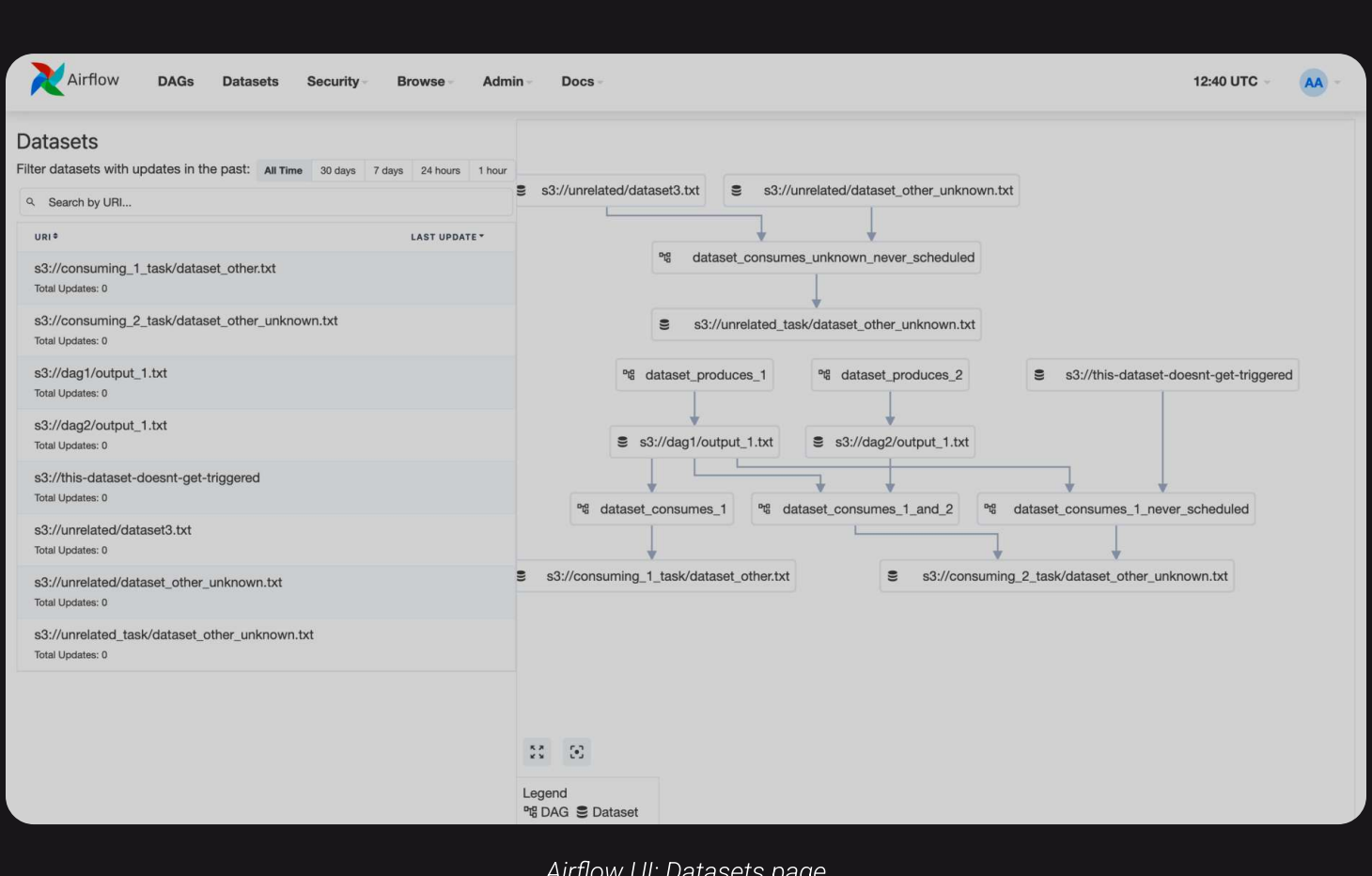
DAG Summary

Airflow UI: a DAG page

Datasets

On the ***Datasets*** page, you can see the list of existing datasets on the left and the graph representation of relationships between datasets and DAGs on the right. When you click on a dataset in the list or the graph, it will highlight the dataset and its connections.

The page provides the ability to filter the list to display the recent history or search for a particular dataset.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:40 UTC
AA
All Time
30 days
7 days
24 hours
1 hour
Search by URI...
URI
LAST UPDATE
s3://unrelated/dataset3.txt
s3://unrelated/dataset_other_unknown.txt
dataset_consumes_unknown_never_scheduled
s3://unrelated_task/dataset_other_unknown.txt
dataset_produces_1
dataset_produces_2
s3://this-dataset-doesnt-get-triggered
s3://dag1/output_1.txt
s3://dag2/output_1.txt
dataset_consumes_1
dataset_consumes_1_and_2
dataset_consumes_1_never_scheduled
s3://consuming_1_task/dataset_other.txt
s3://consuming_2_task/dataset_other_unknown.txt
Legend
DAG
Dataset

Datasets

Filter datasets with updates in the past:

URI	LAST UPDATE
s3://consuming_1_task/dataset_other.txt Total Updates: 0	
s3://consuming_2_task/dataset_other_unknown.txt Total Updates: 0	
s3://dag1/output_1.txt Total Updates: 0	
s3://dag2/output_1.txt Total Updates: 0	
s3://this-dataset-doesnt-get-triggered Total Updates: 0	
s3://unrelated/dataset3.txt Total Updates: 0	
s3://unrelated/dataset_other_unknown.txt Total Updates: 0	
s3://unrelated_task/dataset_other_unknown.txt Total Updates: 0	

Airflow UI: Datasets page

Security

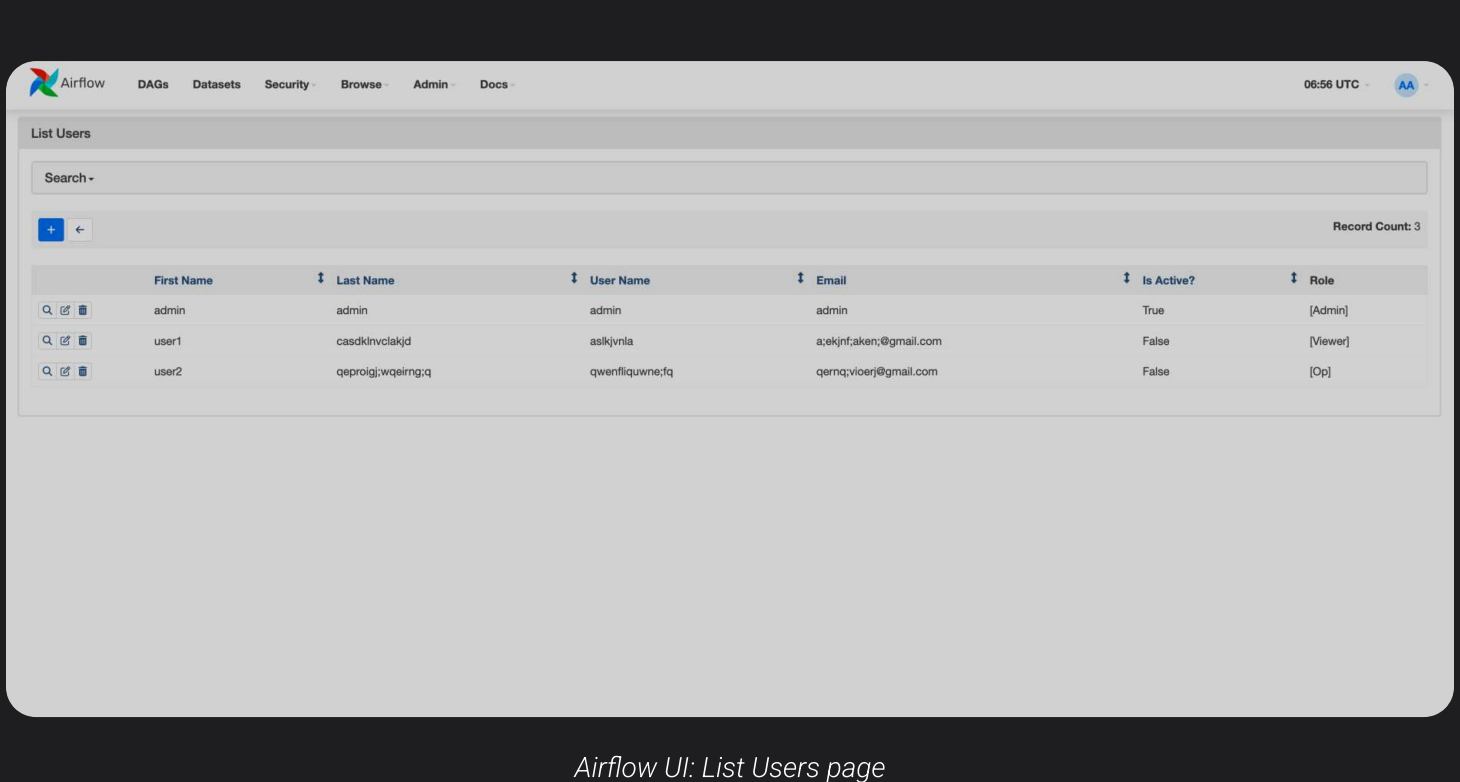
The ***Security*** menu provides access to information about users, roles, security settings, and actions.

List Users

The ***List Users*** page displays a list of Airflow users. Each user has the following traits: first name, last name, user name, email, status, and role.

You can view, edit, or delete a user by selecting the corresponding action.

To add a user, click the plus sign and enter the user's parameters.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
06:56 UTC
AA
Search

List Users

Record Count: 3

First Name	Last Name	User Name	Email	Is Active?	Role
admin	admin	admin	admin	True	[Admin]
user1	casdklnvclakjd	aslkjvnla	a;ekjnf;aken;@gmail.com	False	[Viewer]
user2	qeproigj;wqeirng;q	qwenfliquwne;fq	qernq;vioerj@gmail.com	False	[Op]

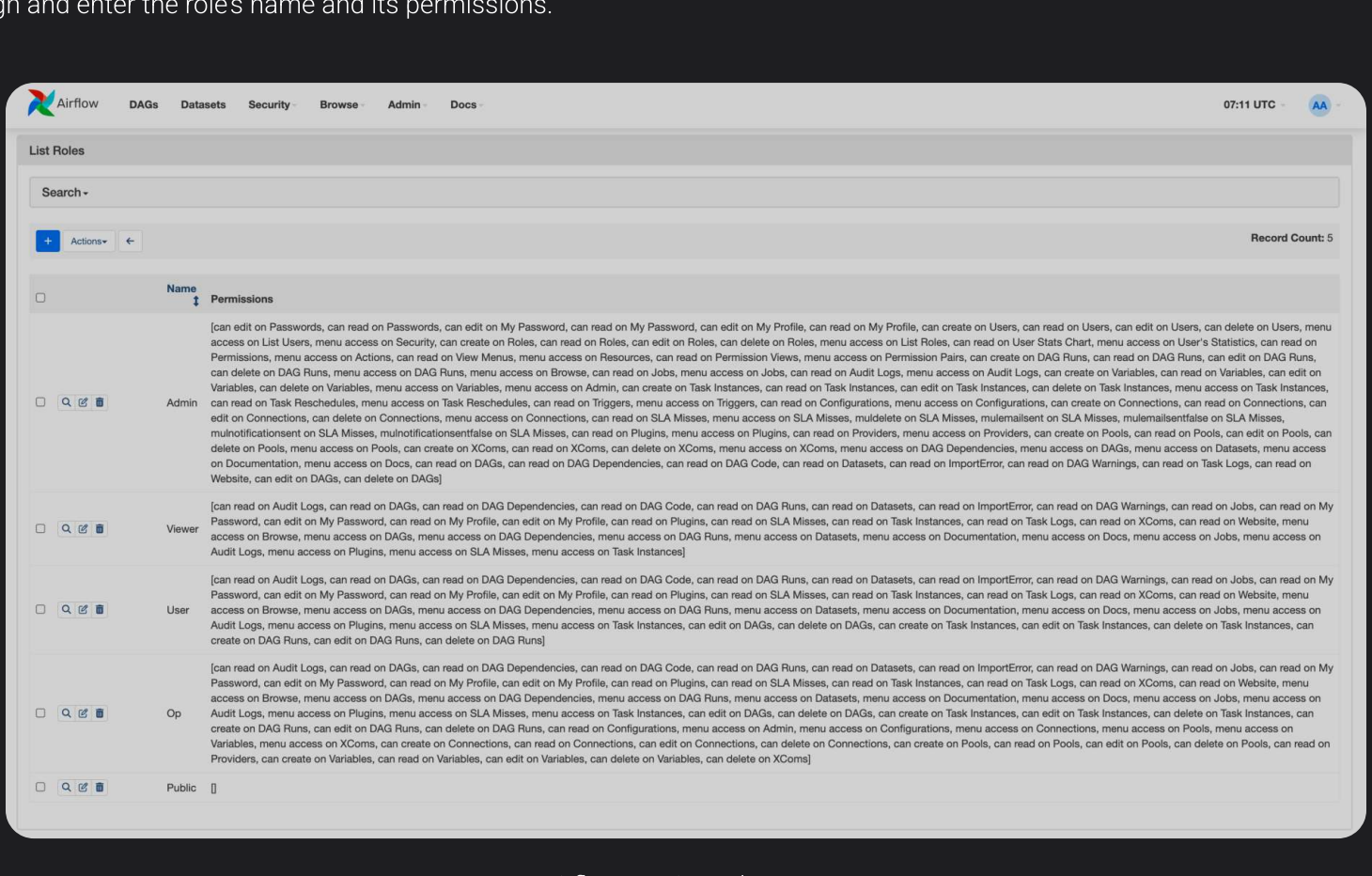
Airflow UI: List Users page

List Roles

The ***List Roles*** page displays a list of user roles. Each role has a name and a list of permissions given to that role.

You can view, edit, or delete a role by selecting the corresponding action.

To add a role, click the plus sign and enter the role's name and its permissions.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
07:11 UTC
AA
Search
Actions

List Roles

Record Count: 5

Name	Permissions
Admin	[can edit on Passwords, can read on Passwords, can edit on My Password, can read on My Password, can edit on My Profile, can read on My Profile, can create on Users, can read on Users, can edit on Users, can delete on Users, menu access on List Users, menu access on Security, can create on Roles, can read on Roles, can edit on Roles, can delete on Roles, menu access on List Roles, can read on User Stats Chart, menu access on User's Statistics, can read on Permissions, menu access on Actions, can read on View Menus, menu access on Resources, can read on Permission Views, menu access on Permission Pairs, can create on DAG Runs, can read on DAG Runs, can edit on DAG Runs, can delete on DAG Runs, menu access on DAG Runs, menu access on Browse, can read on Jobs, menu access on Jobs, can read on Audit Logs, menu access on Audit Logs, can create on Variables, can read on Variables, can edit on Variables, can delete on Variables, menu access on Variables, menu access on Admin, can create on Task Instances, can read on Task Instances, can edit on Task Instances, can delete on Task Instances, menu access on Task Instances, can read on Task Reschedules, menu access on Task Reschedules, can read on Triggers, menu access on Triggers, can read on Configurations, menu access on Configurations, can create on Connections, can read on Connections, can edit on Connections, can delete on Connections, menu access on Connections, can read on SLA Misses, menu access on SLA Misses, muldelete on SLA Misses, mulemailsent on SLA Misses, mulemailsentfalse on SLA Misses, mulnotificationsent on SLA Misses, mulnotificationsentfalse on SLA Misses, can read on Plugins, menu access on Plugins, can read on Providers, menu access on Providers, can create on Pools, can read on Pools, can edit on Pools, can delete on Pools, menu access on Pools, can create on XComs, can read on XComs, can delete on XComs, menu access on XComs, menu access on DAG Dependencies, menu access on DAGs, menu access on Datasets, menu access on Documentation, menu access on Docs, can read on DAGs, can read on DAG Dependencies, can read on DAG Code, can read on Datasets, can read on ImportError, can read on DAG Warnings, can read on Task Logs, can read on Website, can edit on DAGs, can delete on DAGs]
Viewer	[can read on Audit Logs, can read on DAGs, can read on DAG Dependencies, can read on DAG Code, can read on DAG Runs, can read on Datasets, can read on ImportError, can read on DAG Warnings, can read on Jobs, can read on My Password, can edit on My Password, can read on My Profile, can edit on My Profile, can read on Plugins, can read on SLA Misses, can read on Task Instances, can read on Task Logs, can read on XComs, can read on Website, menu access on Browse, menu access on DAGs, menu access on DAG Dependencies, menu access on DAG Runs, menu access on Datasets, menu access on Documentation, menu access on Docs, menu access on Jobs, menu access on Audit Logs, menu access on Plugins, menu access on SLA Misses, menu access on Task Instances]
User	[can read on Audit Logs, can read on DAGs, can read on DAG Dependencies, can read on DAG Code, can read on DAG Runs, can read on Datasets, can read on ImportError, can read on DAG Warnings, can read on Jobs, can read on My Password, can edit on My Password, can read on My Profile, can edit on My Profile, can read on Plugins, can read on SLA Misses, can read on Task Instances, can read on Task Logs, can read on XComs, can read on Website, menu access on Browse, menu access on DAGs, menu access on DAG Dependencies, menu access on DAG Runs, menu access on Datasets, menu access on Documentation, menu access on Docs, menu access on Jobs, menu access on Audit Logs, menu access on Plugins, menu access on SLA Misses, menu access on Task Instances, can edit on DAGs, can delete on DAGs, can create on Task Instances, can edit on Task Instances, can delete on Task Instances, can create on DAG Runs, can edit on DAG Runs, can delete on DAG Runs]
Op	[can read on Audit Logs, can read on DAGs, can read on DAG Dependencies, can read on DAG Code, can read on DAG Runs, can read on Datasets, can read on ImportError, can read on DAG Warnings, can read on Jobs, can read on My Password, can edit on My Password, can read on My Profile, can edit on My Profile, can read on Plugins, can read on SLA Misses, can read on Task Instances, can read on Task Logs, can read on XComs, can read on Website, menu access on Browse, menu access on DAGs, menu access on DAG Dependencies, menu access on DAG Runs, menu access on Datasets, menu access on Documentation, menu access on Docs, menu access on Jobs, menu access on Audit Logs, menu access on Plugins, menu access on SLA Misses, menu access on Task Instances, can edit on DAGs, can delete on DAGs, can create on Task Instances, can edit on Task Instances, can delete on Task Instances, can create on DAG Runs, can edit on DAG Runs, can delete on DAG Runs, can read on Configurations, menu access on Admin, menu access on Configurations, menu access on Connections, menu access on Pools, menu access on Variables, menu access on XComs, can create on Connections, can read on Connections, can edit on Connections, can delete on Connections, can create on Pools, can read on Pools, can edit on Pools, can delete on Pools, can read on Providers, can create on Variables, can read on Variables, can edit on Variables, can delete on Variables, can delete on XComs]
Public	[]

Airflow UI: List Roles page

User's statistics

The ***User's statistics*** page displays a chart of users' login counts. You can change the view by selecting either login counts or failed login attempts.

Actions, Resources, Permissions

- The ***Actions*** page displays a list of actions that can be granted to a user.
- The ***Resources*** page displays a list of resources to which you can give access.
- The ***Permissions*** page displays a list of actions that can be granted for a particular resource.

Browse

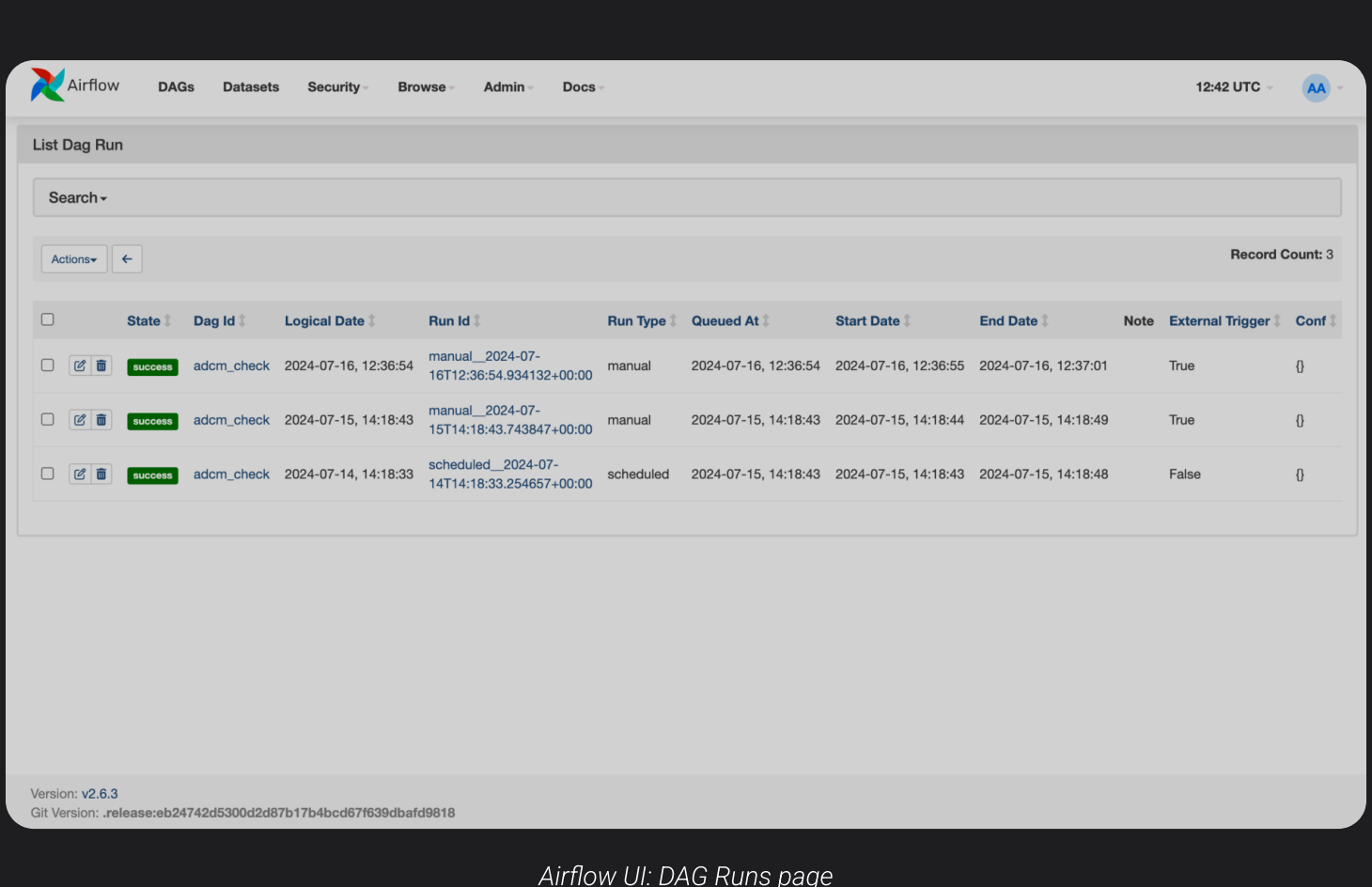
The ***Browse*** menu provides a set of pages with lists that allow you to control several entities at once.

DAG Runs

The ***DAG Runs*** page displays a list of all DAG runs.

You can manage several DAGs at once by picking the necessary DAG runs and selecting a common action for them.

By clicking on the ***Dag Id*** or ***Run Id*** of a specific DAG run, you can navigate to that DAG's page.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:42 UTC
AA
Search
Actions

List Dag Run

Record Count: 3

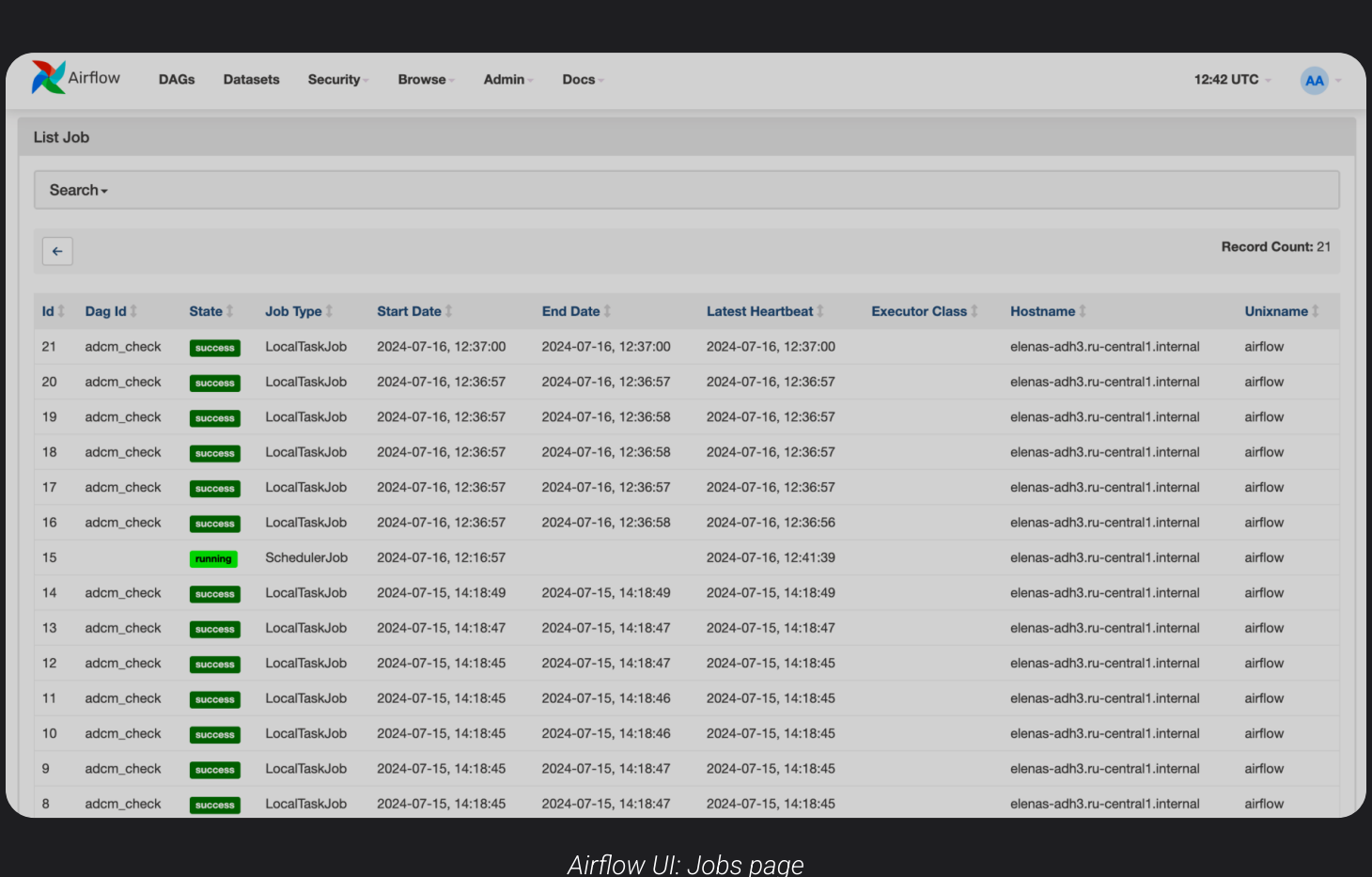
State	Dag Id	Logical Date	Run Id	Run Type	Queued At	Start Date	End Date	Note	External Trigger	Conf
success	adcm_check	2024-07-16, 12:36:54	manual__2024-07-16T12:36:54.934132+00:00	manual	2024-07-16, 12:36:54	2024-07-16, 12:36:55	2024-07-16, 12:37:01		True	{}
success	adcm_check	2024-07-15, 14:18:43	manual__2024-07-15T14:18:43.743847+00:00	manual	2024-07-15, 14:18:43	2024-07-15, 14:18:44	2024-07-15, 14:18:49		True	{}
success	adcm_check	2024-07-14, 14:18:33	scheduled__2024-07-14T14:18:33.254657+00:00	scheduled	2024-07-15, 14:18:43	2024-07-15, 14:18:43	2024-07-15, 14:18:48		False	{}

Version: v2.6.3
Git Version: .release:eb24742d5300d2d87b17b4bcd67f639dbafd9818

Airflow UI: DAG Runs page

Jobs

The ***Jobs*** page displays a list of all jobs with information about their DAGs, current state, start and finish dates, latest heartbeat, executor class, hostname, and unixname.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:42 UTC
AA
Search

List Job

Record Count: 21

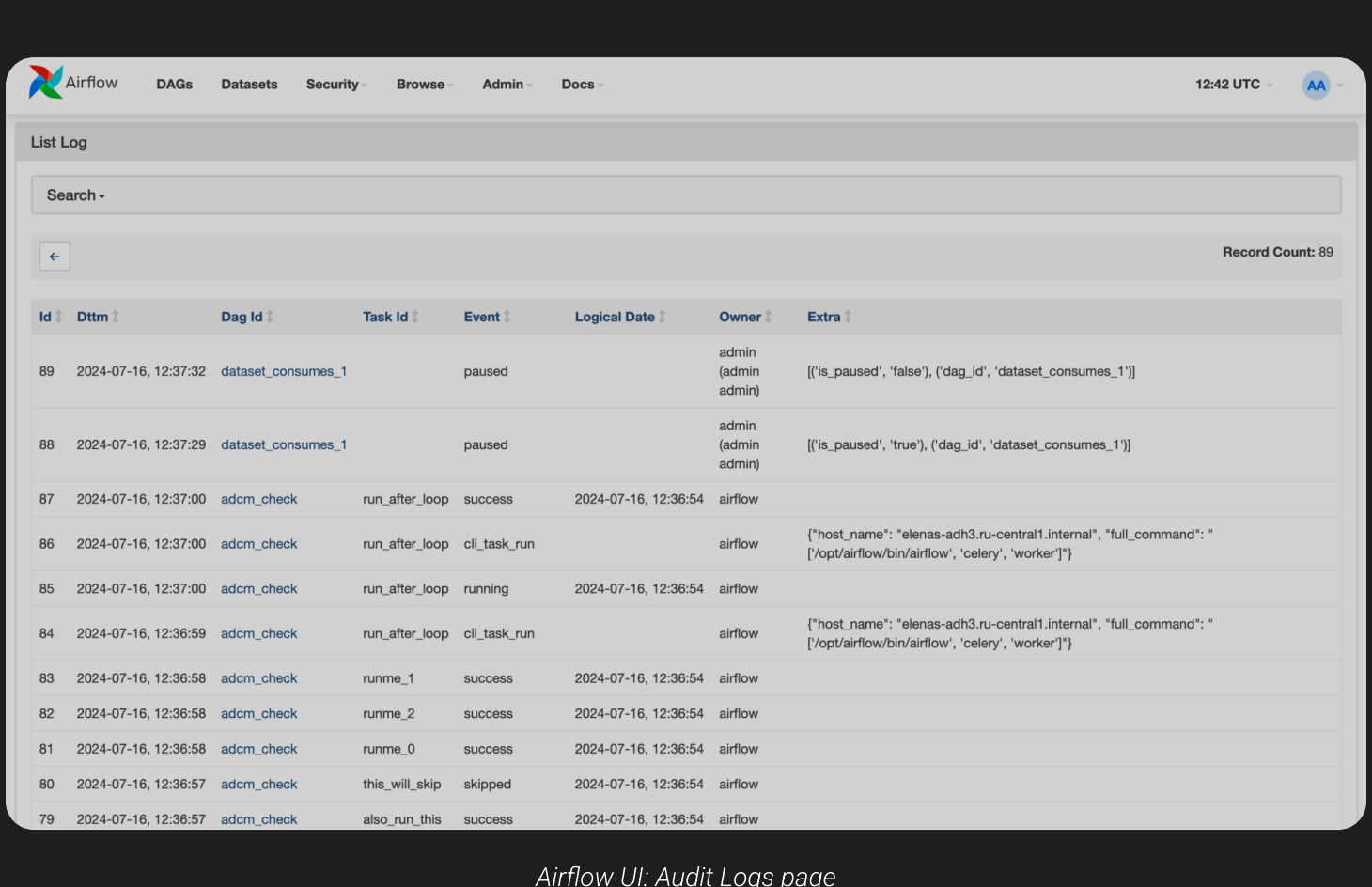
Id	Dag Id	State	Job Type	Start Date	End Date	Latest Heartbeat	Executor Class	Hostname	Unixname
21	adcm_check	success	LocalTaskJob	2024-07-16, 12:37:00	2024-07-16, 12:37:00	2024-07-16, 12:37:00		elenas-adh3.ru-central1.internal	airflow
20	adcm_check	success	LocalTaskJob	2024-07-16, 12:36:57	2024-07-16, 12:36:57	2024-07-16, 12:36:57		elenas-adh3.ru-central1.internal	airflow
19	adcm_check	success	LocalTaskJob	2024-07-16, 12:36:57	2024-07-16, 12:36:58	2024-07-16, 12:36:57		elenas-adh3.ru-central1.internal	airflow
18	adcm_check	success	LocalTaskJob	2024-07-16, 12:36:57	2024-07-16, 12:36:58	2024-07-16, 12:36:57		elenas-adh3.ru-central1.internal	airflow
17	adcm_check	success	LocalTaskJob	2024-07-16, 12:36:57	2024-07-16, 12:36:57	2024-07-16, 12:36:57		elenas-adh3.ru-central1.internal	airflow
16	adcm_check	success	LocalTaskJob	2024-07-16, 12:36:57	2024-07-16, 12:36:58	2024-07-16, 12:36:56		elenas-adh3.ru-central1.internal	airflow
15		running	SchedulerJob	2024-07-16, 12:16:57		2024-07-16, 12:41:39		elenas-adh3.ru-central1.internal	airflow
14	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:49	2024-07-15, 14:18:49	2024-07-15, 14:18:49		elenas-adh3.ru-central1.internal	airflow
13	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:47	2024-07-15, 14:18:47	2024-07-15, 14:18:47		elenas-adh3.ru-central1.internal	airflow
12	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:45	2024-07-15, 14:18:47	2024-07-15, 14:18:45		elenas-adh3.ru-central1.internal	airflow
11	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:45	2024-07-15, 14:18:46	2024-07-15, 14:18:45		elenas-adh3.ru-central1.internal	airflow
10	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:45	2024-07-15, 14:18:46	2024-07-15, 14:18:45		elenas-adh3.ru-central1.internal	airflow
9	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:45	2024-07-15, 14:18:47	2024-07-15, 14:18:45		elenas-adh3.ru-central1.internal	airflow
8	adcm_check	success	LocalTaskJob	2024-07-15, 14:18:45	2024-07-15, 14:18:47	2024-07-15, 14:18:45		elenas-adh3.ru-central1.internal	airflow

Airflow UI: Jobs page

Audit Logs

In Airflow, audit logs are used to track user actions and system events that occur during the execution of DAGs and tasks.

The ***Audit Logs*** page provides information about the events triggered by DAG runs or users.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:42 UTC
AA
Search

List Log

Record Count: 89

Id	Dttm	Dag Id	Task Id	Event	Logical Date	Owner	Extra
89	2024-07-16, 12:37:32	dataset_consumes_1		paused		admin (admin admin)	[('is_paused', 'false'), ('dag_id', 'dataset_consumes_1')]
88	2024-07-16, 12:37:29	dataset_consumes_1		paused		admin (admin admin)	[('is_paused', 'true'), ('dag_id', 'dataset_consumes_1')]
87	2024-07-16, 12:37:00	adcm_check	run_after_loop	success	2024-07-16, 12:36:54	airflow	
86	2024-07-16, 12:37:00	adcm_check	run_after_loop	cli_task_run		airflow	{"host_name": "elenas-adh3.ru-central1.internal", "full_command": "['/opt/airflow/bin/airflow', 'celery', 'worker']"}
85	2024-07-16, 12:37:00	adcm_check	run_after_loop	running	2024-07-16, 12:36:54	airflow	
84	2024-07-16, 12:36:59	adcm_check	run_after_loop	cli_task_run		airflow	{"host_name": "elenas-adh3.ru-central1.internal", "full_command": "['/opt/airflow/bin/airflow', 'celery', 'worker']"}
83	2024-07-16, 12:36:58	adcm_check	runme_1	success	2024-07-16, 12:36:54	airflow	
82	2024-07-16, 12:36:58	adcm_check	runme_2	success	2024-07-16, 12:36:54	airflow	
81	2024-07-16, 12:36:58	adcm_check	runme_0	success	2024-07-16, 12:36:54	airflow	
80	2024-07-16, 12:36:57	adcm_check	this_will_skip	skipped	2024-07-16, 12:36:54	airflow	
79	2024-07-16, 12:36:57	adcm_check	also_run_this	success	2024-07-16, 12:36:54	airflow	

Airflow UI: Audit Logs page

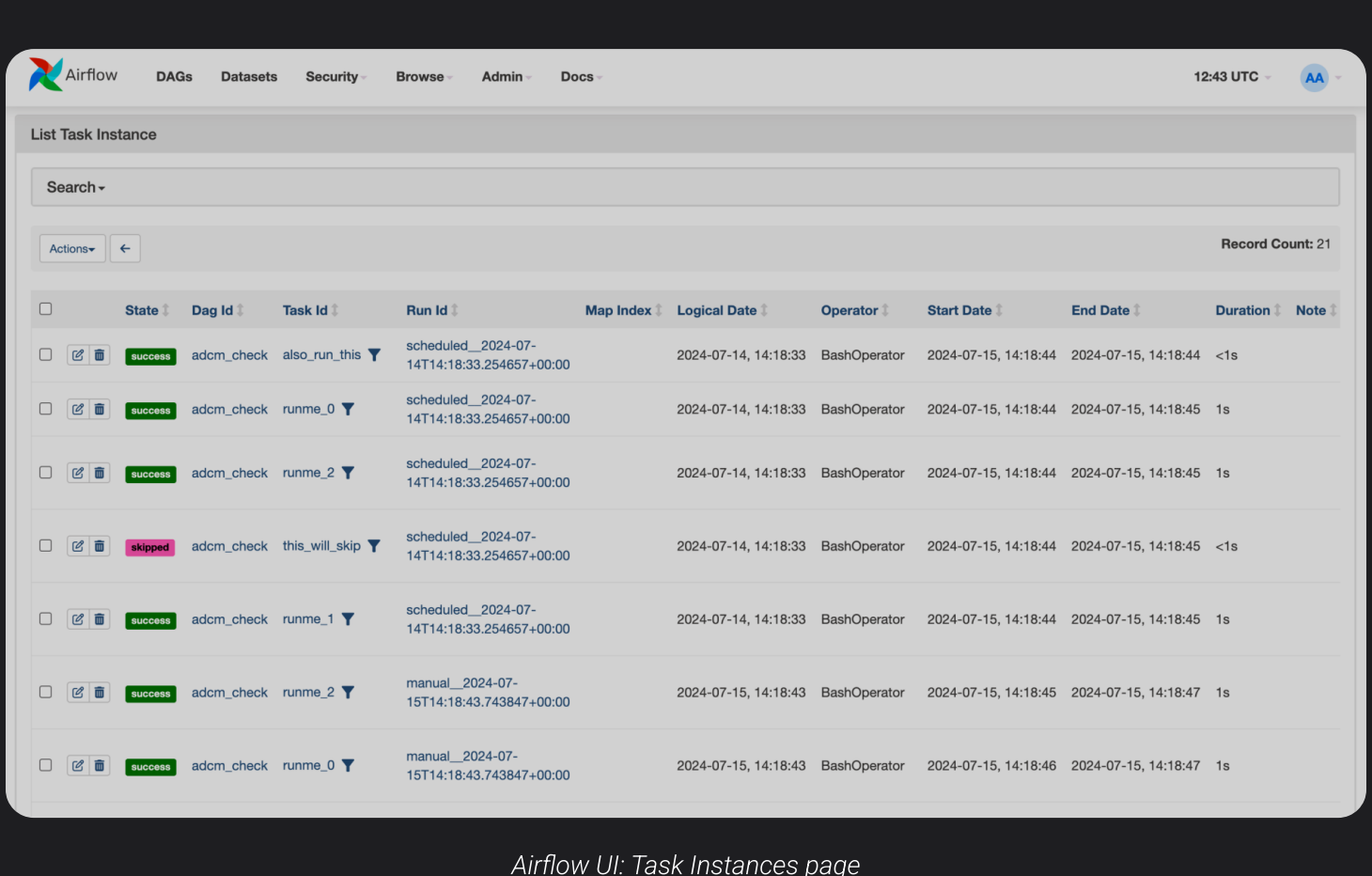
Task Instances and Task Reschedules

The ***Task Instances*** page provides information about each task: their DAGs, start and finish times, and more.

You can manage several tasks at once by picking the necessary tasks and selecting a common action for them.

By clicking on the ***Dag Id***, ***Task Id*** or ***Run Id*** of a specific task, you can navigate to that task's DAG page.

The ***Task Reschedules*** page displays a list of rescheduled tasks and information about them.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:43 UTC
AA
Search
Actions

List Task Instance

Record Count: 21

State	Dag Id	Task Id	Run Id	Map Index	Logical Date	Operator	Start Date	End Date	Duration	Note
success	adcm_check	also_run_this	scheduled__2024-07-14T14:18:33.254657+00:00		2024-07-14, 14:18:33	BashOperator	2024-07-15, 14:18:44	2024-07-15, 14:18:44	<1s	
success	adcm_check	runme_0	scheduled__2024-07-14T14:18:33.254657+00:00		2024-07-14, 14:18:33	BashOperator	2024-07-15, 14:18:44	2024-07-15, 14:18:45	1s	
success	adcm_check	runme_2	scheduled__2024-07-14T14:18:33.254657+00:00		2024-07-14, 14:18:33	BashOperator	2024-07-15, 14:18:44	2024-07-15, 14:18:45	1s	
skipped	adcm_check	this_will_skip	scheduled__2024-07-14T14:18:33.254657+00:00		2024-07-14, 14:18:33	BashOperator	2024-07-15, 14:18:44	2024-07-15, 14:18:45	<1s	
success	adcm_check	runme_1	scheduled__2024-07-14T14:18:33.254657+00:00		2024-07-14, 14:18:33	BashOperator	2024-07-15, 14:18:44	2024-07-15, 14:18:45	1s	
success	adcm_check	runme_2	manual__2024-07-15T14:18:43.743847+00:00		2024-07-15, 14:18:43	BashOperator	2024-07-15, 14:18:45	2024-07-15, 14:18:47	1s	
success	adcm_check	runme_0	manual__2024-07-15T14:18:43.743847+00:00		2024-07-15, 14:18:43	BashOperator	2024-07-15, 14:18:46	2024-07-15, 14:18:47	1s	

Airflow UI: Task Instances page

Triggers

The ***Triggers*** page displays a list of existing triggers.

For more information about triggers, see the Airflow documentation.

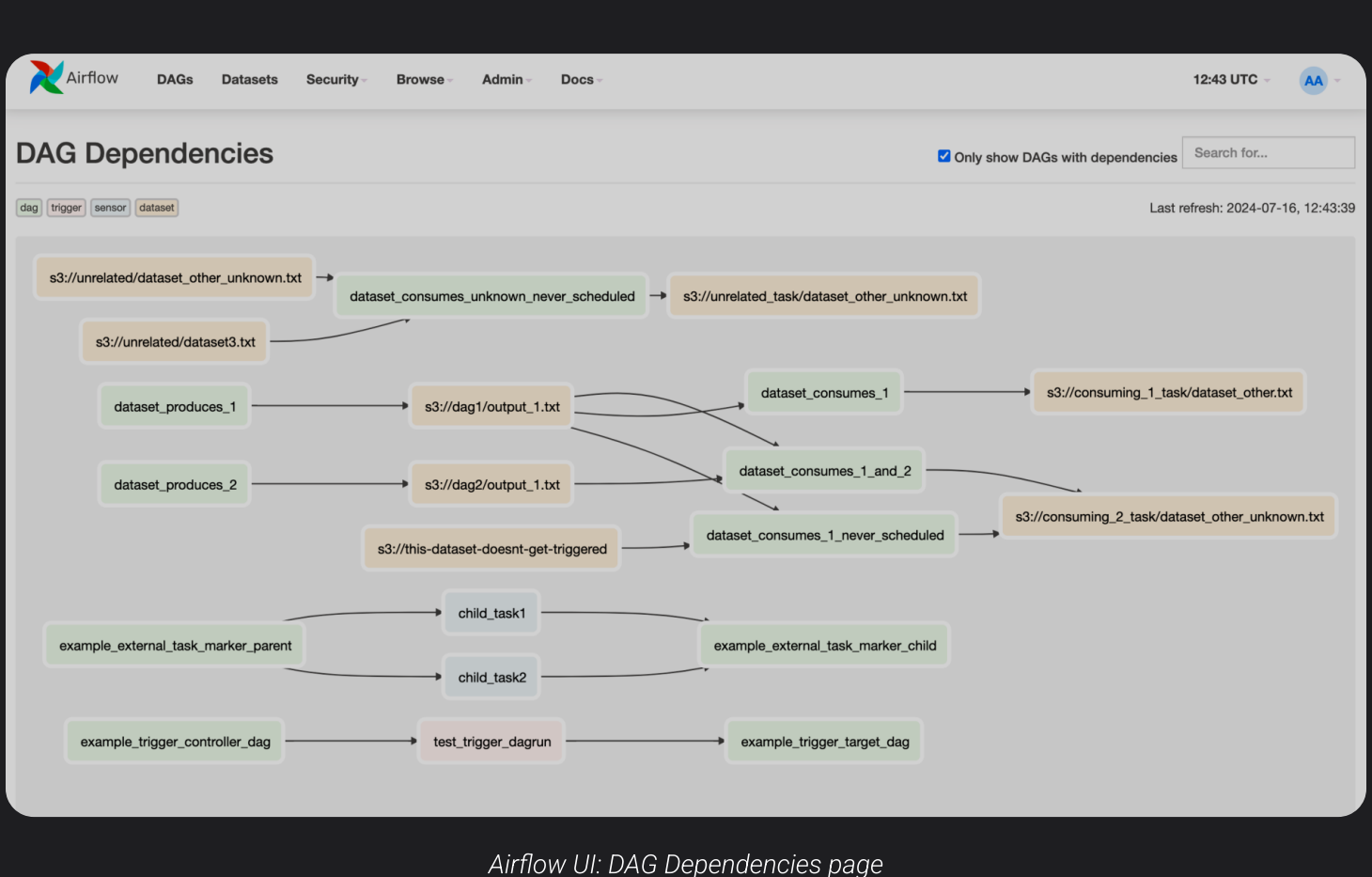
SLA Misses

An SLA, or service level agreement, is an expectation for the maximum time a task should be completed relative to the DAG Run start time. If a task takes longer than this, it will appear on the ***SLA Misses*** page.

For more information about the SLAs, see the Airflow documentation.

DAG Dependencies

The ***DAG Dependencies*** page shows a graph of connections between DAGs, triggers, sensors, and datasets.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:43 UTC
AA
DAG Dependencies
Only show DAGs with dependencies
Search for...
dag
trigger
sensor
dataset
Last refresh: 2024-07-16, 12:43:39
s3://unrelated/dataset_other_unknown.txt
dataset_consumes_unknown_never_scheduled
s3://unrelated_task/dataset_other_unknown.txt
s3://unrelated/dataset3.txt
dataset_produces_1
s3://dag1/output_1.txt
dataset_consumes_1
s3://consuming_1_task/dataset_other.txt
dataset_produces_2
s3://dag2/output_1.txt
dataset_consumes_1_and_2
s3://consuming_2_task/dataset_other_unknown.txt
s3://this-dataset-doesnt-get-triggered
dataset_consumes_1_never_scheduled
child_task1
example_external_task_marker_parent
example_external_task_marker_child
child_task2
example_trigger_controller_dag
test_trigger_dagrun
example_trigger_target_dag

Airflow UI: DAG Dependencies page

Admin

The ***Admin*** menu provides resource control actions and information about the Airflow configuration.

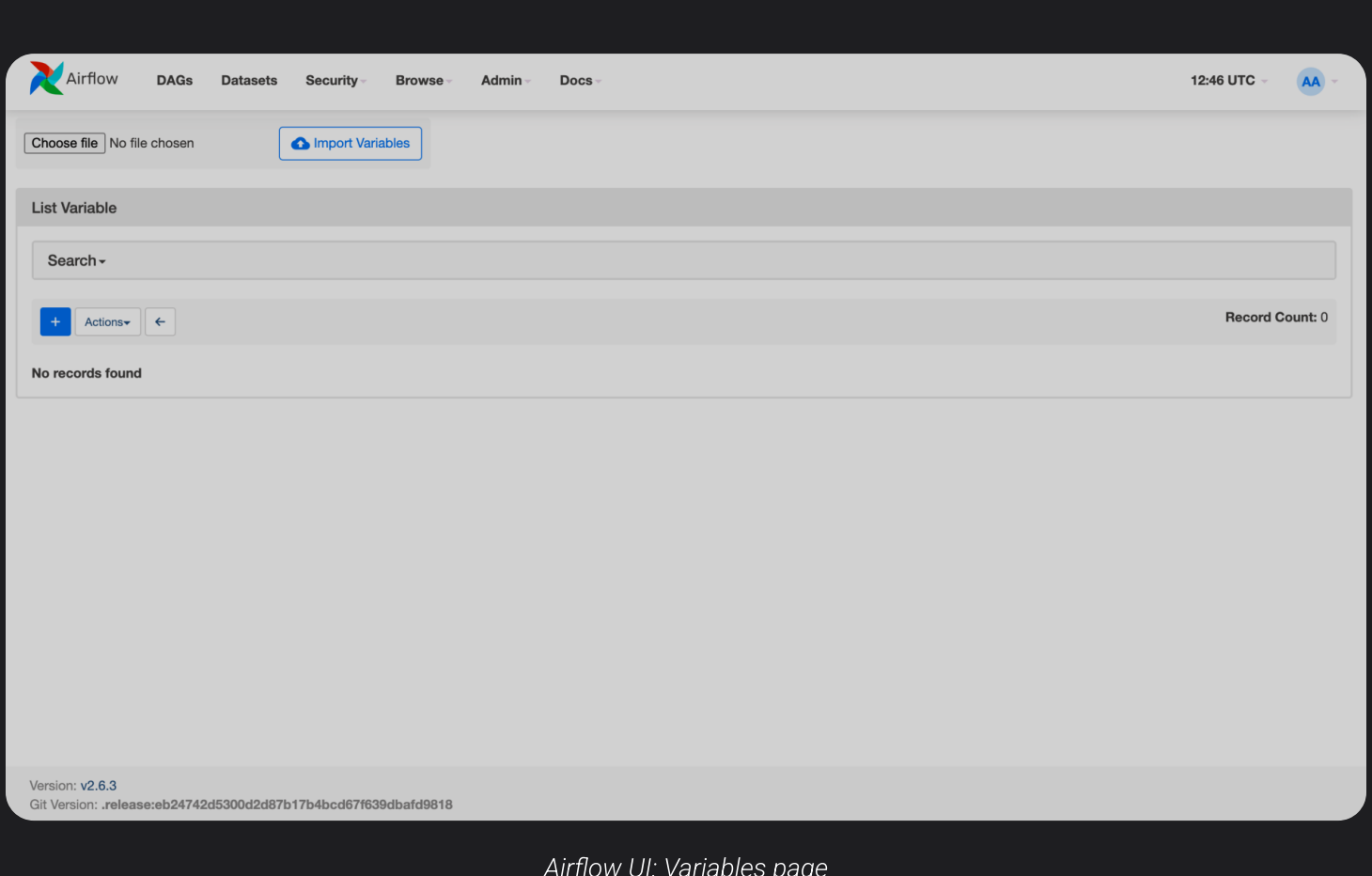
Variables

A variable in Airflow is a key/value pair that expects a string value. On this page, you can create variables that can be used in DAGs and task code.

By default, the value of a variable will be hidden if the key contains the following words: `password`, `secret`, `passwd`, `authorization`, `api_key`, `apikey`, `access_token`. If you don't want the value to be hidden, you can configure the variable to show plain text.

To add a single variable, click the plus sign and enter the variable key/value.

To upload several variables at once, click ***Choose file***, upload a variables file, and click ***Import Variables***.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:46 UTC
AA
Choose file
No file chosen
Import Variables
Search
Actions

List Variable

Record Count: 0

No records found

Version: v2.6.3
Git Version: .release:eb24742d5300d2d87b17b4bcd67f639dbafd9818

Airflow UI: Variables page

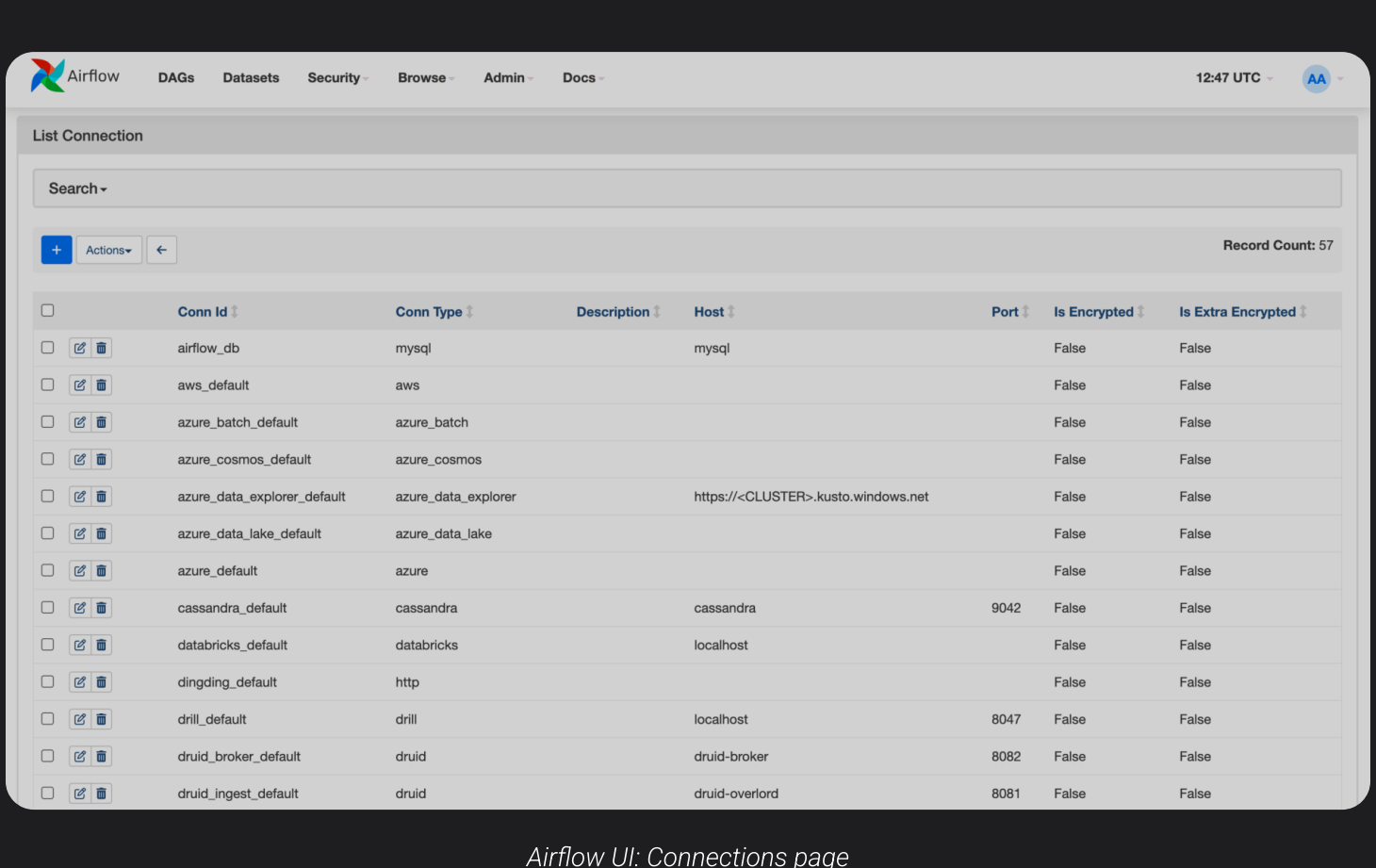
Configurations

The page displays the table with the configuration of your Airflow instance.

Connections

The connections enable integration of Airflow with other services. This page displays a list of created connections.

To create a connection, click the plus sign and enter the connection parameters. After the connection information is saved, you can check the connection by clicking ***Test***.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:47 UTC
AA
Search
Actions

List Connection

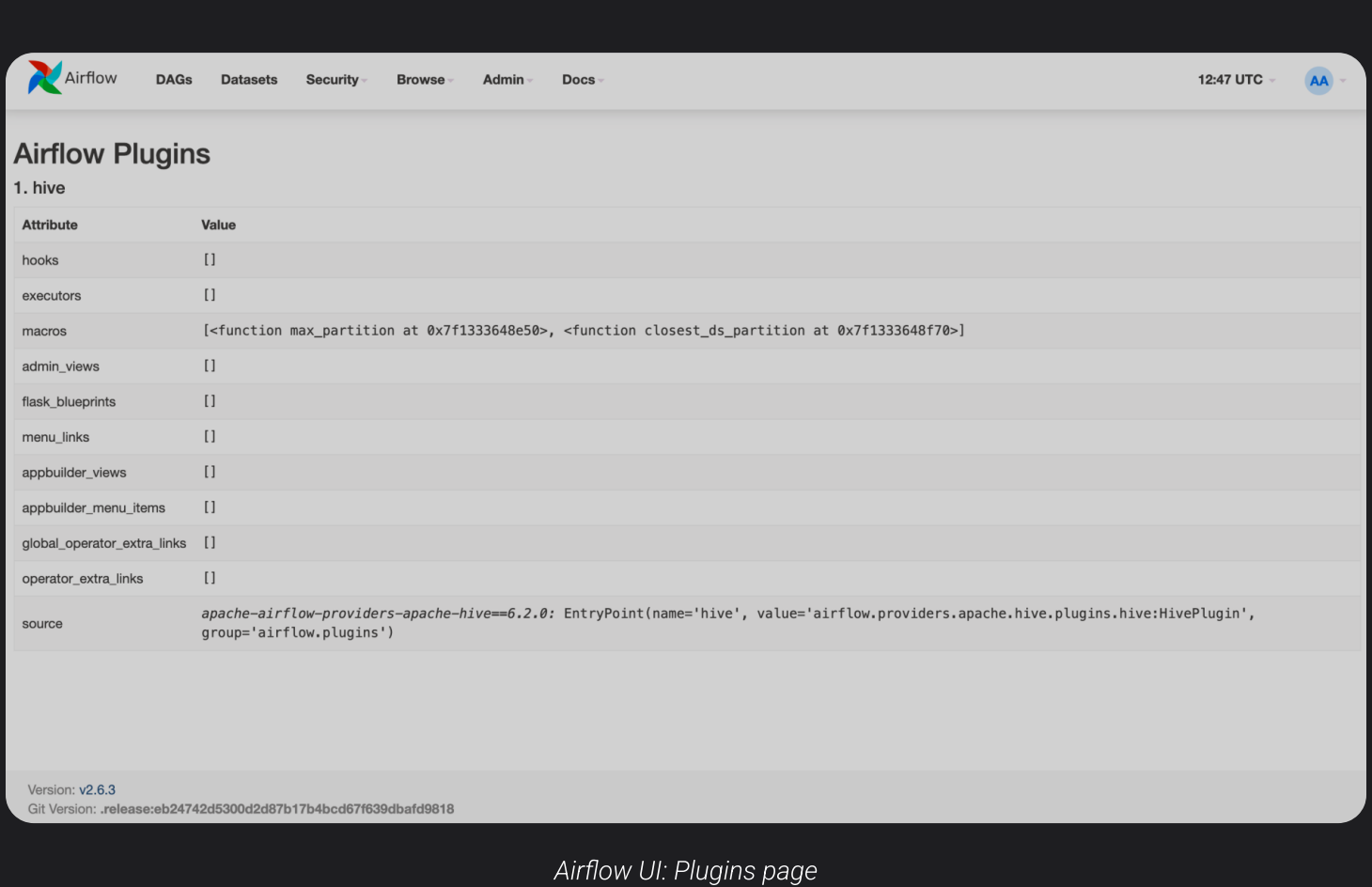
Record Count: 57

Conn Id	Conn Type	Description	Host	Port	Is Encrypted	Is Extra Encrypted
airflow_db	mysql		mysql		False	False
aws_default	aws				False	False
azure_batch_default	azure_batch				False	False
azure_cosmos_default	azure_cosmos				False	False
azure_data_explorer_default	azure_data_explorer		https://<CLUSTER>.kusto.windows.net		False	False
azure_data_lake_default	azure_data_lake				False	False
azure_default	azure				False	False
cassandra_default	cassandra		cassandra	9042	False	False
databricks_default	databricks		localhost		False	False
dingding_default	http				False	False
drill_default	drill		localhost	8047	False	False
druid_broker_default	druid		druid-broker	8082	False	False
druid_ingest_default	druid		druid-overlord	8081	False	False

Airflow UI: Connections page

Plugins

This page provides a list of available plugins. In order to add a plugin, you have to add a connection first.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:47 UTC
AA

Airflow Plugins

1. hive

Attribute	Value
hooks	[]
executors	[]
macros	[<function max_partition at 0x7f1333648e50>, <function closest_ds_partition at 0x7f1333648f70>]
admin_views	[]
flask_blueprints	[]
menu_links	[]
appbuilder_views	[]
appbuilder_menu_items	[]
global_operator_extra_links	[]
operator_extra_links	[]
source	*apache-airflow-providers-apache-hive==6.2.0:* EntryPoint(name='hive', value='airflow.providers.apache.hive.plugins.hive:HivePlugin', group='airflow.plugins')

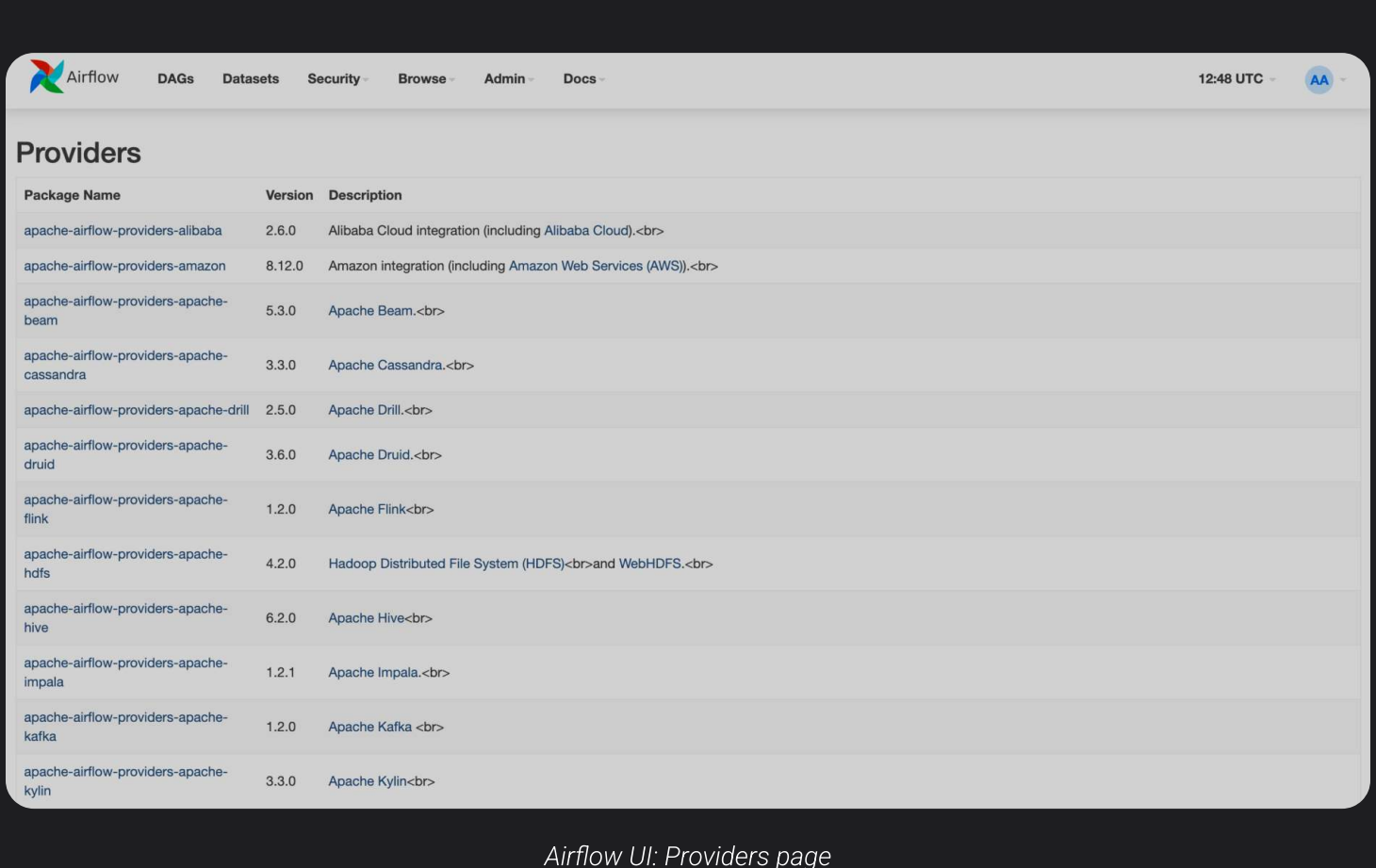
Version: v2.6.3
Git Version: .release:eb24742d5300d2d87b17b4bcd67f639dbafd9818

Airflow UI: Plugins page

Providers

This page displays available providers for integration with other services.

By clicking on the provider's name, you can navigate to the Airflow documentation page for that provider. The links in the description column lead to the services' official web pages.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:48 UTC
AA

Providers

Package Name	Version	Description
apache-airflow-providers-alibaba	2.6.0	Alibaba Cloud integration (including Alibaba Cloud).
apache-airflow-providers-amazon	8.12.0	Amazon integration (including Amazon Web Services (AWS)).
apache-airflow-providers-apache-beam	5.3.0	Apache Beam.
apache-airflow-providers-apache-cassandra	3.3.0	Apache Cassandra.
apache-airflow-providers-apache-drill	2.5.0	Apache Drill.
apache-airflow-providers-apache-druid	3.6.0	Apache Druid.
apache-airflow-providers-apache-flink	1.2.0	Apache Flink
apache-airflow-providers-apache-hdfs	4.2.0	Hadoop Distributed File System (HDFS) and WebHDFS.
apache-airflow-providers-apache-hive	6.2.0	Apache Hive
apache-airflow-providers-apache-impala	1.2.1	Apache Impala.
apache-airflow-providers-apache-kafka	1.2.0	Apache Kafka
apache-airflow-providers-apache-kylin	3.3.0	Apache Kylin

Airflow UI: Providers page

Pools

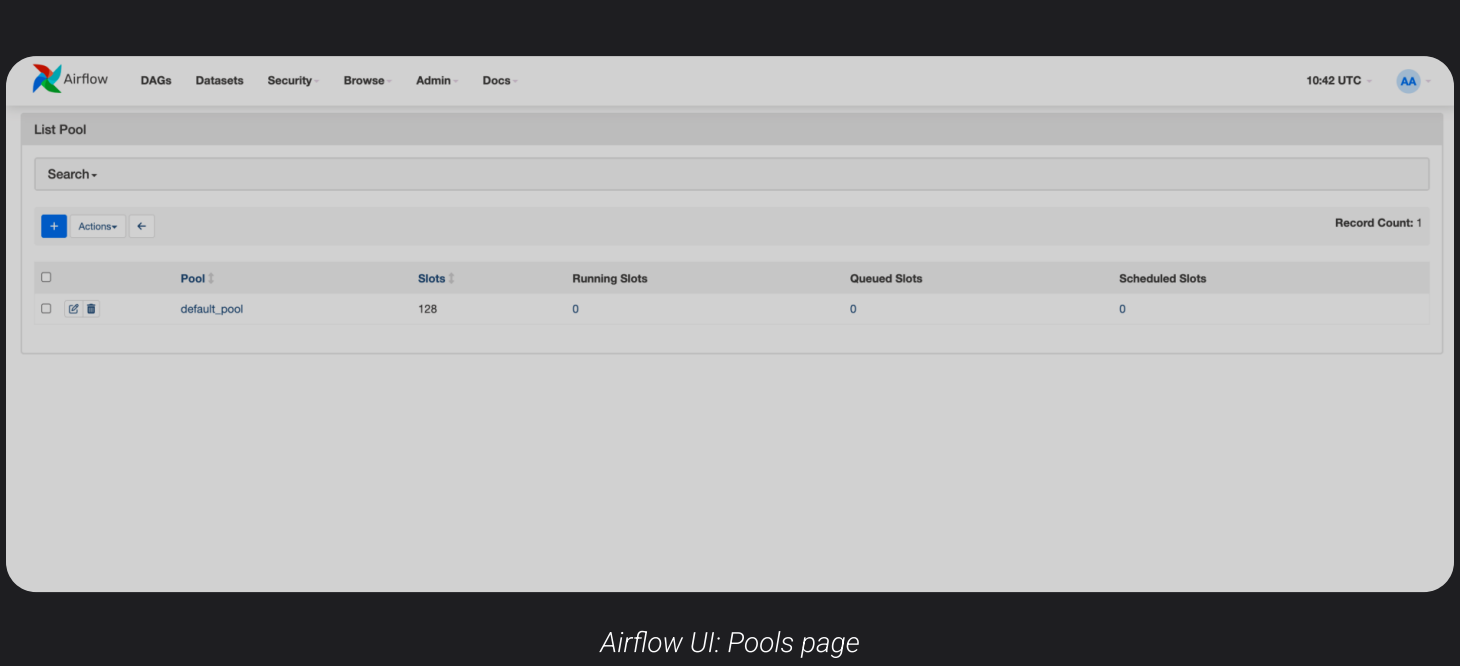
This page displays a list of existing pools. In Airflow, the pools allow you to limit the concurrency of a set of tasks. By default, all tasks run through a default pool.

Each pool has the following parameters:

- ***Pool*** — the name of the pool. By clicking on a pool's name, you can navigate to the list of tasks that belong to that pool.
- ***Slots*** — the amount of tasks a pool can have.
- ***Running Slots*** — the number of busy slots. By clicking on the amount of slots, you can navigate to the list of tasks that occupy those slots.
- ***Queued Slots*** — the number of reserved slots for tasks in the queue. By clicking on the amount of slots, you can navigate to the list of tasks that occupy those slots.
- ***Scheduled Slots*** — the number of slots required for the scheduled tasks. By clicking on the amount of slots, you can navigate to the list of tasks that occupy those slots.

To create a pool, click the plus sign and enter the pool parameters.

You can edit or delete a pool by selecting the corresponding action.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
10:42 UTC
AA
Search
Actions

List Pool

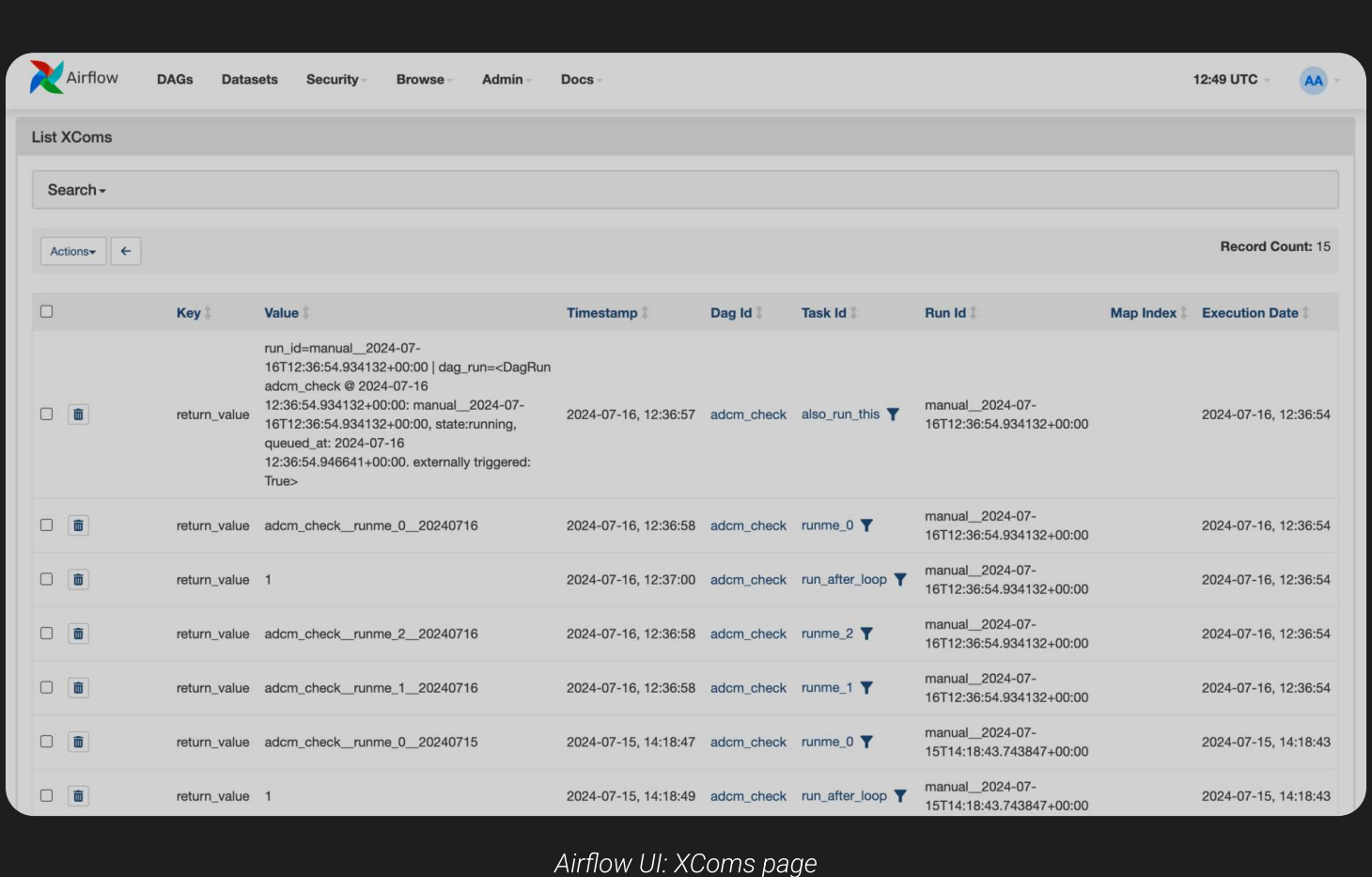
Record Count: 1

Pool	Slots	Running Slots	Queued Slots	Scheduled Slots
default_pool	128	0	0	0

Airflow UI: Pools page

XComs

In Airflow, XComs are tools that enable sharing data between tasks. On this page, you can see a list of available XComs.


Airflow
DAGs
Datasets
Security
Browse
Admin
Docs
12:49 UTC
AA
Search
Actions

List XComs

Record Count: 15

Key	Value	Timestamp	Dag Id	Task Id	Run Id	Map Index	Execution Date
return_value	run_id=manual__2024-07-16T12:36:54.934132+00:00 | dag_run=<DagRun adcm_check @ 2024-07-16 12:36:54.934132+00:00: manual__2024-07-16T12:36:54.934132+00:00, state:running, queued_at: 2024-07-16 12:36:54.946641+00:00. externally triggered: True>	2024-07-16, 12:36:57	adcm_check	also_run_this	manual__2024-07-16T12:36:54.934132+00:00		2024-07-16, 12:36:54
return_value	adcm_check__runme_0__20240716	2024-07-16, 12:36:58	adcm_check	runme_0	manual__2024-07-16T12:36:54.934132+00:00		2024-07-16, 12:36:54
return_value	1	2024-07-16, 12:37:00	adcm_check	run_after_loop	manual__2024-07-16T12:36:54.934132+00:00		2024-07-16, 12:36:54
return_value	adcm_check__runme_2__20240716	2024-07-16, 12:36:58	adcm_check	runme_2	manual__2024-07-16T12:36:54.934132+00:00		2024-07-16, 12:36:54
return_value	adcm_check__runme_1__20240716	2024-07-16, 12:36:58	adcm_check	runme_1	manual__2024-07-16T12:36:54.934132+00:00		2024-07-16, 12:36:54
return_value	adcm_check__runme_0__20240715	2024-07-15, 14:18:47	adcm_check	runme_0	manual__2024-07-15T14:18:43.743847+00:00		2024-07-15, 14:18:43
return_value	1	2024-07-15, 14:18:49	adcm_check	run_after_loop	manual__2024-07-15T14:18:43.743847+00:00		2024-07-15, 14:18:43

Airflow UI: XComs page

Docs

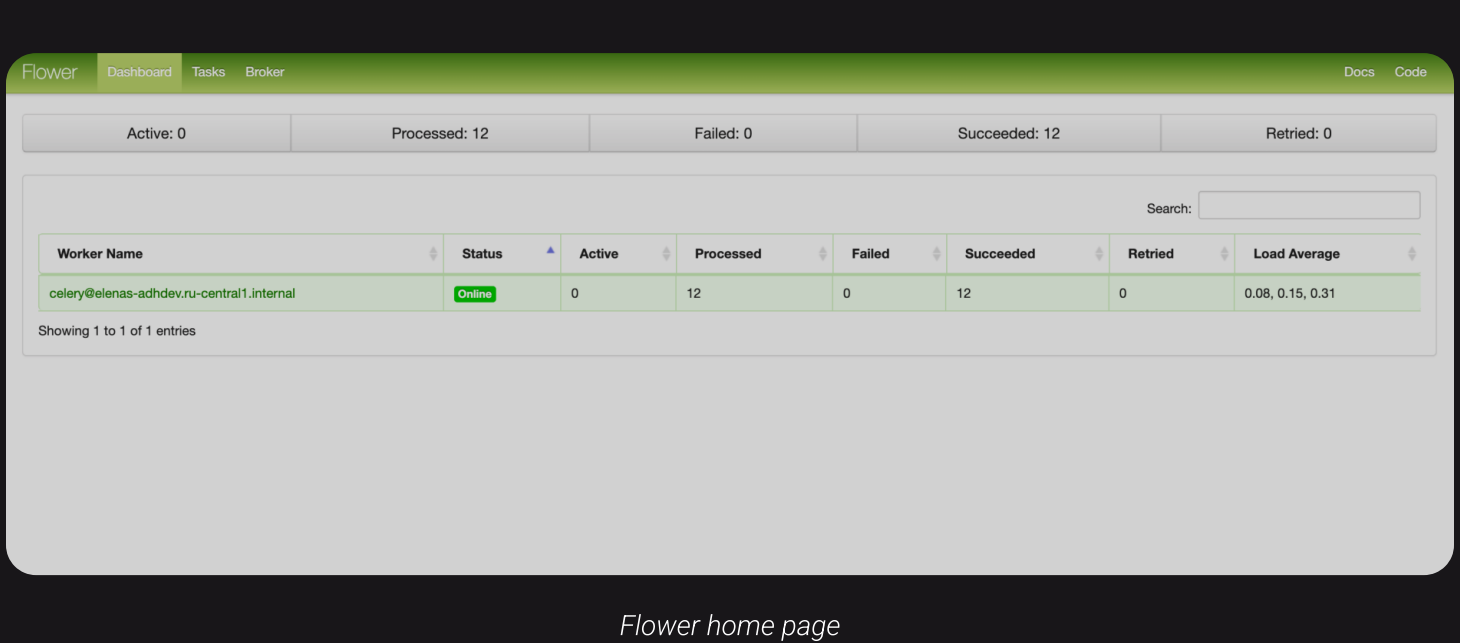
The ***Docs*** menu provides links to the following helpful resources:

- ***Documentation*** — the official Airflow documentation.
- ***Airflow Website*** — the official Airflow website.
- ***GitHub Repo*** — the Airflow project repository on GitHub.
- ***REST API Reference (Swagger UI)*** — the REST API methods description in the Swagger format.
- ***REST API Reference (Redoc)*** — the REST API methods description in the Redoc format.

Flower UI

Flower is a web UI tool for monitoring and administrating Celery clusters. The Flower web UI has four tabs:

- ***Workers*** — the page provides a list of existing Celery workers and information about them.
- ***Tasks*** — the page displays a list of tasks with their worker, UUID, state, `args` / `kwargs`, and start and runtime information.
- ***Broker*** — the page shows information about the Celery brokers.
- ***Documentation*** — the tab redirects to the official Flower documentation.


Flower
Dashboard
Tasks
Broker
Docs
Code
Search:

Active: 0 Processed: 12 Failed: 0 Succeeded: 12 Retried: 0

Worker Name	Status	Active	Processed	Failed	Succeeded	Retried	Load Average
celery@elenas-adhdev.ru-central1.internal	Online	0	12	0	12	0	0.08, 0.15, 0.31

Showing 1 to 1 of 1 entries

Flower home page

For more information about Flower, see the Flower documentation.

Create a simple DAG

Daria Barysheva

Contents

To Table of Contents

Step 1. Prepare a DAG file
Step 2. Run the DAG via Web UI
Step 3. Check the results

This article explains how to create and run your first DAG to process CSV files. This DAG will simulate an ETL process and include the following tasks:

- **Extract**. This task loads the records from a source CSV file and transfers them to the next task. We will take the test file people_ages_titles.csv containing a list of 1000 people with their names and ages.
- **Transform**. This task selects the records received from the previous task according to the following criterion: the age of the people should be less or equal to 18.
- **Load**. This task loads the transformed records received from the previous task to a new CSV file.

NOTE

- In practice, you can modify this example by changing input files to database connections.
- See more examples in the Airflow documentation.
- The example described in this section refers to the Airflow version 1.10.11.

Step 1. Prepare a DAG file

Prepare a DAG by following these steps:

1. Add the ETL_test.py file to your DAG folder that is usually located in the home AirFlow folder and called *dags*. The content of the file is described below:

```
import pandas as pd ①
from datetime import timedelta, datetime
from airflow import DAG
from airflow.operators.python_operator import PythonOperator

dag = DAG('ETL_test', ②
    schedule_interval=timedelta(days=1),
    start_date=datetime(2021, 12, 17, 0))

def extract_people(path: str, **context) -> None: ③
    extracted_people = pd.read_csv(path,sep=',',header=1)
    context['ti'].xcom_push(key='extracted_people', value=extracted_people)

def transform_people(**context) -> None: ④
    extracted_people = context['ti'].xcom_pull(key='extracted_people', task_ids=['extract_people'])[0]
    transformed_people = []
    for person in extracted_people.values:
        if int(person[1]) <= 18:
            transformed_people.append({
                'Name': person[0],
                'Age': person[1]
            })
    context['ti'].xcom_push(key='transformed_people', value=transformed_people)

def load_people(path: str, **context) -> None: ⑤
    transformed_people = context['ti'].xcom_pull(key='transformed_people', task_ids=['transform_people'])
    loaded_people = pd.DataFrame(transformed_people[0])
    loaded_people.to_csv(path, index=None)

task_extract_people = PythonOperator( ⑥
    task_id='extract_people',
    python_callable=extract_people,
    op_kwargs={'path': '/opt/airflow/input/people_ages_titles.csv'},
    dag=dag,
    provide_context=True
)

task_transform_people = PythonOperator( ⑥
    task_id='transform_people',
    python_callable=transform_people,
    dag=dag,
    provide_context=True
)

task_load_people = PythonOperator( ⑥
    task_id='load_people',
    python_callable=load_people,
    op_kwargs={'path': '/opt/airflow/output/loaded_people.csv'},
    dag=dag,
    provide_context=True
)

task_extract_people >> task_transform_people >> task_load_people ⑦
```

① **Importing**. We import pandas to work with CSV files.

② **The DAG definition**. We create the `ETL_test` object of type DAG and make it run every day since 2021-12-17 00:00:00.

③ **The first Python function**. The first task `extract_people` will use it. This function gets the CSV file from the defined `path` and loads it using the `pandas` module. It uses the `context` object to transfer the loaded data through the XCOM data transport service to the next task `transform_people`.

④ **The second Python function**. The second task `transform_people` will use it. This function gets the extracted data from the first task `extract_people` and finds in it the records with the specified value of the column `Age` (less or equal to 18). Then it copies the found data to an array and makes that array available for the next task `load_people`.

⑤ **The third Python function**. The third task `load_people` will use it. The function gets the transformed data from the second task `transform_people` and saves it in a new file located at the specified `path`. For this purpose, it uses the module `pandas`.

⑥ **Declaration of tasks**. This part defines operators, functions, and input parameters that the tasks will use. It also binds the tasks to the DAG defined earlier.

⑦ **Dependency definition**. This code defines dependencies between created tasks.

IMPORTANT

If you use AirFlow installed in a Docker container (in ADH, it comes exactly like this), be careful with paths. For example, use */opt/airflow/* instead of */srv/airflow/home/*.

2. Make sure that the file is located in the DAG directory by running this command:

```
$ ls -la /srv/airflow/home/dags/
```

The output looks similar to this:

```
total 8
drwxr-xr-x. 3 50000 50000   65 Dec 18 13:11 .
drwxr-xr-x. 6 50000 50000  126 Dec 18 11:07 ..
-rw-r--r--. 1 50000 50000  799 Dec 16 07:18 adcm_check.py
-rw-r--r--  1 root  root  1771 Dec 18 13:10 ETL_test.py
drwxr-xr-x. 2 50000 50000   74 Dec 18 13:10 __pycache__
```

3. Make the *input* and *output* folders in the Airflow home directory:

```
$ sudo mkdir /srv/airflow/home/input
$ sudo mkdir /srv/airflow/home/output
```

Make sure the folders are created by listing the home directory:

```
$ ls -la /srv/airflow/home/
```

The output looks similar to this:

```
total 48
drwxr-xr-x. 6 50000 50000   126 Dec 18 13:58 .
drwxr-xr-x. 4 root  root     35 Dec 16 07:17 ..
-rw-r--r--. 1 root  root  37224 Dec 16 07:18 airflow.cfg
-rw-r--r--  1 50000 50000     3 Dec 18 11:07 airflow-webserver.pid
drwxr-xr-x. 3 50000 50000    65 Dec 18 13:11 dags
drwxr-xr-x  2 root  root      6 Dec 18 13:57 input
drwxr-xr-x. 7 50000 50000   121 Dec 17 15:55 logs
drwxr-xr-x  2 root  root      6 Dec 18 13:58 output
-rw-r--r--. 1 50000 50000  2528 Dec 16 07:19 unittests.cfg
```

4. Download the source file people_ages_titles.csv to the local file system of the Airflow server. Make sure the file is uploaded successfully using the following command:

```
$ ls -la ~
```

The result looks similar to this:

```
total 64
drwx------. 7 dasha dasha   196 Dec 17 15:44 .
drwxr-xr-x. 3 root  root     19 Aug 31 11:55 ..
drwx------. 3 dasha dasha    17 Aug 31 15:29 .ansible
-rw-------. 1 dasha dasha 23583 Dec 17 18:17 .bash_history
-rw-r--r--. 1 dasha dasha    18 Apr  1  2020 .bash_logout
-rw-r--r--. 1 dasha dasha   193 Apr  1  2020 .bash_profile
-rw-r--r--. 1 dasha dasha   231 Apr  1  2020 .bashrc
drwx------. 3 dasha dasha    17 Dec  7 11:46 .cache
-rw-rw-r--. 1 dasha dasha  7217 Nov 25 07:05 dasha
-rw-rw-r--. 1 dasha dasha 17661 Dec 17 15:44 people_ages_titles.csv
drwxrw----. 3 dasha dasha    19 Dec  6 14:06 .pki
drwxrwxr-x. 2 dasha dasha    21 Nov 30 09:25 .sqlline
drwx------. 2 dasha dasha    29 Aug 31 11:55 .ssh
```

5. Copy the *people_ages_titles.csv* file to the previously created folder *input*:

```
$ sudo cp ~/people_ages_titles.csv /srv/airflow/home/input
```

Make sure the operation is successful by running the following command:

```
$ ls -la /srv/airflow/home/input
```

The output looks similar to this:

```
total 20
drwxr-xr-x  2 root  root     36 Dec 18 14:03 .
drwxr-xr-x. 6 50000 50000   126 Dec 18 13:58 ..
-rw-r--r--  1 root  root  17661 Dec 18 14:03 people_ages_titles.csv
```

6. Change the owner of the created *input* and *output* folders and their contents to `airflow`. Otherwise, errors may occur when accessing files during the DAG execution. Use the following commands:

```
$ sudo chown -R 50000:50000 /srv/airflow/home/input
$ sudo chown -R 50000:50000 /srv/airflow/home/output
```

To check the result, run this command:

```
$ ls -la /srv/airflow/home/
```

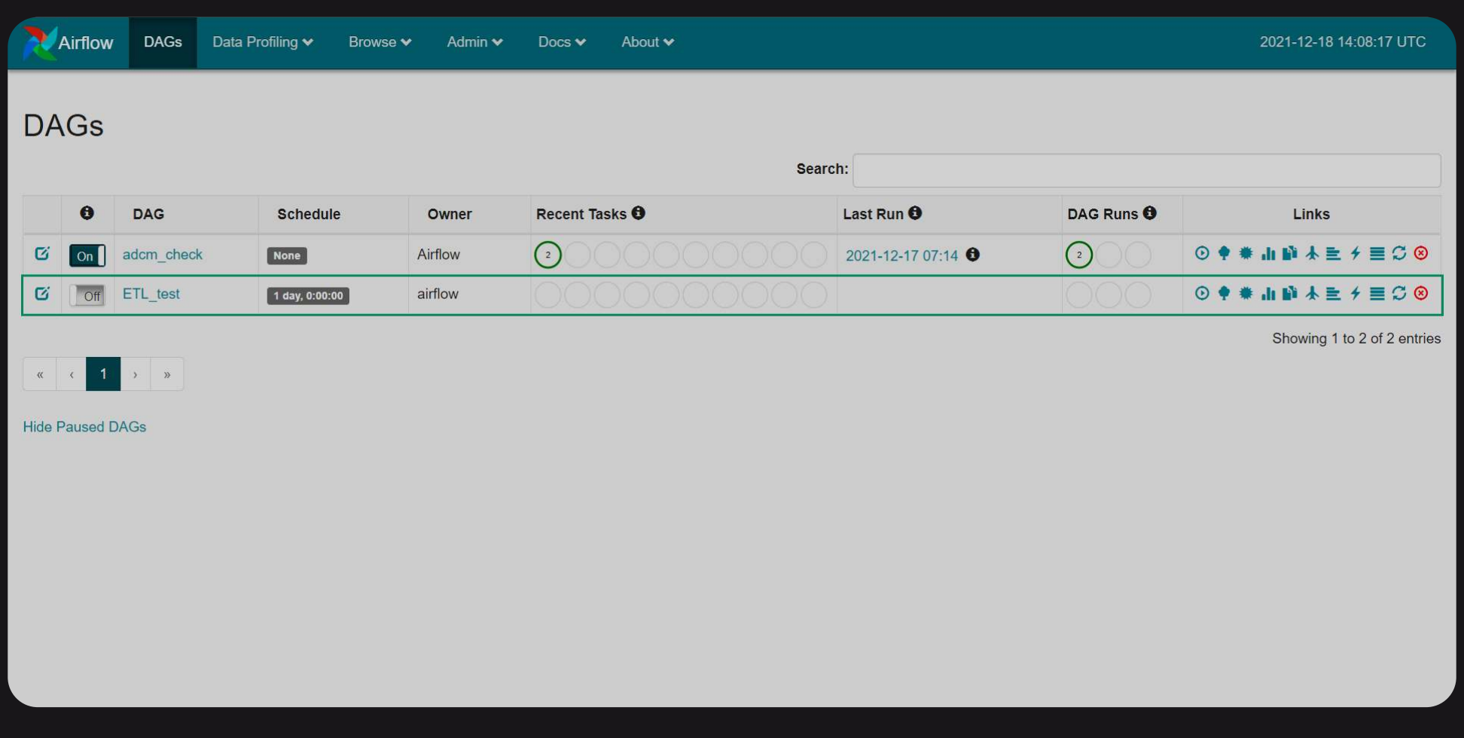
The output looks similar to this:

```
total 48
drwxr-xr-x. 6 50000 50000   126 Dec 18 13:58 .
drwxr-xr-x. 4 root  root     35 Dec 16 07:17 ..
-rw-r--r--. 1 root  root  37224 Dec 16 07:18 airflow.cfg
-rw-r--r--  1 50000 50000     3 Dec 18 11:07 airflow-webserver.pid
drwxr-xr-x. 3 50000 50000    65 Dec 18 13:11 dags
drwxr-xr-x  2 50000 50000    36 Dec 18 14:03 input
drwxr-xr-x. 8 50000 50000   137 Dec 18 14:12 logs
drwxr-xr-x  2 50000 50000     6 Dec 18 13:58 output
-rw-r--r--. 1 50000 50000  2528 Dec 16 07:19 unittests.cfg
```

Step 2. Run the DAG via Web UI

To run and monitor the DAG created earlier, follow these steps:

1. Open the home page of the Airflow Web UI. If everything is configured correctly, you will see the `ETL_test` DAG at the ***DAGs*** tab. Otherwise, the errors will be displayed.


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2021-12-18 14:08:17 UTC
Search:
Showing 1 to 2 of 2 entries
Hide Paused DAGs

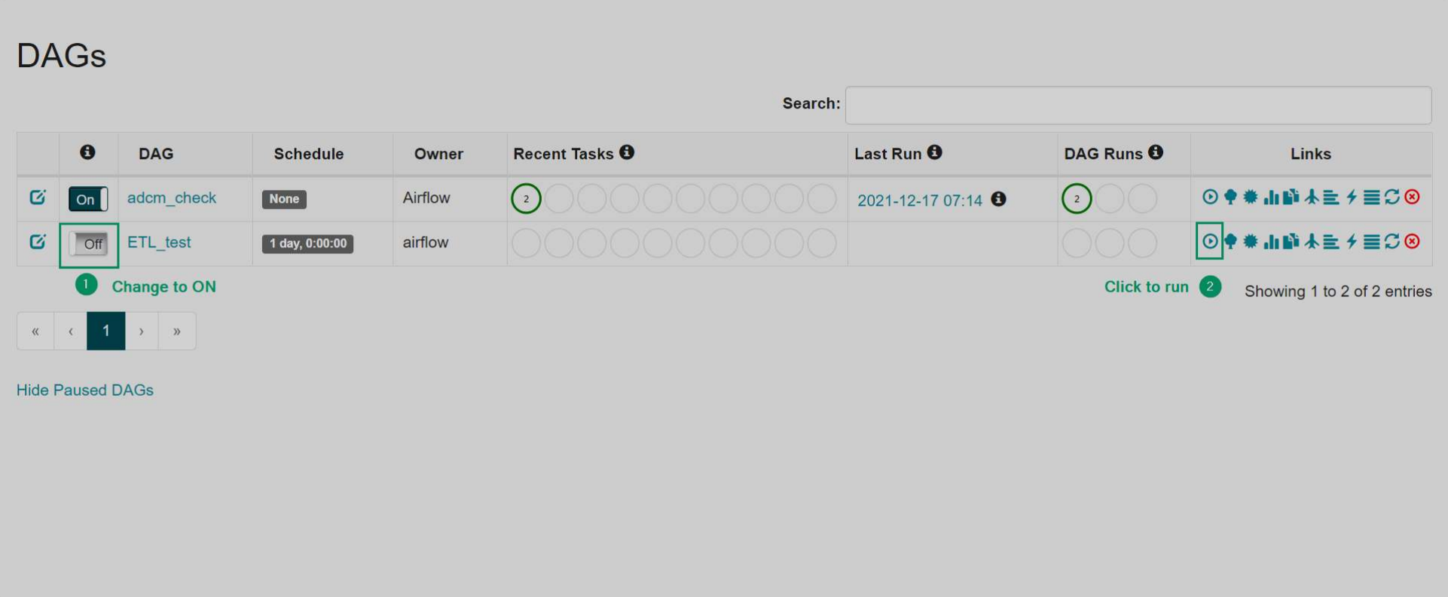
DAGs

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On	adcm_check	None	Airflow	2	2021-12-17 07:14	2	
	Off	ETL_test	1 day, 0:00:00	airflow				

The new DAG is displayed

2. Activate the DAG using one of the following ways:

   - Click the toggle and wait for the next running according to the configured schedule.
   - Open the DAG and click the icon ***Trigger Dag*** to run the DAG immediately.

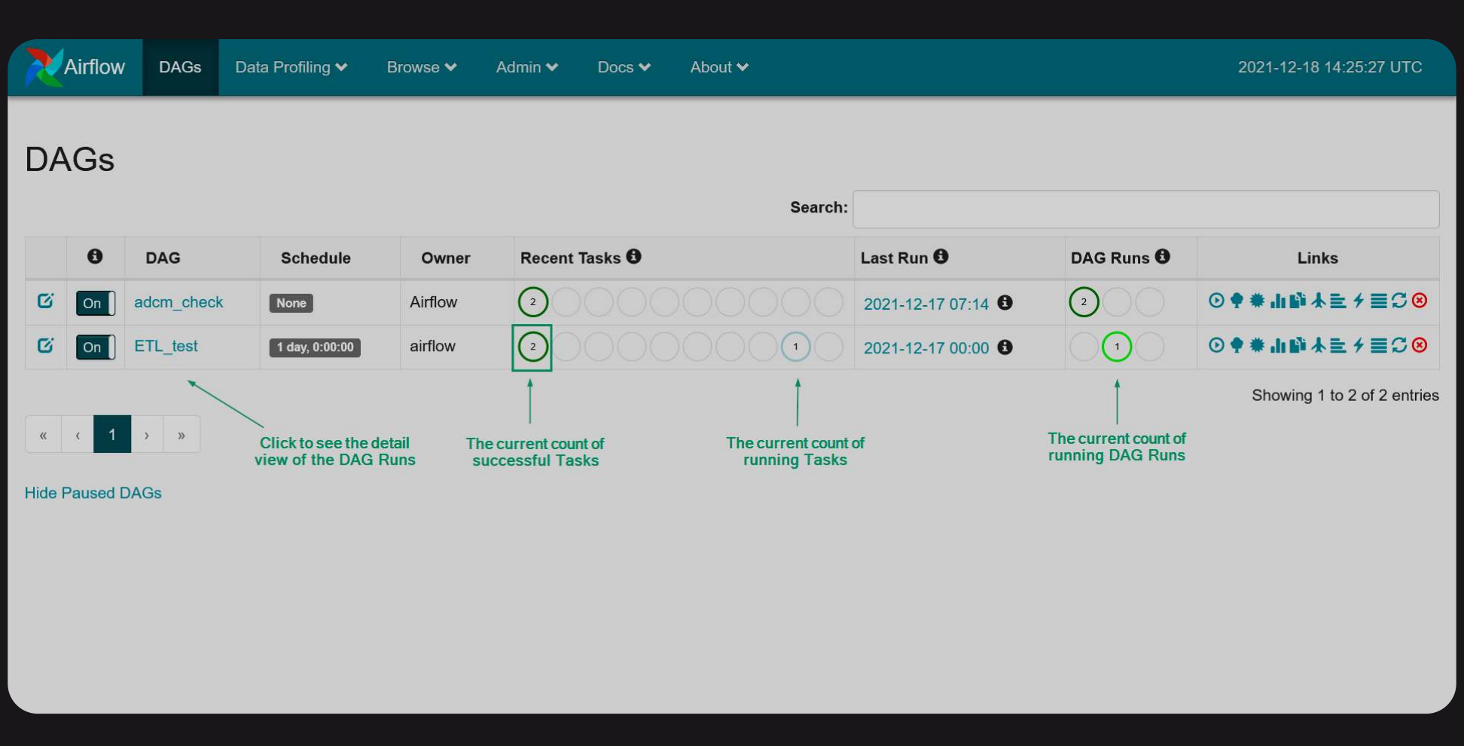

Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2021-12-18 14:26:49 UTC
Search:
1 Change to ON
Click to run 2
Showing 1 to 2 of 2 entries
Hide Paused DAGs

DAGs

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On	adcm_check	None	Airflow	2	2021-12-17 07:14	2	
	Off	ETL_test	1 day, 0:00:00	airflow				

Activating the DAG

3. After a DAG run starts, you can see the status information at the home page: the full number of successful, failed, running, and other types of DAG runs and task instances. To view the detailed information about the DAG runs, click the name of the DAG.

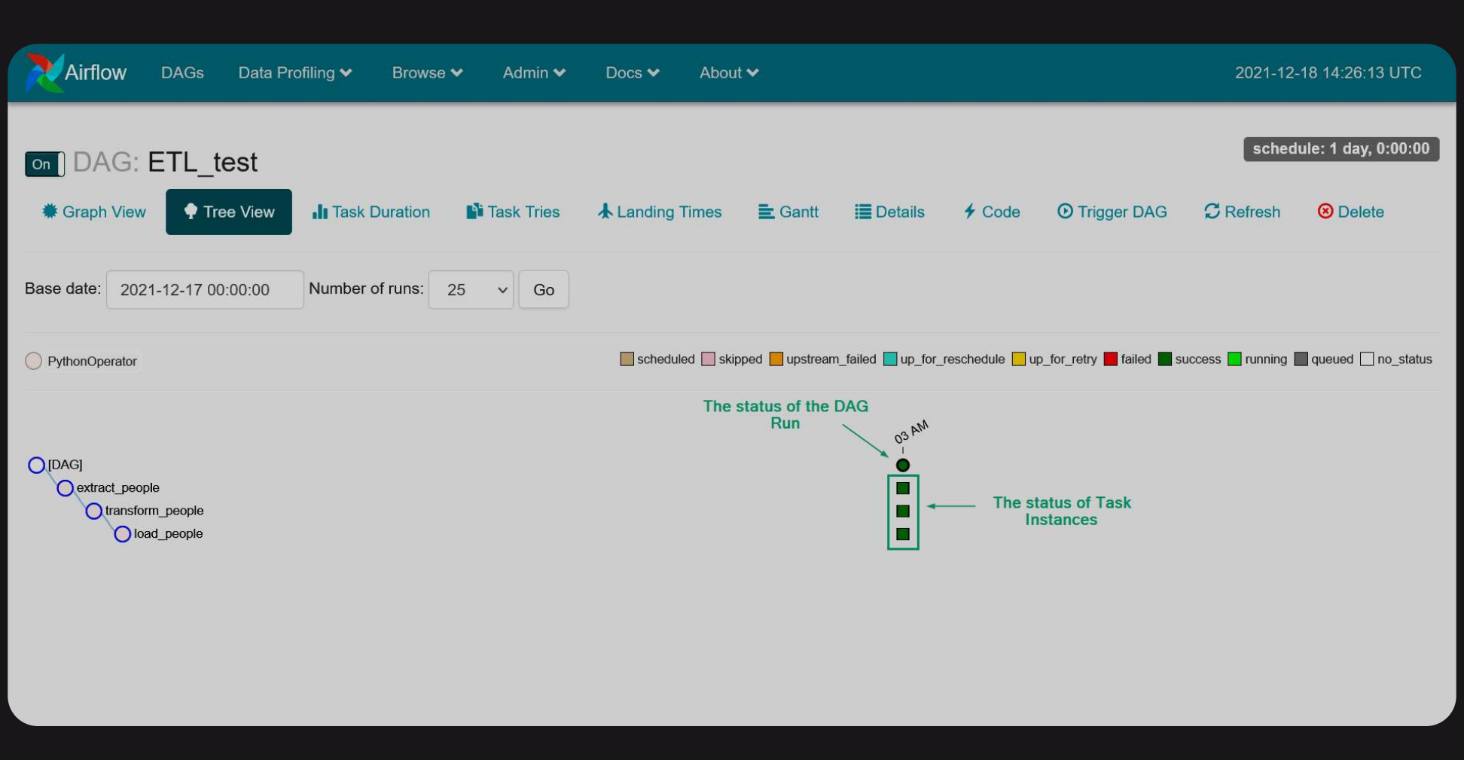

Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2021-12-18 14:25:27 UTC
Search:
Click to see the detail view of the DAG Runs
The current count of successful Tasks
The current count of running Tasks
The current count of running DAG Runs
Showing 1 to 2 of 2 entries
Hide Paused DAGs

DAGs

		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On	adcm_check	None	Airflow	2	2021-12-17 07:14	2	
	On	ETL_test	1 day, 0:00:00	airflow	2 1	2021-12-17 00:00	1	

The DAG is running

4. In the detailed information, you can see the statuses of all the DAG runs and task Instances. If necessary, you can change the view mode to ***Graph View*** in order to analyze task duration or perform other actions available in the main menu of the current page.


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2021-12-18 14:26:13 UTC
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Trigger DAG
Refresh
Delete
Go
PythonOperator
scheduled
skipped
upstream_failed
up_for_reschedule
up_for_retry
failed
success
running
queued
no_status
[DAG]
extract_people
transform_people
load_people
03 AM
The status of the DAG Run
The status of Task Instances

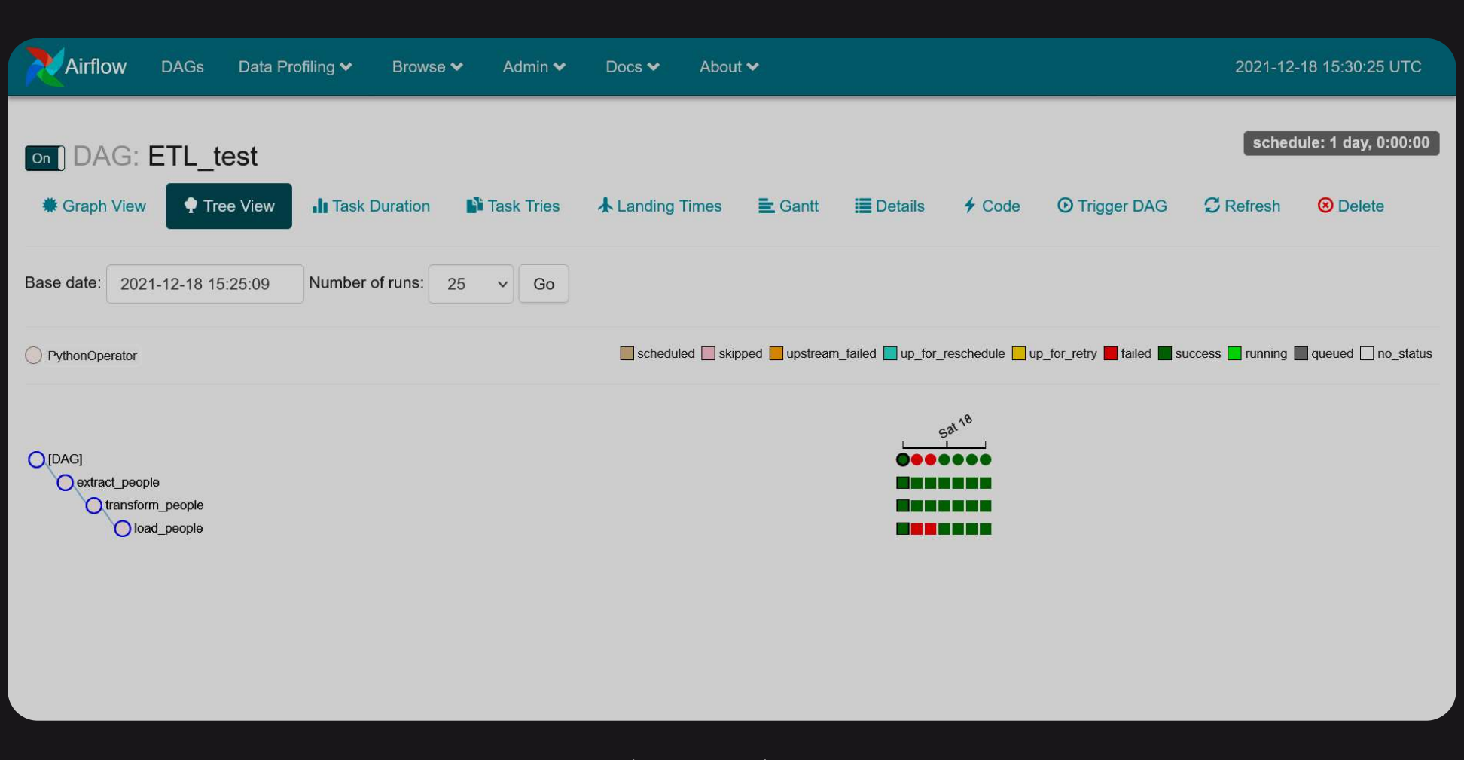
On DAG: ETL_test

schedule: 1 day, 0:00:00

Base date: 2021-12-17 00:00:00 Number of runs: 25

The detailed view of the DAG Runs

If some errors occur while running the DAG, the page looks as presented below.


Airflow
DAGs
Data Profiling
Browse
Admin
Docs
About
2021-12-18 15:30:25 UTC
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Trigger DAG
Refresh
Delete
Go
PythonOperator
scheduled
skipped
upstream_failed
up_for_reschedule
up_for_retry
failed
success
running
queued
no_status
[DAG]
extract_people
transform_people
load_people
Sat 18

On DAG: ETL_test

schedule: 1 day, 0:00:00

Base date: 2021-12-18 15:25:09 Number of runs: 25

The view with errors

Step 3. Check the results

In conclusion, let’s check whether the DAG has completed all the tasks correctly. Make sure the output file *loaded_people.csv* is located in the *output* folder:

1. Check the contents of the *output* folder:

```
$ ls -la /srv/airflow/home/output/
```

The list of files looks similar to this:

```
total 4
drwxr-xr-x  2 50000 50000  31 Dec 18 14:25 .
drwxr-xr-x. 6 50000 50000 126 Dec 18 13:58 ..
-rw-r--r--  1 50000 50000 832 Dec 18 14:25 loaded_people.csv
```

2. Get the output data:

```
$ cat /srv/airflow/home/output/loaded_people.csv
```

The output looks similar to this:

```
Name,Age
Arnold Bettie,18
Campbell Todd,18
Thompson Warren,18
Stokes Mittie,18
Davis Alan,18
Vaughn Belle,18
Munoz Ricardo,18
Norton Bertha,18
McGee Isabelle,18
Bailey Cameron,18
James Loretta,18
Myers Ricky,18
Torres Elsie,18
Phelps Rena,18
Hill Katharine,18
Hodges Abbie,18
Fitzgerald Timothy,18
Miller Francis,18
Banks Catherine,18
Casey Louise,18
```

Work with TaskFlow

Elena Kostyuchenko

Contents

To Table of Contents

- Use TaskFlow with traditional operators
  - Pass data between tasks
  - Cross-functional dependencies
- TaskFlow DAG structure

The TaskFlow API, added in Airflow 2, introduces a new approach for creating DAGs that enables direct function-to-function parameter passing and helps to make the DAGs code more concise and easier to read.

In previous Airflow versions, the most common way to write a DAG was using Python operators. When this approach is used, the tasks would be defined as follows:

```
def extract_people(path: str, **context) -> None:
    extracted_people = pd.read_csv(path,sep=',',header=1)
    context['ti'].xcom_push(key='extracted_people', value=extracted_people)

def load_people(path: str, **context) -> None:
    extracted_people = context['ti'].xcom_pull(key='extracted_people', task_ids=['extract_people'])[0]
    loaded_people = pd.DataFrame(extracted_people[0])
    loaded_people.to_csv(path, index=None)

task_extract_people = PythonOperator(
    task_id='extract_people',
    python_callable=extract_people,
    op_kwargs={'path': '/opt/airflow/input/people_ages_titles.csv'},
    dag=dag,
    provide_context=True
)

task_load_people = PythonOperator(
    task_id='load_people',
    python_callable=load_people,
    op_kwargs={'path': '/opt/airflow/output/loaded_people.csv'},
    dag=dag,
    provide_context=True
)

task_extract_people >> task_load_people
```

This is a simplified version of the traditional DAG. The full example is available in the Create a simple DAG article.

The example above illustrates limitations of the traditional approach that could lead to bloated code:

- It is necessary to use a context object to transfer the loaded data through the XCom data transport service to the next task.
- You have to separetely define operators, functions, and input parameters that the tasks will use.

The TaskFlow API introduces specific decorators that handle the data transferring between tasks using the same XCom (but without the need of explicitly calling it) and eliminate the need to instantiate Python operators.

When you write a decorated function, you don't need to specify `task_id` or `python_callable`. TaskFlow uses the name of the function as a task ID, but you can pass a `task_id` to the decorator as a parameter.

Airflow supports decorators for tasks, groups of tasks, and DAGs. In the TaskFlow paradigm, the same example would look like this:

```
@task(task_id="extract_people", retries=2) ①
def extract_people(path: str) -> pd.DataFrame: ②
    extracted_people = pd.read_csv(path, sep=',', header=1)
    return extracted_people ③

@task()
def transform_people(extracted_people: pd.DataFrame) -> list:
    transformed_people = []
    for person in extracted_people.values:
        if int(person[1]) <= 18:
            transformed_people.append({
                'Name': person[0],
                'Age': person[1]
            })
    return transformed_people

extracted_data = extract_people(SOURCE_PATH) ④
transformed_data = transform_people(extracted_data)
```

① Define a task by using the `@task` decorator. If required, pass the necessary task-level parameters to the decorator.

② Define the task's function.

③ There's no need to use `xcom_pull` and `xcom_push` since the function's return value is automatically passed to other tasks via XCom when the dependencies are defined.

④ Define your task dependencies by calling the task functions.

Use TaskFlow with traditional operators

While traditional operators offer more fine-grained control over data pipelines, TaskFlow makes cleaner transitions in workflows and more readable code.

In Airflow, you can use both traditional operators and the TaskFlow API.

Pass data between tasks

To transfer data from a decorated task to a traditional one, use the `.output` attribute of the task object when calling the decorated function.

For example:

```
def traditional_function():
    return 10

@task
def taskflow_function(x):
    return x + 5

sum_task = PythonOperator(
    task_id="sum_task",
    python_callable=traditional_function,
)

taskflow_function(sum_task.output)  # the task will return 15
```

To transfer data from a traditional task to a decorated one, you can call the decorated function in the traditional operator.

For example:

```
@task
def taskflow_function():
    return 10

def traditional_function(x):
    return x + 5

sum_task = PythonOperator(
    task_id="sum_task",
    python_callable=traditional_function,
    op_args=[taskflow_function()]
)

traditional_function >> taskflow_function()

# sum_task will return 15
```

Additionally, you can use `xcom_pull` and `xcom_push` between traditional and decorated functions as well.

Since TaskFlow decorators use the same traditional XCom functionality to save a function's return values, you can pull from a decorated task using the same `xcom_pull` syntax.

For example:

```
@task
def taskflow_function():
    return 10

def traditional_function(**kwargs):
    ti = kwargs["ti"]
    received_data = ti.xcom_pull(task_ids="taskflow_function")
    print(received_data)

traditional_task = PythonOperator(
    task_id="traditional_task",
    python_callable=traditional_function,
)

traditional_function >> taskflow_function()

# sum_task will return 15
```

IMPORTANT

TaskFlow saves the return value of a function only if the return value is used elsewhere in the DAG.

Cross-functional dependencies

When using both types of functions in the same DAG, you have to define dependencies according to each function type: function-to-function parameter passing for TaskFlow and using bit-shift operators for traditional functions.

For example:

```
@task
def taskflow_function_10():
    return 10

@task
def taskflow_function_20(x):
    return x + 20

def traditional_function(x):
    return x + 30

sum_task = PythonOperator(
    task_id="sum_task",
    python_callable=traditional_function,
)

taskflow_function_20(taskflow_function_10()) >> traditional_function

# sum_task will return 60
```

TaskFlow DAG structure

In Airflow 2, tasks are written as Python functions, decorated with `@task`. When creating a DAG, the task functions must be located inside the DAG function, which is decorated with `@dag`.

For example:

```
from datetime import timedelta, datetime
import pandas as pd
from airflow.decorators import dag, task ①


SOURCE_PATH = '/opt/airflow/input/people_ages_titles.csv'
TARGET_PATH = '/opt/airflow/output/loaded_people.csv'

@dag(                                          ②
    schedule_interval=timedelta(days=1),
    start_date=datetime(2021, 12, 17),
    catchup=False,
    tags=['example'],
)
def etl_example():

    @task()                                    ③
    def extract_people(path: str) -> pd.DataFrame:
        extracted_people = pd.read_csv(path, sep=',', header=1)
        return extracted_people

    @task()
    def transform_people(extracted_people: pd.DataFrame) -> list:
        transformed_people = []
        for person in extracted_people.values:
            if int(person[1]) <= 18:
                transformed_people.append({
                    'Name': person[0],
                    'Age': person[1]
                })
        return transformed_people

    @task()
    def load_people(transformed_people: list, path: str) -> None:
        loaded_people = pd.DataFrame(transformed_people)
        loaded_people.to_csv(path, index=None)

    extracted_data = extract_people(SOURCE_PATH)    ④
    transformed_data = transform_people(extracted_data)
    load_people(transformed_data, TARGET_PATH)

etl_dag = etl_example()    ⑤
```

① Import the task and DAG decorators.

② Instantiate a DAG using the `@dag` decorator and its Python function `example_etl`. By default, the name of the function acts as the DAG identifier.

③ Inside the DAG function, define your task functions.

④ Define the dependencies to ensure data transfer between the tasks.

⑤ Invoke the DAG function. All the decorated functions in your DAG must be called in that DAG's file so that Airflow can register them.

Use sensors in Airflow

Elena Kostyuchenko

Contents

To Table of Contents

Sensor parameters

Work with sensors

Write a custom sensor

In Airflow, a sensor is an operator that is used when, in a DAG, it is required to wait until a condition is met to continue with the tasks.

The conditions depend on the type of sensor. Airflow provides standard operators that, for example, allow a task to wait for a certain time to pass (TimeDeltaSensor), wait until a file appears in the system (FileSensor), or wait until another task in another DAG has finished (ExternalTaskSensor). When a sensor's condition is met, it can trigger downstream tasks.

Sensor parameters

Sensors have the following base parameters:

- `poke_interval` — the time that the job should wait in between each try. The default is 60 seconds.
- `timeout` — the maximum number of seconds a sensor is allowed to run for. The default timeout is 7 days, or 60 * 60 * 24 * 7 seconds, after which the sensor's task will be terminated. The value must be less than the schedule interval of your DAG to avoid hanging tasks.
- `mode` — the indicator of how long a sensor task will occupy a worker slot. Can have one of two values:
  - `poke` — the sensor takes up a worker slot until it gets completed;
  - `reschedule` — the sensor takes up a worker slot only during checks and, in between each `poke_interval`, the scheduler releases the slot.

Work with sensors

Sensor operators are derived from the `airflow.sensors.base` class and inherit its attributes. Airflow provides several sensors, which are described in the table below, and some of the sensors can be available through different providers.

Standard Airflow sensors

Name	Description
airflow.sensors.bash	Executes a bash command or a script that can be used as a custom sensor. The command should return `0` when it succeeds, any other value otherwise
airflow.sensors.date_time	Waits until the specified datetime. An advantage of this sensor is idempotence for the `target_time`. It can be applied to cases where TimeSensor and TimeDeltaSensor are not suited
airflow.sensors.external_task	Waits for a task in another DAG to complete before getting itself completed
airflow.sensors.filesystem	Waits for a file or folder to land in a filesystem. If the path given is a directory, then this sensor will return `True` if the directory is not empty
airflow.sensors.python	Executes an arbitrary callable and waits for it to finish. The `@task.sensor` decorator is recommended over the classic PythonSensor to execute Python callables to check for `True` condition
airflow.sensors.time_delta	Waits for a specified time period
airflow.sensors.time_sensor	Waits until the specified time of the day
airflow.sensors.weekday	Waits until the first specified day of the week. For example, if the execution day of the task is `2018-12-22` (Saturday) and you pass `FRIDAY`, the task will wait until next Friday

A more detailed description of each sensor is available in the Airflow documentation.

Here's an example of a possible sensor usage in a DAG:

```
from datetime import datetime, timedelta

from airflow import DAG
from airflow.sensors.time_delta import TimeDeltaSensor, TimeDeltaSensorAsync ①

dag = DAG(
    dag_id="sensor_example",
    schedule=None,
    start_date=datetime(2023, 6, 4),
)

time_sensor_example = TimeDeltaSensor( ②
   task_id='time_sensor_example',
   dag=dag,
   delta=timedelta(minutes=1), ③
   poke_interval=10, ④
   timeout=30, ⑤
   soft_fail=True, ⑥
)
```

① Import the sensor operators. `TimeDeltaSensorAsync` is an async version of the `TimeDeltaSensor` operator.

② Instantiate the sensor. If the sensor mode is not defined, it defaults to `poke`.

③ Specifies for how long to wait before the task can be marked as succeeded.

④ Defines how often a sensor checks the conditions (in seconds).

⑤ Defines for how long the sensor will be working.

⑥ Marks the sensor's task as `SKIPPED` if it fails.

To avoid deadlocks and other performance issues, it is recommended to check the following conditions:

- Set up a timeout for your sensor that will work for each particular case. The default value for this parameter might be too long in some cases.
- Use the reschedule mode for long-running sensors so that they do not constantly occupy a working slot, and the poke mode for a short polling interval (i.e., less than 5 minutes) to avoid scheduler overload.
- Make sure that the `poke_interval` parameter is applicable to your case. The default value might be unnecessarily frequent.

Write a custom sensor

To create a custom sensor, you have to inherit the `BaseSensorOperator` class in your Python function.

In this example, we will create a new sensor that checks the existence of a file.

1. In the *dags* directory of your Airflow home, add a new Python file that will contain your custom sensor:

```
$ sudo vi /opt/airflow/dags/custom_sensor.py
```

2. Add the code for your sensor:

```
import os
from airflow.sensors.base import BaseSensorOperator ①

class MyCustomFileSensor(BaseSensorOperator): ②
    def __init__(self, filepath, *args, **kwargs): ③
        self.filepath = filepath
        super().__init__(*args, **kwargs)

    def poke(self, context): ④
        print(f'checking if a file {self.filepath} exists...')
        return os.path.exists(self.filepath) ⑤
```

① Import the base sensor class.

② Inherit the base class in your custom sensor.

③ Add additional arguments and keep the arguments of the parent class.

④ Override the `poke` method.

⑤ Add the logic for checking the existence of the file.

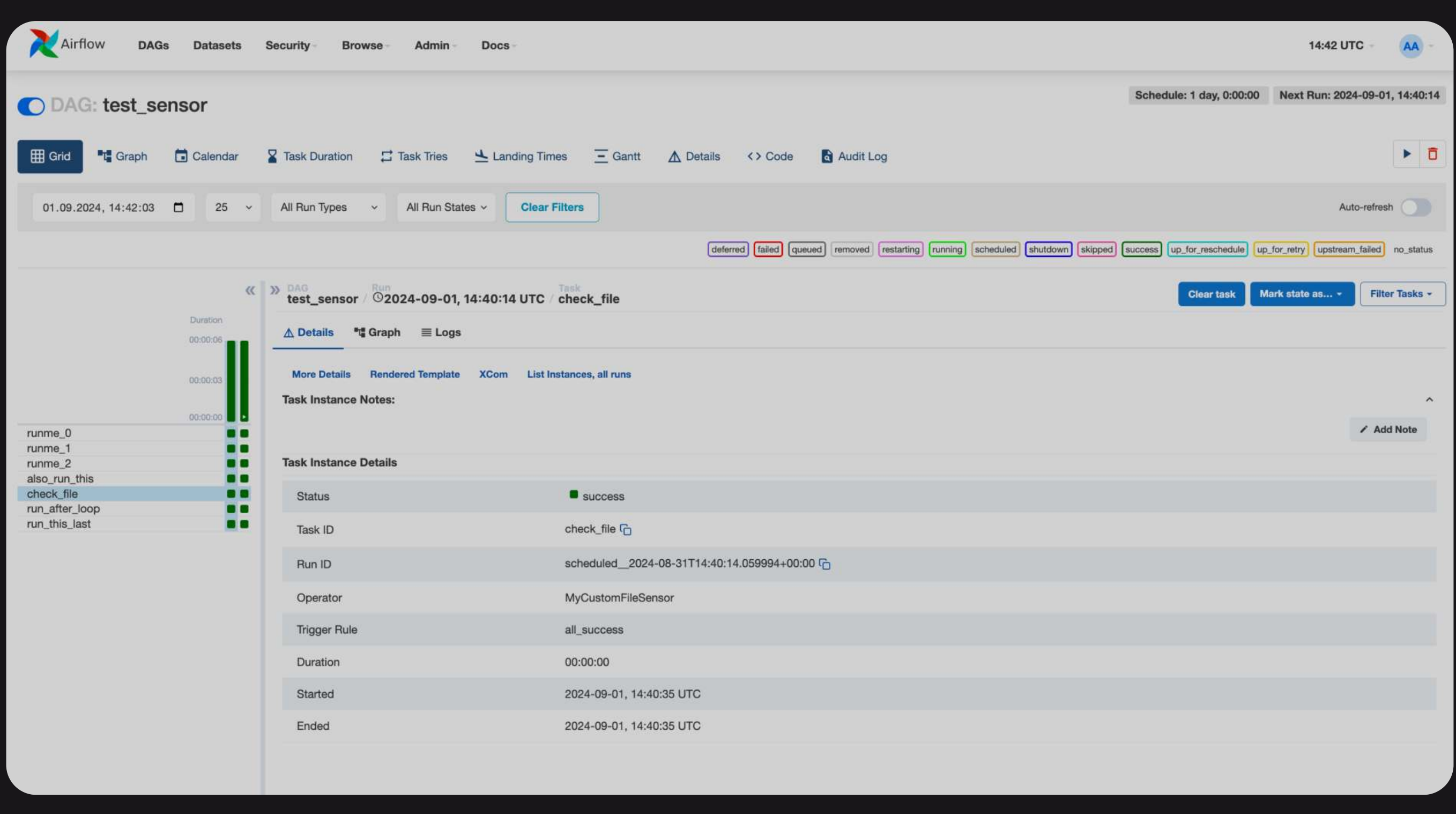
3. Call your custom sensor inside a DAG:

```
from custom_sensor import MyCustomFileSensor ①

check_file = MyCustomFileSensor(     ②
    task_id='check_file',
    filepath='/opt/airflow/test_file.txt',
    poke_interval=10,
    timeout=60,
    dag=dag
)
```

① Import the custom sensor class.

② Call the custom sensor function with the required parameters.

4. Go to the Airflow web UI to trigger the DAG and make sure that the custom sensor function is working.



DAG: test_sensor

Schedule: 1 day, 0:00:00 Next Run: 2024-09-01, 14:40:14

01.09.2024, 14:42:03 25 All Run Types All Run States

DAG test_sensor / Run 2024-09-01, 14:40:14 UTC / Task check_file

Task Instance Notes:

Task Instance Details

Status	success
Task ID	check_file
Run ID	scheduled__2024-08-31T14:40:14.059994+00:00
Operator	MyCustomFileSensor
Trigger Rule	all_success
Duration	00:00:00
Started	2024-09-01, 14:40:35 UTC
Ended	2024-09-01, 14:40:35 UTC

Airflow UI: DAG page

The DAG page shows that the `check_file` task with the `MyCustomFileSensor` class has finished successfully.

Customize DAG scheduling

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Cron-based schedule
- Timedelta schedule
- Events timetable
- Custom timetables

Overview

Airflow provides the ability to run data pipelines on a schedule, allowing you to define the data interval and logical date of each DAG run.

A DAG schedule can be defined with one of the following tools:

- built-in timedelta object of the Python `datetime` module;
- Linux Cron;
- custom timetable classes.

All the schedule parameters, like data interval, date and time, and other, are determined in a DAG's internal timetable. DAGs scheduled with a cron expression or a timedelta object are internally converted into a timetable as well.

Airflow has default timetables that handle cron expressions or timedeltas, but for DAGs with a more specific schedule, it might be required to create a custom timetable class and pass it to the DAG's schedule argument.

The examples of cases when a custom timetable is needed:

- Data pipelines start at different times each day.
- The DAG schedule requires the use of a different calendar.
- DAG runs with overlapping data intervals. For example, a DAG run must cover the period of the previous runs.
- DAG runs with uneven intervals.

Cron-based schedule

Cron expressions in Airflow follow the standard cron syntax, which consists of five fields separated by whitespaces, where each field can accept values listed in the table below.

A cron expression syntax is `<1> <2> <3> <4> <5>`, where:

- `<1>` — the minute in the range: 0-59.
- `<2>` — the hour in the range: 0-23.
- `<3>` — the day of the month in the range: 1-31.
- `<4>` — the month in the year. The range is from 1 to 12, where 1 is January.
- `<5>` — the numeric indicator of a day in the week. Each day is represented by a number in a range from 0 to 6, where 0 is Sunday.

For example, a DAG's schedule represented by a cron expression `0 18 * * 5` runs every Friday at 6 PM.

Possible value	Example
A single value	5
A range of values defined using a dash	1-5
A list of values separated by commas	1,3,5
Every possible value is represented by an asterisk	*

Airflow has two timetables `CronTriggerTimetable` and `CronDataIntervalTimetable` for handling cron schedules. Both timetables trigger DAG runs at the same time but the timestamp for the `run_id` for DAGs will be different.

For example, if a DAG with the schedule defined as `0 0 * * *` (12 AM every day) is enabled at 3 PM on January 31, `CronTriggerTimetable` will trigger a new DAG run at 12 AM on February 1, but `CronDataIntervalTimetable` will trigger the DAG immediately, because the DAG run for the daily time interval beginning at 12 AM on January 31 did not occur yet.

An example DAG for `CronTriggerTimetable` that triggers a DAG run at 1 AM on Wednesday:

```
from datetime import timedelta
from airflow import DAG
from airflow.timetables.trigger import CronTriggerTimetable

def cron_timetable(*args, **kwargs):
    return CronTriggerTimetable(kwargs['cron_expression'])

dag = DAG(
    dag_id='my_dag',
    timetable=cron_timetable(cron_expression='0 1 * * 3', timezone="UTC"),
    start_date=days_ago(2),
    tags=['example'],
)

@dag(schedule=CronTriggerTimetable("0 1 * * 3", timezone="UTC"), ...)
def example_dag():
    pass
```

When using `CronDataIntervalTimetable`, you can specify a static data interval using the `interval` argument. It must use `datetime.timedelta` or `dateutil.relativedelta.relativedelta` objects.

If specified, the DAG's data interval spans the specified duration and ends with the trigger time.

For example:

```
from datetime import timedelta
from airflow.timetables.trigger import CronTriggerTimetable

@dag(
    schedule=CronTriggerTimetable(
        "0 18 * * 5",
        timezone="UTC",
        interval=timedelta(days=4, hours=9),
    ),
    ...,
)
def example_dag():
    pass
```

The DAG from the above example will run every Friday at 6 PM.

Timedelta schedule

The timedelta schedule uses a timetable that runs DAGs with a specified time interval. This timetable relies on the data interval value provided by the user and does not necessarily align execution dates with the start of the day or an hour.

You can use a timedelta schedule by providing a `datetime.timedelta` or `dateutil.relativedelta.relativedelta` values to the schedule parameter.

For example:

```
@dag(schedule=datetime.timedelta(minutes=30))
def example_dag():
    pass
```

Events timetable

Events timetable enables running DAGs at specified datetimes. The list of dates should have a moderate size since it must be loaded every time the DAG is parsed.

For example:

```
from airflow import DAG
import pendulum
from airflow.timetables.events import EventsTimetable

@dag(
    schedule=EventsTimetable(
        event_dates=[
            pendulum.datetime(2024, 4, 5, 8, 27, tz="America/Vancouver"),
            pendulum.datetime(2024, 4, 17, 8, 27, tz="America/Vancouver"),
            pendulum.datetime(2024, 4, 22, 20, 50, tz="America/Vancouver"),
        ],
        description="Example dates",
        restrict_to_events=False,
    ),
    ...,
)
def example_dag():
    pass
```

Custom timetables

Custom timetables allow users to define their own schedules in Python code when cron-based and timedelta timetables are not sufficient.

To create a custom timetable, you need to:

- create a class that inherits from a timetable;
- implement the `next_dagrun_info()` and `infer_manual_data_interval()` methods, both of which return a `DataInterval` object;
- register the custom timetable as part of an Airflow plugin.

The `next_dagrun_info()` method returns the data interval for the DAG's regular schedule and specifies the logic to calculate it. The method also contains the logic for calculating the `start_date`, `end_date`, and `catchup` parameters.

The `infer_manual_data_interval()` method determines how to define the interval in the case of manual execution.

The example below illustrates how to write a DAG with a custom timetable.

To create a DAG with a custom timetable:

1. In the *plugins* directory of your Airflow home, add a new Python file that will contain your custom timetable:

```
$ sudo vi /opt/airflow/plugins/custom_timetable.py
```

2. In the *custom_timetable.py* file, define the structure of the timetable:

```
from datetime import timedelta
from typing import Optional
from pendulum import Date, DateTime, Time, timezone
from airflow.plugins_manager import AirflowPlugin
from airflow.timetables.base import DagRunInfo, DataInterval, TimeRestriction, Timetable ①

UTC = timezone("UTC") ②

class CustomTimetable(Timetable): ③
    pass

class CustomTimetablePlugin(AirflowPlugin): ④
    name = "custom_timetable_plugin"
    timetables = [CustomTimetable]
```

① Import the dependencies.
② Declare global variables if necessary.
③ Inherit the base class in your custom timetable.
④ Register your timetable as an Airflow plugin.

3. In the `CustomTimetable` class, write the implementation of the `next_dagrun_info()` method:

```
class CustomTimetable(Timetable):

    def next_dagrun_info(
        self,
        *,
        last_automated_data_interval: Optional[DataInterval],
        restriction: TimeRestriction,
    ) -> Optional[DagRunInfo]:
        if last_automated_data_interval is not None:  ①
            last_start = last_automated_data_interval.start
            delta = timedelta(days=1)
            if last_start.hour == 6:
                next_start = last_start.set(hour=16, minute=30).replace(tzinfo=UTC)
                next_end = (last_start+delta).replace(tzinfo=UTC)
            else:
                next_start = (last_start+delta).set(hour=6, minute=0).replace(tzinfo=UTC)
                next_end = (last_start+delta).replace(tzinfo=UTC)
        else: ②
            next_start = restriction.earliest
            if next_start is None:
                return None
            if not restriction.catchup: ③
                next_start = max(next_start, DateTime.combine(Date.today(), Time.min).replace(tzinfo=UTC))
            next_start = next_start.set(hour=6, minute=0).replace(tzinfo=UTC)
            next_end = next_start.set(hour=16, minute=30).replace(tzinfo=UTC)
        if restriction.latest is not None and next_start > restriction.latest: ④
            return None
        return DagRunInfo.interval(start=next_start, end=next_end)
```

① Write the logic for the case when there was a previous run.
② Write the logic for the first run.
③ Implement the catchup behavior.
④ Make sure the next run won't start out of schedule.

4. In the `CustomTimetable` class, write the logic for scheduling runs in case of a manual start in the `infer_manual_data_interval()` method:

```
class CustomTimetable(Timetable):

    def infer_manual_data_interval(self, run_after: DateTime) -> DataInterval:
        delta = timedelta(days=1)
        if run_after >= run_after.set(hour=6, minute=0) and run_after <= run_after.set(hour=16, minute=30):
            start = (run_after-delta).set(hour=16, minute=30, second=0).replace(tzinfo=UTC)
            end = run_after.set(hour=6, minute=0, second=0).replace(tzinfo=UTC)
        elif run_after >= run_after.set(hour=16, minute=30) and run_after.hour <= 23:
            start = run_after.set(hour=6, minute=0, second=0).replace(tzinfo=UTC)
            end = run_after.set(hour=16, minute=30, second=0).replace(tzinfo=UTC)
        else:
            start = (run_after-delta).set(hour=6, minute=0).replace(tzinfo=UTC)
            end = (run_after-delta).set(hour=16, minute=30).replace(tzinfo=UTC)
        return DataInterval(start=start, end=end)
```

5. Use the custom timetable in a DAG:

```
from custom_timetable import CustomTimetable ①

@dag(
    dag_id="example_dag",
    start_date=datetime(2024, 3, 10),
    schedule=CustomTimetable(), ②
    default_args={
        "retries": 1,
        "retry_delay": duration(minutes=3),
    },
    catchup=True
)
```

① Import the custom timetable.
② Call the custom timetable class to use it as the DAG's schedule.

For more information on how to run a DAG, see the Create a simple DAG article.

IMPORTANT

Adding or updating an Airflow plugin, including custom timetables, requires a restart of Airflow Scheduler and the web server.

Add custom operators and hooks in Airflow

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Create a custom operator

Create a custom hook

Overview

Airflow provides standard operators that, for example, allow users to execute bash commands, call arbitrary Python functions, create a connection to an external app, or check for a specific condition.

The list of operators that Airflow provides out-of-the-box is extensive, but in some cases it is necessary to modify an operator or create your own.

This article describes how to write custom operators and hooks. To learn about custom sensors, see the Use sensors in Airflow article.

When creating any custom Airflow operator, you need to:

1. Write the code for a new operator in the `__init__` method.
2. Inherit an Airflow class and either rewrite or extend its functionality by overriding methods in a derived class:
   - In the constructor, define the parameters of your operator.
   - Provide the code to execute when the runner calls the operator.
3. Place the file in a directory that is present in the Airflow's `PYTHONPATH`. By default, it's the following directories: *dags*, *plugins*, and *config*.

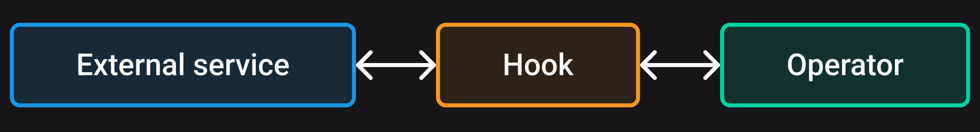
The examples below illustrate implementations of these steps for writing a custom operator and a hook.

Division of the logic between operators and hooks

An operator should contain the business logic of your task and in-memory states. If you need to establish communication with an external service (API, database, etc.), it's better to write it as a hook.

A hook should not contain the logic of the operator but only expose methods that will then be used by an operator for interfacing with an external service.

By keeping the logic separated into two components, you will be able to build multiple operators that use the same hook without duplicating the code.



Operators and hooks interaction path

Create a custom operator

Here's an example of creating a custom operator based on the `airflow.models.baseoperator.BaseOperator` class:

1. In the *dags* directory of your Airflow home, add a new Python file that will contain your custom operator:

```
$ sudo vi /opt/airflow/dags/custom_operator.py
```

2. Add the code for your operator:

```
from airflow.models.baseoperator import BaseOperator  ①

class CustomOperator(BaseOperator):  ②
    def __init__(self, name: str, **kwargs) -> None:  ③
        super().__init__(**kwargs)
        self.name = name

    def execute(self, context):  ④
        message = f"Hello, {self.name}!"
        print(message)
        return message
```

① Import the base operator class.

② Inherit the base class in your custom operator.

③ Add new arguments and keep the arguments of the parent class.

④ Override the `execute` method with the desired functionality.

3. Call your custom operator inside a DAG:

```
from custom_operator import CustomOperator ①

print_message = CustomOperator(    ②
    task_id='print_message',
    name='CustomOperator',
    poke_interval=10,
    timeout=60,
    dag=dag
)
```

① Import the custom operator class.

② Call the custom operator function.

Create a custom hook

Creating a custom hook is different from the process of writing a custom operator in three ways:

- The custom hook must inherit from a hook class, for example, the `BaseHook` class or any other existing hook.
- You can include a `.get_conn()` method wrapping around a call to the `.get_connection()` method to retrieve information from an Airflow connection.
- There must exist an Airflow connection to be used by the hook.

Below is an example of how to create a custom hook function that initiates a new connection to an external tool:

1. In the *dags* directory of your Airflow home, add a new Python file that will contain your custom hook:

```
$ sudo vi /opt/airflow/dags/custom_hook.py
```

2. Add the code for your hook:

```
from airflow.hooks.base import BaseHook ①

class CustomHook(BaseHook): ②

    conn_name_attr = "custom_conn_id" ③
    default_conn_name = "custom_conn_default" ④
    conn_type = "http" ⑤
    hook_name = "CustomHook" ⑥

    def __init__(
        self, custom_conn_id: str = default_conn_name, *args, **kwargs
    ) -> None: ⑦
        super().__init__(*args, **kwargs) ⑧
        self.custom_conn_id = custom_conn_id ⑨
        self.get_conn() ⑩

    def get_conn(self):
        conn_id = getattr(self, self.conn_name_attr)
        conn = self.get_connection(conn_id)
        return conn
```

① Import the base hook class.

② Define the class to inherit from the base class.

③ Add the parameter that receives the connection ID.

④ Specify a default connection ID.

⑤ Specify the connection type.

⑥ Specify the name of the hook.

⑦ Define the `.init()` method.

⑧ Initialize the parent hook.

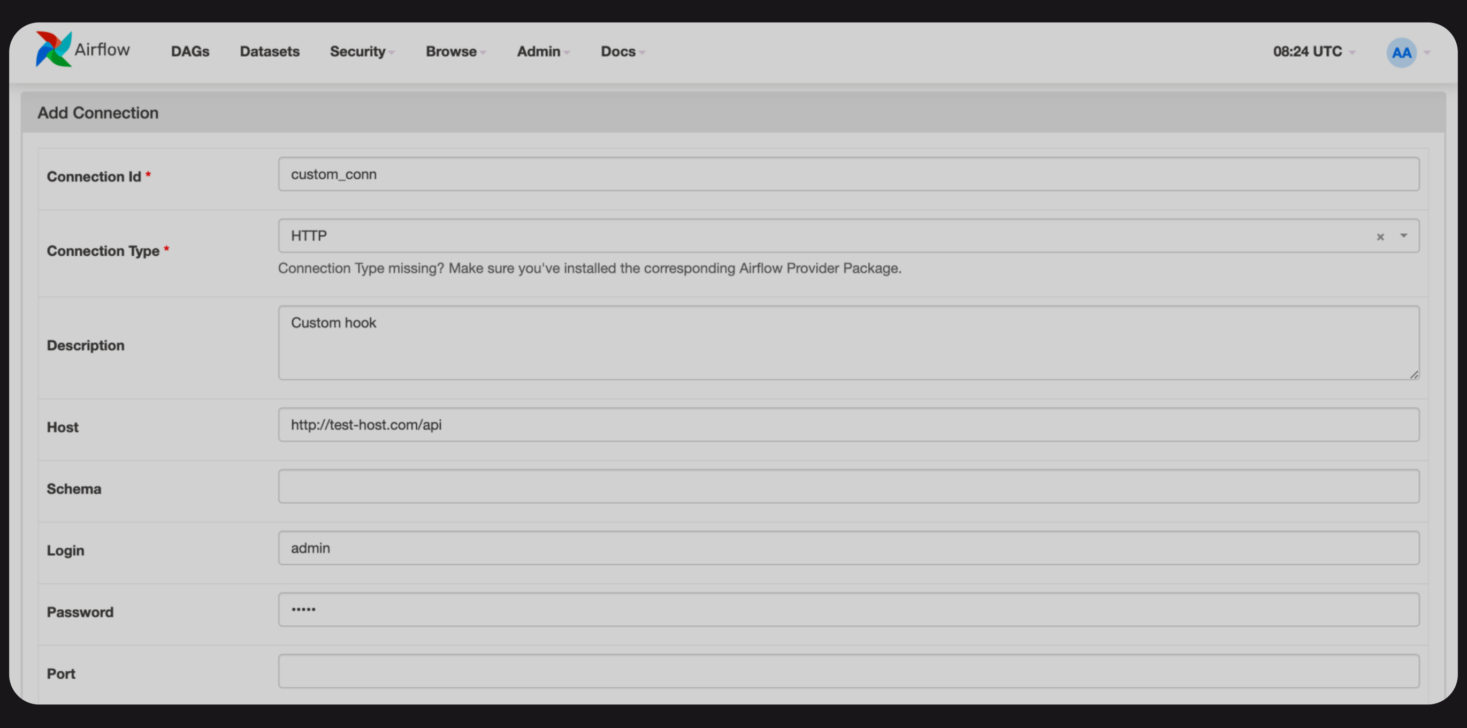
⑨ Assign class variables.

⑩ Optionally, call the `.get_conn()` method upon initialization.

3. Call your custom hook inside a DAG:

```
from custom_hook import CustomHook
```

4. To use the custom hook, you need to create an Airflow connection with the parameters specified in the hook code.



Add Connection

Connection Id *: custom_conn

Connection Type *: HTTP

Connection Type missing? Make sure you've installed the corresponding Airflow Provider Package.

Description: Custom hook

Host: http://test-host.com/api

Schema:

Login: admin

Password: •••••

Port:

Creating a new connection in Airflow UI

For more information about the Airflow web interface, see the Airflow UI overview article.

Generate DAGs dynamically in Airflow

Elena Kostyuchenko

Contents

To Table of Contents

Overview
Dynamic DAGs generation
Dynamic task mapping

Overview

Traditional DAGs are convenient to use for a single-source pipeline. For example, to extract data from a file, perform the necessary actions, and load the results.

For cases where you need to run the same data pipeline with different parameters on several data sources, it's more useful to generate DAGs dynamically for each case.

There are two ways of creating DAGs for variable scenarios:

- **Dynamic DAGs generation**
  In dynamically generated DAGs, the structure of the DAG changes based on static predefined values (configuration files, environment variables, file names, etc.).
- **Dynamic task mapping**
  Dynamic task mapping enables generating parallel tasks at runtime based on the output of a previous task.

The examples in this article are written using the TaskFlow API for convenience. You can generate DAGs dynamically using traditional syntax as well.

Dynamic DAGs generation

Creating a dynamically generated DAG is similar to the process of creating a single DAG. Since Airflow executes all Python code in the *dags* folder, you can execute any Python code that generates DAG objects.

The DAGs are generated every `min_file_process_interval` seconds, which by default is every 30 seconds. You can change the value of this parameter by editing the ***cfg_properties_template*** field via ADCM.

To edit the `min_file_process_interval` property via ADCM:

1. Go to the ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***Airflow2*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***airflow.cfg*** section and click ***cfg_properties_template***.
4. Set the desired value for `min_file_process_interval`.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Restart***.

CAUTION

If the value of the `min_file_process_interval` parameter is too low, many generated DAGs might overload the scheduler or a separate `dag-processor` process.

To create a dynamic DAG:

1. In the *dags* directory of your Airflow home, add a new Python file that will contain your DAG generator:

```
$ sudo vi /opt/airflow/dags/dag_generator.py
```

2. Add the code for the DAG generator:

```
from airflow import DAG
from airflow.decorators import task
from datetime import datetime   ①

for file in ("dev_data.csv", "test_data.csv", "prod_data.csv"):  ②
    dag_id = f"generated_dag_{file}"

    @dag(dag_id=dag_id, schedule="@daily",
    default_args=default_args, catchup=False,
    start_date=datetime(2025,1,1)) ③

    def create_dag(filename):

        @task ④
        def extract(filename):
            return filename

        @task
        def process(filename):
            return filename

        @task
        def load(filename):
            print(filename)
            return filename

        load(process(extract(filename))) ⑤

    create_dag(filename)  ⑥
```

① Import the dependencies.

② Create the loop for generating DAGs.

③ Create the DAG function.

④ Add tasks.

⑤ Define your task dependencies by calling the task functions.

⑥ Invoke the DAG function.

If you want to use variables to configure your code, it's better to use environment variables rather than Airflow variables.

Using Airflow variables in top-level code creates a connection to the metadata database, which can slow down parsing and place extra load on the database.

You can also import variables from one of several external sources, such as configuration files.

For example, you have two JSON configuration files in the *include/dag-configs/* directory of your Airflow home. The files have the following contents:

- *dev_config.json*:

```
{
    "dag_id": "development_dag",
    "schedule": "@daily",
    "input": "dev_data.csv"
}
```

- *prod_config.json*:

```
{
    "dag_id": "production_dag",
    "schedule": "@daily",
    "input": "prod_data.csv"
}
```

You can create a Python script that will generate separate DAG files for each configuration file using the DAG file you specified as a template.

To generate DAGs using a template:

1. In the Airflow home directory, add a new Python file that will contain your DAG template:

```
$ sudo vi /opt/airflow/include/templates/dag_template.py
```

2. Add the code that the generator will use as a template for DAGs. For example, you can use the code from the previous example and substitute the parameters you want to change with placeholders:

```
from airflow import DAG
from airflow.decorators import task
from datetime import datetime

@dag(dag_id = "dag_id_placeholder", schedule="schedule_placeholder",
catchup=False, start_date=datetime(2025,1,1)) ①

def create_dag(filename):

    @task
    def extract(filename):
        return filename

    @task
    def process(filename):
        return filename

    @task
    def load(filename):
        print(filename)
        return filename

    load(process(extract("input_placeholder"))) ②

create_dag(filename)
```

① Add placeholders for the `dag_id` and `schedule_interval` parameters.

② Add a placeholder for the input.

3. In the Airflow home directory, add a Python file that will generate DAGs using the template:

```
$ sudo vi /opt/airflow/include/scripts/dag_generator.py
```

4. Write the code for the DAG generator:

```
import json
import os
import shutil
import fileinput

config_filepath = "include/dag-configs/" ①
template_filepath = "include/templates/dag_template.py" ②

for filename in os.listdir(config_filepath): ③
    if filename.endswith('.json'):
        config = json.load(open(f"include/data/{filename}"))
        generated_dag = f"dags/generated_dag_{config['dag_id']}.py"
        shutil.copyfile(template_filepath, generated_dag)
        for line in fileinput.input(generated_dag, inplace=True):
            line = line.replace("dag_id_placeholder", config['dag_id'])
            line = line.replace("schedule_placeholder", config['schedule'])
            line = line.replace("input_placeholder", config['input'])
            print(line, end="")
```

① Specify the path to the config files.

② Specify the path to the template file.

③ Describe the logic of substituting parameter placeholders with correct values from the configs.

When the script is executed, it will check the *dag-configs* directory for available JSON configuration files and generate a DAG file for each configuration file based on the specified template. In the template, it will match the placeholders with the fields from the configuration files and will substitute the values.

Once the DAG files have been generated, Airflow will parse and load them as regular DAGs.

Dynamic task mapping

The above examples describe how to generate DAGs with different parameters, but the number of tasks for generated DAGs stays the same.

You can create tasks based on the unexpected output of the other task using dynamic task mapping.

This method employs the MapReduce framework and works using the `expand()` method, which creates tasks at runtime based on the output by multiplying the caller task using the provided arguments.

The `expand()` method is used with an operator and accepts a list or a dictionary of values as input. These values can also be an output of the other task in the `XComArg` format.

The example of dynamic task mapping:

```
from airflow.decorators import dag, task
from datetime import datetime

@dag(dag_id = "dynamic_task_dag", schedule="@daily", catchup=False, start_date=datetime(2025,1,1))
def create_tasks():

    @task
    def download_files(file: str): ①
        print(file)

    files = download_files.expand(file=["file_a", "file_b", "file_c"]) ②

    download_files(file)

create_tasks()
```

① Create a task function. For example, a function that prints a file.

② Call `expand()` and pass the list of files as an argument. The DAG will generate three tasks for each file.

The example of dynamic task mapping when the output of the previous task is unexpected:

```
from airflow.decorators import dag, task
from datetime import datetime
import random

@dag(dag_id = "dynamic_task_dag", schedule="@daily",
catchup=False, start_date=datetime(2025,1,1))

def create_tasks():

    @task
    def get_files():
        return ["file_{i}" for i in range(random.randint(3,5))] ①

    @task
    def download_files(file: str):
        print(file)

    files = download_files.expand(file=get_files()) ②

create_tasks()
```

① Create a function that has a random output.

② Call `expand()` and pass the output of the random function as an argument. The DAG will generate a task for each file.

Logging in Airflow

Daria Barysheva, Elena Kostyuchenko

Contents

To Table of Contents

Overview

Grep logs

Logging levels

Logging configuration

Overview

Airflow writes text logs used for analyzing errors that can occur while running DAGs. These logs are located in the */var/log/airflow/* directory of the Airflow server's host, but they are also available in the Airflow UIs.

Log paths

The full path to each DAG log file looks like this:

/var/log/airflow/dag_id=<DAG_ID>/run_id=<DAG_Run_ID>/task_id=<Task_ID>/<log_number>.log

where:

- `<DAG_ID>` is the DAG identifier.
- `<DAG_Run_ID>` is the identifier of the DAG run, which combines in itself the type of run and a timestamp. For example, *run_id=scheduled__2024-07-14T14:18:33.254657+00:00*.
- `<Task_ID>` is the task identifier.
- `<log_number>` is the sequence number of the log (begins with 1).

The full path to the process manager log files is */var/log/airflow/dag_processor_manager/*.

The full path to the scheduler logs is */var/log/airflow/scheduler/*.

To view Airflow logs on the host:

1. Connect to the Airflow server via SSH and run the following command:

```
$ ls -la /var/log/airflow/
```

The output looks similar to this:

```
total 12
drwxr-xr-x.  5 airflow airflow   77 Jul 15 14:18 .
drwxr-xr-x. 16 root    root    4096 Aug  5 08:44 ..
drwxr-xr-x. 13 airflow airflow 4096 Jul 30 10:12 dag_id=adcm_check
drwxr-xr-x.  2 airflow airflow  109 Jul 29 23:54 dag_processor_manager
drwxr-xr-x. 20 airflow airflow 4096 Aug  5 07:56 scheduler
```

2. View the desired log file:

```
$ cat /var/log/airflow/dag_id=adcm_check/run_id=manual__2024-07-15T14:18:43.743847+00:00/task_id=runme_1/attempt=1.log
```

The output is similar to this:

```
[2024-07-15T14:18:46.143+0000] {taskinstance.py:1103} INFO - Dependencies all met for dep_context=requeueable deps ti=
<TaskInstance: adcm_check.runme_1 manual__2024-07-15T14:18:43.743847+00:00 [queued]>
[2024-07-15T14:18:46.144+0000] {taskinstance.py:1308} INFO - Starting attempt 1 of 4
[2024-07-15T14:18:46.157+0000] {taskinstance.py:1327} INFO - Executing <Task(BashOperator): runme_1> on 2024-07-15 14:18:43.
743847+00:00
[2024-07-15T14:18:46.164+0000] {standard_task_runner.py:57} INFO - Started process 7082 to run task
[2024-07-15T14:18:46.168+0000] {standard_task_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'adcm_check', 'runme_1',
'manual__2024-07-15T14:18:43.743847+00:00', '--job-id', '12', '--raw', '--subdir', 'DAGS_FOLDER/adcm_check.py', '--cfg-path',
'/tmp/tmpr_3qshf4']
[2024-07-15T14:18:46.171+0000] {standard_task_runner.py:85} INFO - Job 12: Subtask runme_1
[2024-07-15T14:18:46.233+0000] {task_command.py:410} INFO - Running <TaskInstance: adcm_check.runme_1 manual__2024-07-15T14:18:43.
743847+00:00 [running]> on host elenas-adh3.ru-central1.internal
[2024-07-15T14:18:46.326+0000] {taskinstance.py:1545} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow'
AIRFLOW_CTX_DAG_ID='adcm_check' AIRFLOW_CTX_TASK_ID='runme_1' AIRFLOW_CTX_EXECUTION_DATE='2024-07-15T14:18:43.743847+00:00'
AIRFLOW_CTX_TRY_NUMBER='1' AIRFLOW_CTX_DAG_RUN_ID='manual__2024-07-15T14:18:43.743847+00:00'
```

Grep logs

You can search through the logs for specific information, like error messages. To do this, connect to the host with the logs you want to inspect and use a `grep` command.

For example:

```
$ cat /var/log/airflow/dag_processor_manager/dag_processor_manager.log | grep -i -A3 -B1 error | grep -i -A3 -B1 error
```

This command searches for messages containing the word `error` in the process manager log. The `-i` option allows you to ignore case distinctions. The `-A3 -B1` options expand the output to one line before and three lines after the line containing the error.

Example output:

```
File Path                                                                                                                  PID
Runtime      # DAGs    # Errors  Last Runtime    Last Run
-------------------------------------------------------------------------------------------------------------------------  -----
---------  --------  ----------  --------------  -------------------
/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_setup_teardown.py
0           0  0.03s           2024-08-05T10:38:25
/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_setup_teardown_taskflow.py
0           0  0.03s           2024-08-05T10:38:25
--
...
File Path                                                                                                                  PID
Runtime      # DAGs    # Errors  Last Runtime    Last Run
-------------------------------------------------------------------------------------------------------------------------  -----
---------  --------  ----------  --------------  -------------------
/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_setup_teardown.py
0           0  0.03s           2024-08-05T10:38:55
/opt/airflow/lib/python3.10/site-packages/airflow/example_dags/example_setup_teardown_taskflow.py
0           0  0.03s           2024-08-05T10:38:55
--
```

Logging levels

Airflow uses the standard Python logging framework and supports the following log levels (from least to most informative):

1. `CRITICAL` — reports on a serious error, indicating that the service itself may be unable to continue running.
2. `FATAL` — indicates that an operation can't continue execution and will terminate.
3. `ERROR` — notifies that a program is not working correctly or has stopped.
4. `WARN` — warns about potential problems. This doesn't mean that the service is not working but it raises a concern.
5. `INFO` — informs regarding the program lifecycle or state.
6. `DEBUG` — prints debugging information about the internal states of the program.

Enabling one level of logging will enable this level and all levels above it. For example, if you set the logging level to `ERROR`, then only errors, fatal and critical messages would get into the log files, but not `INFO`, `DEBUG`, and `WARN`.

Logging configuration

To change logging properties via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click on ***Airflow2***.
3. Select the required parameter and make the necessary changes.
4. Confirm changes by clicking ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

Airflow logging parameters:

- ***Logging level*** — the logging level for the Airflow service.
- ***Logging level for Flask-appbuilder UI*** — the logging level for Flask-appbuilder.
- ***cfg_properties_template*** — the Airflow configuration file that contains logging settings for tasks.

Logging parameters in the cfg_properties_template config

```
[logging]
base_log_folder = /var/log/airflow
remote_logging = False
remote_log_conn_id =
google_key_path =

remote_base_log_folder =
encrypt_s3_logs = False
{% endraw %}
logging_level = {{ services.airflow2.config.airflow_cfg.logging_level }}
{% raw -%}
celery_logging_level =
{% endraw %}
fab_logging_level = {{ services.airflow2.config.airflow_cfg.fab_logging_level }}
{% raw -%}
logging_config_class =
colored_console_log = True
colored_log_format = [%%(blue)s%%(asctime)s%%(reset)s] {%%(blue)s%%(filename)s:%%(reset)s%%(lineno)d} %%(log_color)s%%
(levelname)s%%(reset)s - %%(log_color)s%%(message)s%%(reset)s
colored_formatter_class = airflow.utils.log.colored_log.CustomTTYColoredFormatter
log_format = [%%(asctime)s] {%%(filename)s:%%(lineno)d} %%(levelname)s - %%(message)s
simple_log_format = %%(asctime)s %%(levelname)s - %%(message)s
task_log_prefix_template =
log_filename_template = dag_id={{ ti.dag_id }}/run_id={{ ti.run_id }}/task_id={{ ti.task_id }}/{%% if ti.map_index >= 0
%%}map_index={{ ti.map_index }}/{%% endif %%}attempt={{ try_number }}.log
log_processor_filename_template = {{ filename }}.log
dag_processor_manager_log_location = /var/log/airflow/dag_processor_manager/dag_processor_manager.log
task_log_reader = task
extra_logger_names =
```

Airflow service management via ADCM

Elena Kostyuchenko

To Table of Contents

The ADCM UI provides actions to manage the Airflow service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Airflow service are listed in the table below.

For more information about Airflow components, see Airflow.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Airflow components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

Airflow components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Get client configurations

Konstantin Alpashkin

Contents

To Table of Contents

- Overview
  - Exported configuration files
- Configuration export example

Starting with ADH 4.0.0, you can export client configurations of the services currently installed in an ADH cluster. The exported configurations can be used, for example, to configure client software that connects to ADH, or to configure the services of another ADH cluster.

Overview

The configuration export functionality is embedded into the ***Core configuration*** service. Particularly, the service has the ***Export client configurations*** action, which collects configurations of the currently installed services, archives them, and makes them available for download via an Nginx endpoint.

> **NOTE**
>
> The Configuration server and Hive Tez components must not be mapped on the same ADH host.

The configurations are exported as separate *.tar.gz* archives for each service. The list of exported services and files is presented below.

Exported configuration files

The ***Export client configurations*** action exports configurations for the following services.

Service	Configuration file(s)
Core configuration	core-site.xml ssl-client.xml log4j.properties.xml
HDFS	hdfs-site.xml log4j.properties.xml
YARN	yarn-env.sh yarn-site.xml mapred-site.xml
Hive	hive-site.xml hive-env.sh beeline-hs2-connection.xml beeline-log4j2.properties hive-log4j2.properties hive-log4j.properties hive-metastore-env.sh hive-server2-env.sh tez-site.xml
Solr	solr.xml solr-env.sh log4j2-console.xml log4j2.xml
HBase	hbase-site.xml hbase-env.sh log4j.properties
Flink	flink-conf.yaml flink-env.sh log4j-cli.properties log4j.properties
Impala	impalarc impala-shell-env.sh hive-site.xml
Spark	spark-defaults.conf spark-env.sh hive-site.xml log4j2.properties
Ozone	ozone-site.xml

> **NOTE**
>
> Sensitive data (passwords, secrets, JCEK keystores, etc.) are not included in the exported configuration files.

Configuration export example

The following steps show how to export the configurations of an ADH cluster:

1. In ADCM, go to ***Clusters*** → ***<cluster_name>*** → ***Services*** → ***Core configuration***.
2. Run the ***Export client configurations*** service action.
3. Go to the ***Core configuration*** → ***Info*** page and follow the ***Configuration server web interface*** link (*http://<configuration_server_host>:9998*). The web page displays a list of *tar.gz* archives with configurations for each service. For example:

```
../
all_configs.tar.gz   13-Aug-2025 16:06    7948
core.tar.gz          13-Aug-2025 16:06    4632
hdfs.tar.gz          13-Aug-2025 16:06    1494
yarn.tar.gz          13-Aug-2025 16:06    2486
```

   You can download an archive by clicking the archive link.

Core configuration service management via ADCM

Vladimir Adamenkov

To Table of Contents

The ADCM UI provides actions to manage the Core configuration service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Core configuration service are listed in the table below.

Core configuration service actions

Action	Description
Add/Remove components	Adds/removes the Core configuration component(s). Running this action opens the component-host mapping interface where you can distribute Core configuration components
Check	Checks the port availability of the Configuration server component
Export client configurations	Saves to archives all the client configurations (for all the services and for the cluster), without applying them on hosts
Restart	Restarts the service. When you run this action, the ***Apply configs from ADCM*** option is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM
Reinstall Core	Reinstalls the Core configuration service
Update Core configuration	Updates the *core-site.xml*, *hadoop-env.sh*, *ssl-server.xml*, and *ssl-client.xml* files
Start	Starts the service. When you run this action, the ***Apply configs from ADCM*** option is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service

The actions available for the Core configuration service components are listed in the table below.

Configuration server actions

Action	Description
Check	Runs component-specific tests to check the health of the given component
Restart	Restarts the component. When you run this action, the ***Apply configs from ADCM*** option is available. If it is set to `true`, all component configurations defined in ADCM settings will be applied during the component restart. Otherwise, the component restarts without applying configurations from ADCM
Start	Starts the component. When you run this action, the ***Apply configs from ADCM*** option is available. If it is set to `true`, all component configurations defined in ADCM settings will be applied on the component startup. Otherwise, the component starts without applying configurations from ADCM
Stop	Stops the component
Export client configurations	Saves to archives all the client configurations (for all the services and for the cluster), without applying them on hosts

Flink UI overview

Konstantin Alpashkin

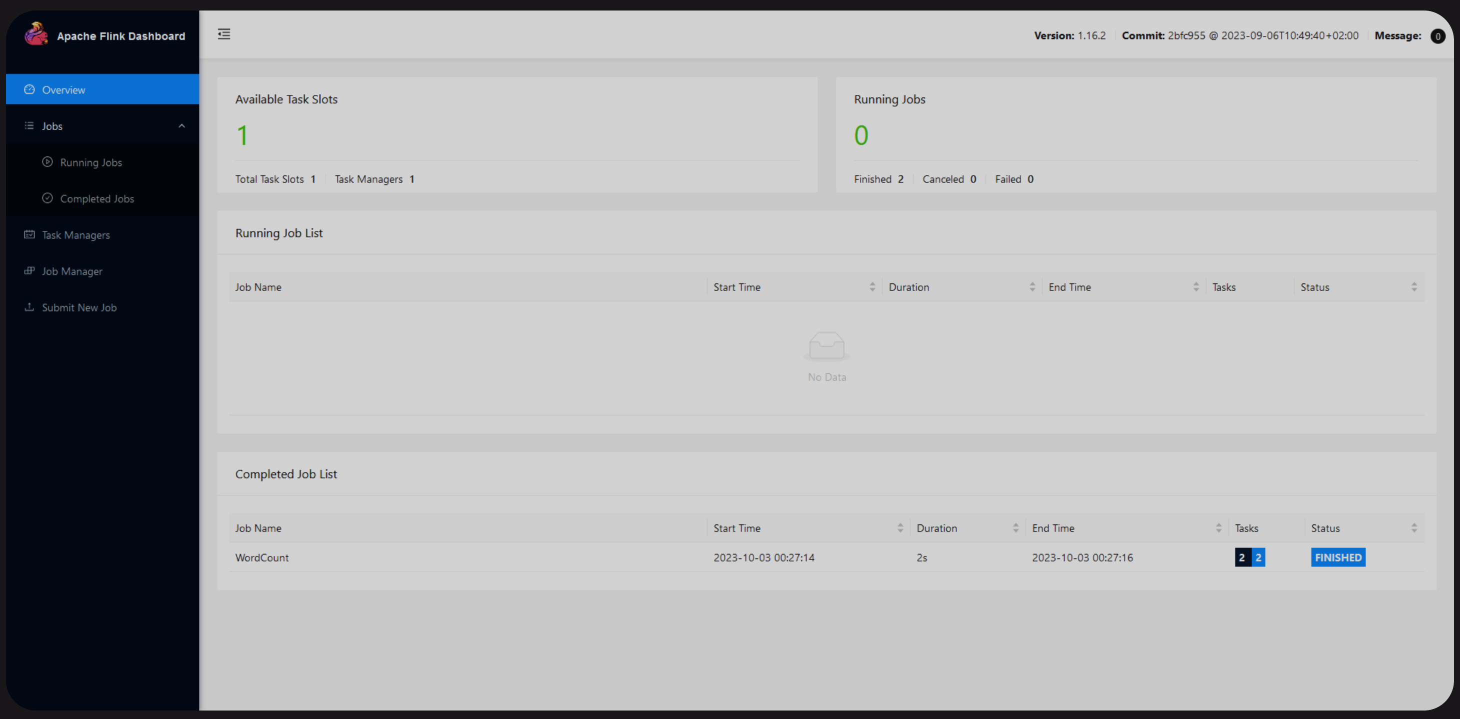
Contents

To Table of Contents

Flink History Server web UI

Flink Dashboard is a built-in monitoring interface for Flink applications. You can monitor your running, completed, and stopped Flink jobs on the dashboard. You reach the Flink UI via ADCM using the main page of the Flink service.

The Flink UI acts as a single point for monitoring all the jobs running on a YARN cluster. It shows all the running, failed, and finished jobs. The URL to the Flink UI looks as follows: *http://FQDN:8081*. The Flink UI is shown below.


Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Task Managers
Job Manager
Submit New Job
No Data

Version: 1.16.2 **Commit:** 2bfc955 @ 2023-09-06T10:49:40+02:00 **Message:** 0

Available Task Slots
1
Total Task Slots 1 Task Managers 1

Running Jobs
0
Finished 2 Canceled 0 Failed 0

Running Job List

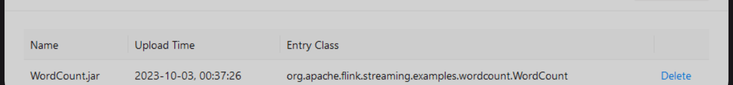
Job Name	Start Time	Duration	End Time	Tasks	Status

Completed Job List

Job Name	Start Time	Duration	End Time	Tasks	Status
WordCount	2023-10-03 00:27:14	2s	2023-10-03 00:27:16	2 2	FINISHED

Flink Dashboard

You can add a jar file on the ***Submit new Job*** page.

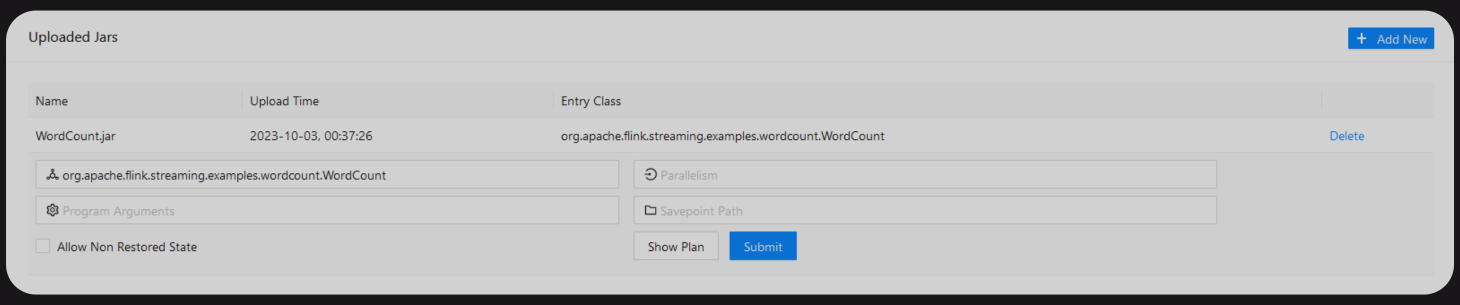


Uploaded Jars

Name	Upload Time	Entry Class	
WordCount.jar	2023-10-03, 00:37:26	org.apache.flink.streaming.examples.wordcount.WordCount	Delete

Submit new Job

After uploading the jar file you can run it with parameters.


Add New
Show Plan
Submit

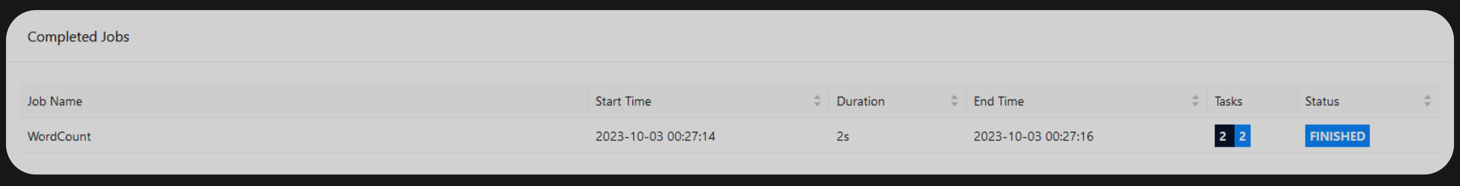
Uploaded Jars

Name	Upload Time	Entry Class	
WordCount.jar	2023-10-03, 00:37:26	org.apache.flink.streaming.examples.wordcount.WordCount	Delete

org.apache.flink.streaming.examples.wordcount.WordCount
Parallelism
Program Arguments
Savepoint Path
☐ Allow Non Restored State

Uploaded Jars

You can specify the parameters and submit the job. When job is completed, you will see all the details on the ***Completed jobs*** page.

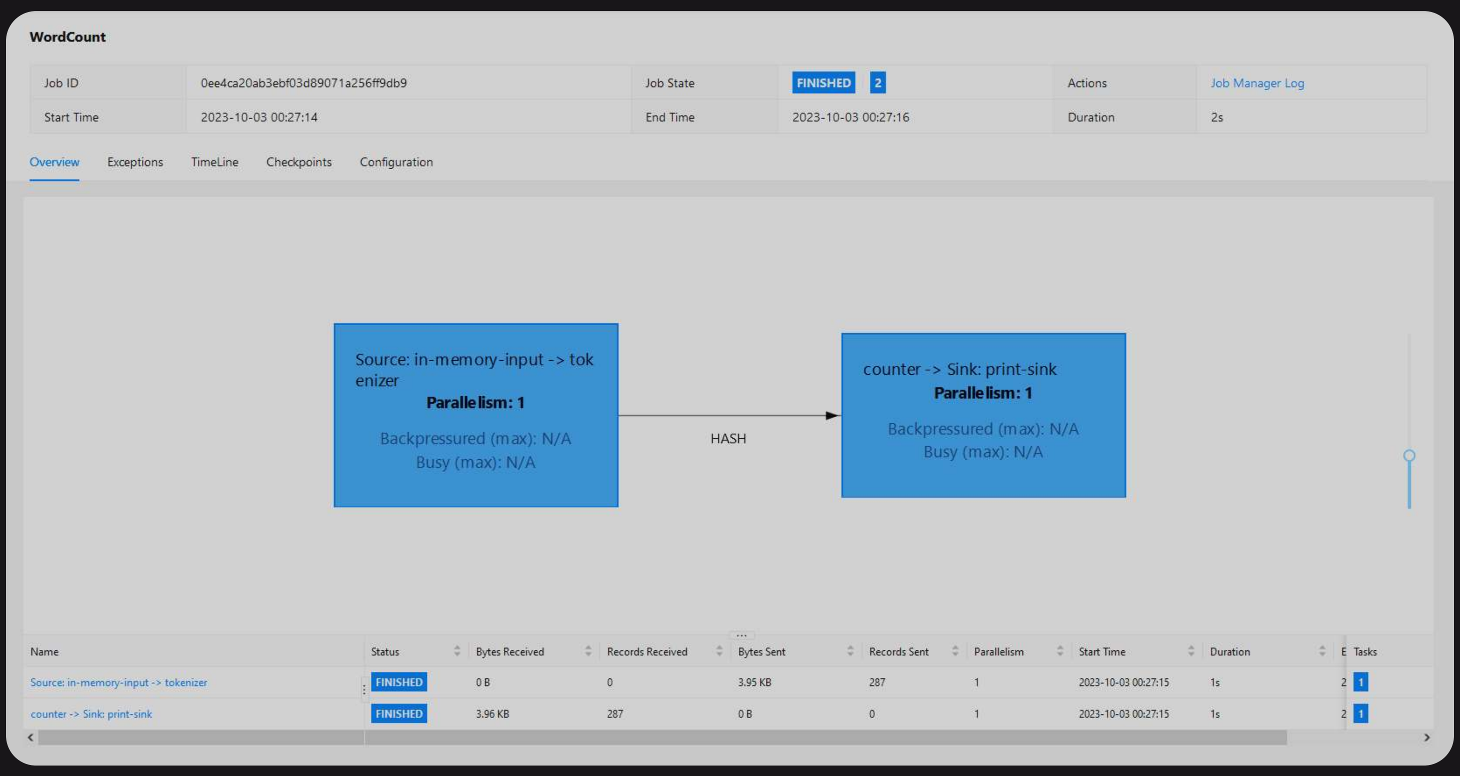


Completed Jobs

Job Name	Start Time	Duration	End Time	Tasks	Status
WordCount	2023-10-03 00:27:14	2s	2023-10-03 00:27:16	2 2	FINISHED

Completed Jobs

If you click on ***Completed Jobs***, you get the detailed jobs overview.


Overview
Exceptions
TimeLine
Checkpoints
Configuration
Source: in-memory-input -> tokenizer
Parallelism: 1
Backpressured (max): N/A
Busy (max): N/A
HASH
counter -> Sink: print-sink
Parallelism: 1
Backpressured (max): N/A
Busy (max): N/A

WordCount

Job ID	0ee4ca20ab3ebf03d89071a256ff9db9	Job State	FINISHED 2	Actions	Job Manager Log
Start Time	2023-10-03 00:27:14	End Time	2023-10-03 00:27:16	Duration	2s

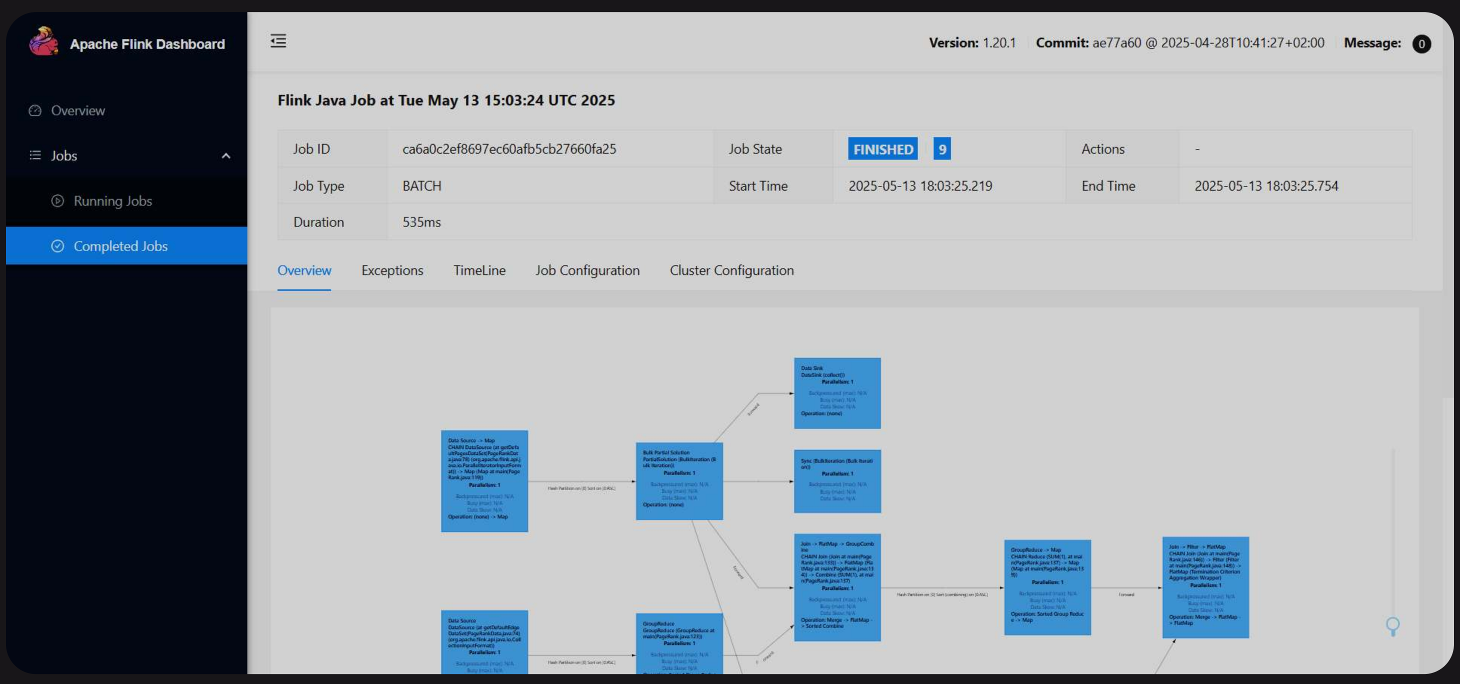
Name	Status	Bytes Received	Records Received	Bytes Sent	Records Sent	Parallelism	Start Time	Duration	E	Tasks
Source: in-memory-input -> tokenizer	FINISHED	0 B	0	3.95 KB	287	1	2023-10-03 00:27:15	1s	2	1
counter -> Sink: print-sink	FINISHED	3.96 KB	287	0 B	0	1	2023-10-03 00:27:15	1s	2	1

Jobs Overview

Flink History Server web UI

The Flink History Server component provides a web interface, where you can view statistics about completed Flink jobs. The up-to-date URL of the Flink History Server UI can be found in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Flink*** → ***Info***).

Flink History Server web UI is shown below.


Apache Flink Dashboard
Overview
Jobs
Running Jobs
Completed Jobs
Overview
Exceptions
TimeLine
Job Configuration
Cluster Configuration

Version: 1.20.1 **Commit:** ae77a60 @ 2025-04-28T10:41:27+02:00 **Message:** 0

Flink Java Job at Tue May 13 15:03:24 UTC 2025

Job ID	ca6a0c2ef8697ec60afb5cb27660fa25	Job State	FINISHED 9	Actions	-
Job Type	BATCH	Start Time	2025-05-13 18:03:25.219	End Time	2025-05-13 18:03:25.754
Duration	535ms				

Flink History Server web UI

Connect to Flink via CLI

Konstantin Alpashkin

To Table of Contents

Flink provides a command-line interface to run programs packaged as Java archives (jar) and to control their execution. To submit a new job means to upload the job's jar file and related dependencies to a running Flink cluster and to execute it.

Flink releases come with example jobs available in the */usr/lib/flink/examples/* directory. To deploy the example word count job to the running cluster, execute the following command:

```
$ flink run /usr/lib/flink/examples/streaming/WordCount.jar
```

The other examples can be started in a similar way. Many examples run without passing any arguments to them, and simply using build-in data. To run *WordCount.jar* with real data, you have to pass the path to the input data:

```
$ ./bin/flink run ./examples/batch/WordCount.jar --input /path/to/some/text/data --output /path/to/result
```

NOTE

Non-local file systems require a schema prefix, such as `hdfs://`.

You can verify the output by viewing the logs:

```
$ tail log/flink-*-taskexecutor-*.out
```

Sample output:

```
(nymph,1)
(in,3)
(thy,1)
(orisons,1)
(be,4)
(all,2)
(my,1)
(sins,1)
(remember,1)
(d,4)
```

PyFlink usage examples

Konstantin Alpashkin

Contents

To Table of Contents

- PyFlink APIs: DataStream vs Table
- Table API
  - Create a source table
  - Queries and table operations
  - Write data to a sink table
- DataStream API
  - Create execution environment
  - Ingest data from a source
  - Transformation operators
  - Write data to a sink
- Dependency management
  - JAR dependencies
  - Python dependencies

PyFlink is a Python API for Flink that allows building scalable batch and streaming pipelines using Python. Under the hood, PyFlink uses Py4J that connects to the target Flink JVM and interacts with the Flink objects.

PyFlink is distributed as a `pip` module. ADH comes with a Python interpreter that already has PyFlink installed, as well as all the required dependencies. This interpreter is located at */opt/pyflink-python/* on ADH hosts with Flink components installed. Using this interpreter, you can run PyFlink applications as regular Python scripts, for example:

```
$ source /opt/pyflink-python/bin/activate
$ /opt/pyflink-python/bin/python3 create_table.py
```

PyFlink APIs: DataStream vs Table

PyFlink provides two different APIs, depending on the abstraction level you need:

- Table API. Allows running relational queries, similarly to using SQL or working with tabular data in Python.
- DataStream API. Provides a lower-level control over Flink functions, like state, checkpoints, processing time, etc. Allows building more complex stream processing logic.

This section provides examples on working with both APIs. Additional PyFlink usage examples can be found on the Flink GitHub page. The detailed PyFlink API reference is available in Flink documentation.

Table API

PyFlink Table API is a unified relational API for batch and stream processing. It allows using an SQL-like domain-specific language for defining table queries and operations. The queries are executed with the same semantics on bounded batch data sets and unbounded streams.

A typical application starts with creating an execution environment object (TableEnvironment). This is the core component of any Table API application and is used to configure and execute table-based operations. Here's an example of instantiating the environment object:

```
from pyflink.table import TableEnvironment, EnvironmentSettings

env_settings = EnvironmentSettings.in_streaming_mode() ①
t_env = TableEnvironment.create(env_settings) ②
t_env.get_config().set("parallelism.default", "1") ③
```

① Sets the execution mode. Can be either batch or streaming.

② Creates a `TableEnvironment` object.

③ Sets Flink configuration properties (parallelism, recovery, state, etc.) on the application level.

Create a source table

Once `TableEnvironment` is instantiated, you can use it to create Flink tables. A `Table` object describes a pipeline of data transformations. It does not store any data itself, but describes where to get data (source table), or where to store the data after the processing (sink table).

Below are several examples of creating Flink source tables.

From a collection

The following snippet creates source tables from an in-memory collection.

```
from pyflink.table import EnvironmentSettings, TableEnvironment
from pyflink.table.expressions import col

env_settings = EnvironmentSettings.in_streaming_mode()
t_env = TableEnvironment.create(env_settings)
t_env.get_config().set("parallelism.default", "1")

mock_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-02"),
    (3, 1003, 70.00, "2025-01-03"),
]
table_1 = t_env.from_elements(elements=mock_data, ①
                              schema=['txn_id', 'acc_id', 'txn_value', 'txn_date'])
print("Inferred data types for table_1:")
table_1.print_schema()

from pyflink.table import DataTypes
table_2 = t_env.from_elements(elements=mock_data, ②
                              schema=DataTypes.ROW([DataTypes.FIELD("txn_id", DataTypes.TINYINT()),
                                                    DataTypes.FIELD("acc_id", DataTypes.INT()),
                                                    DataTypes.FIELD("txn_value", DataTypes.DOUBLE()),
                                                    DataTypes.FIELD("txn_date", DataTypes.STRING())]))
print(f"The data type of 'txn_date': {table_2.get_schema().get_field_data_type('txn_date')}")
table_2.print_schema()
table_2 = table_2.select(
    col("txn_id"),
    col("acc_id"),
    col("txn_value"),
    col("txn_date").cast(DataTypes.DATE()).alias("txn_date")
)
print(f"Data type of 'txn_date' after casting:"
      f"{table_2.get_schema().get_field_data_type('txn_date')}")
print("SELECT * FROM table_2:")
table_2.execute().print() ③
```

① Creates a table, whose schema infers data types from the collection.

② Creates a table with explicit data types in schema.

③ Prints the table's content.

Sample output

```
Inferred data types for table_1:
(
  `txn_id` BIGINT,
  `acc_id` BIGINT,
  `txn_value` DOUBLE,
  `txn_date` STRING
)
The data type of 'txn_date': VARCHAR
(
  `txn_id` TINYINT,
  `acc_id` INT,
  `txn_value` DOUBLE,
  `txn_date` STRING
)
Data type of 'txn_date' after casting:DATE
SELECT * FROM table_2:
+----+--------+-------------+--------------------------------+------------+
| op | txn_id |      acc_id |                      txn_value |   txn_date |
+----+--------+-------------+--------------------------------+------------+
| +I |      1 |        1001 |                          100.0 | 2025-01-01 |
| +I |      2 |        1002 |                           20.0 | 2025-01-02 |
| +I |      3 |        1003 |                           70.0 | 2025-01-03 |
+----+--------+-------------+--------------------------------+------------+
3 rows in set
```

From a Kafka topic

The following snippet creates a source table subscribed to a Kafka topic via the TableDescriptor.

```
from pyflink.table import EnvironmentSettings, TableEnvironment, Schema, DataTypes, TableDescriptor

env_settings = EnvironmentSettings.in_streaming_mode()
t_env = TableEnvironment.create(env_settings)
t_env.get_config().set("parallelism.default", "1")
t_env.get_config().set("pipeline.jars", "file:///home/konstantin/pyflink_demo/flink-connector-kafka-3.3.0-1.19.jar") ①

t_env.create_temporary_table( ②
    "kafka_source_table",
    TableDescriptor.for_connector("kafka")
    .schema(Schema.new_builder()
            .column("txn_id", DataTypes.INT())
            .column("acc_id", DataTypes.INT())
            .column("txn_value", DataTypes.DOUBLE())
            .column("txn_date", DataTypes.DATE())
            .build())
    .option("properties.bootstrap.servers", "<host>:<port>")
    .option("topic", "transactions_test")
    .option("properties.group.id", "txn_group")
    .option("scan.startup.mode", "earliest-offset")
    .option("format", "json")
    .option("json.fail-on-missing-field", "false")
    .option("json.ignore-parse-errors", "true")
    .build()
)
```

① Adds the Kafka connector JAR to the PyFlink job context. To get an up-to-date Kafka connector version, see Flink documentation.

② Creates a temporary table that ingests data from a Kafka topic.

From a file using DDL

The following snippet uses SQL expressions to create a table from a CSV file.

```
from pyflink.table import EnvironmentSettings, TableEnvironment

env_settings = EnvironmentSettings.in_streaming_mode()
t_env = TableEnvironment.create(env_settings)

t_env.execute_sql("""
CREATE TABLE src_table_from_csv (
    txn_id INT,
    acc_id INT,
    txn_value INT,
    txn_date STRING
) WITH (
    'connector' = 'filesystem', ①
    'format' = 'csv',
    'path' = 'test.csv',
    'csv.field-delimiter' = ','
)
""")

table = t_env.from_path("src_table_from_csv")
table.execute().print()
```

① Uses the built-in connector for working with files.

Sample output

```
+----+-------------+-------------+-------------+--------------------------------+
| op |      txn_id |      acc_id |   txn_value |                       txn_date |
+----+-------------+-------------+-------------+--------------------------------+
| +I |           1 |        1001 |          20 |                     2025-01-02 |
| +I |           2 |        1002 |         110 |                     2025-01-01 |
| +I |           3 |        1003 |          23 |                     2025-01-01 |
| +I |           4 |        1002 |          50 |                     2025-01-03 |
| +I |           5 |        1001 |          78 |                     2025-01-02 |
+----+-------------+-------------+-------------+--------------------------------+
```

For information on other ways of creating tables using Table API, see Flink documentation.

Queries and table operations

The `Table` object provides numerous methods for executing relational operations. The up-to-date API reference is available in the Flink documentation. The following snippet covers major operations.

```
from pyflink.table import EnvironmentSettings, TableEnvironment
from pyflink.table import DataTypes
from pyflink.table.expressions import col, call

env_settings = EnvironmentSettings.in_batch_mode() ①
t_env = TableEnvironment.create(env_settings)

txn_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-01"),
    (3, 1002, 70.00, "2025-01-01"),
    (4, 1001, 40.00, "2025-01-02"),
    (5, 1003, 50.00, "2025-01-01"),
    (6, 1001, 10.00, "2025-01-02"),
]

schema = DataTypes.ROW([
    DataTypes.FIELD("txn_id", DataTypes.INT()),
    DataTypes.FIELD("acc_id", DataTypes.INT()),
    DataTypes.FIELD("txn_value", DataTypes.DOUBLE()),
    DataTypes.FIELD("txn_date", DataTypes.STRING())
])

t_txns = t_env.from_elements(txn_data, schema)
print("Added table column:")
t_with_net_value = t_txns.add_columns( ②
    (col("txn_value") * 0.87).alias("net_value")
)
t_with_net_value.execute().print()

t_result = t_txns \ ③
    .filter(col("txn_date") != "2025-01-02") \
    .group_by(col("acc_id")) \
    .select(
    col("acc_id"),
    call("sum", col("txn_value")).alias("total_txn_value")
)
print("Filtered, groupped, and aggregated: ")
t_result.execute().print()

invoice_data = [
    (1, 1, True),
    (2, 4, False),
    (3, 3, True),
    (4, 2, True),
    (5, 6, False),
    (5, 5, True),
]
schema_i = DataTypes.ROW([
    DataTypes.FIELD("invoice_id", DataTypes.INT()),
    DataTypes.FIELD("tx_id", DataTypes.INT()),
    DataTypes.FIELD("is_confirmed", DataTypes.BOOLEAN()),
])

t_invoices = t_env.from_elements(invoice_data, schema_i)
t_joined = t_invoices.join(t_txns).where(col("tx_id") == col("txn_id")) \ ④
    .select(col("txn_id"),
            col("acc_id"),
            col("invoice_id"),
            col("is_confirmed"))
print("JOINed tables:")
t_joined.execute().print()
```

① Sets the execution mode to batch.

② Adds a column to the table.

③ Applies a chain of operators and aggregation functions, like `filter()`, `group_by()`, `sum()`.

④ Applies a Join operation.

Sample output

```
Added table column:
+-------------+-------------+--------------------------------+--------------------------------+-------------------------------
-+
|      txn_id |      acc_id |                      txn_value |                       txn_date |                      net_value
|
+-------------+-------------+--------------------------------+--------------------------------+-------------------------------
-+
|           1 |        1001 |                          100.0 |                     2025-01-01 |                           87.0
|
|           2 |        1002 |                           20.0 |                     2025-01-01 |                           17.4
|
|           3 |        1002 |                           70.0 |                     2025-01-01 |                           60.9
|
|           4 |        1001 |                           40.0 |                     2025-01-02 |                           34.8
|
|           5 |        1003 |                           50.0 |                     2025-01-01 |                           43.5
|
|           6 |        1001 |                           10.0 |                     2025-01-02 |                            8.7
|
+-------------+-------------+--------------------------------+--------------------------------+-------------------------------
-+
6 rows in set
Filtered, groupped, and aggregated:
+-------------+--------------------------------+
|      acc_id |                total_txn_value |
+-------------+--------------------------------+
|        1003 |                           50.0 |
|        1002 |                           90.0 |
|        1001 |                          100.0 |
+-------------+--------------------------------+
3 rows in set
JOINed tables:
+-------------+-------------+-------------+--------------+
|      txn_id |      acc_id |  invoice_id | is_confirmed |
+-------------+-------------+-------------+--------------+
|           5 |        1003 |           5 |         TRUE |
|           2 |        1002 |           4 |         TRUE |
|           3 |        1002 |           3 |         TRUE |
|           6 |        1001 |           5 |        FALSE |
|           1 |        1001 |           1 |         TRUE |
|           4 |        1001 |           2 |        FALSE |
+-------------+-------------+-------------+--------------+
6 rows in set
```

Alternatively, you can run table operations using SQL, for example:

```
from pyflink.table import EnvironmentSettings, TableEnvironment

env_settings = EnvironmentSettings.in_streaming_mode()
table_env = TableEnvironment.create(env_settings)

table_env.execute_sql("""
    CREATE TEMPORARY TABLE t_txns (
        txn_id INT,
        acc_id INT,
        txn_value DOUBLE,
        txn_date STRING
    ) WITH (
        'connector' = 'filesystem',
        'path' = 'test.csv',
        'format' = 'csv'
    )
""")

table_env.execute_sql("""
    CREATE TABLE t_sink_print (
        acc_id INT,
        max_txn_value DOUBLE
    ) WITH (
        'connector' = 'print'
    )
""")

table_env.execute_sql("""
    INSERT INTO t_sink_print
    SELECT acc_id, MAX(txn_value) AS max_txn_value
    FROM t_txns
    GROUP BY acc_id
""").wait()
```

Sample output

```
15> +I[1002, 110.0]
3> +I[1001, 20.0]
2> +I[1003, 23.0]
3> -U[1001, 20.0]
3> +U[1001, 78.0]
```

Write data to a sink table

The following example creates sink tables that store data in external locations:

```
from pyflink.table import EnvironmentSettings, TableEnvironment
from pyflink.table import DataTypes

env_settings = EnvironmentSettings.in_streaming_mode()
t_env = TableEnvironment.create(env_settings)
t_env.get_config().set("pipeline.jars",
                       "file:///home/konstantin/flink-sql-connector-kafka-3.3.0-1.19.jar")
mock_data = [
    (1, 1001, 50.75, "2025-01-01"),
    (2, 1002, 70.00, "2025-01-02"),
    (3, 1001, 20.25, "2025-01-02")
]
t_txns = t_env.from_elements(
    elements=mock_data,
    schema=DataTypes.ROW([
        DataTypes.FIELD("txn_id", DataTypes.INT()),
        DataTypes.FIELD("acc_id", DataTypes.INT()),
        DataTypes.FIELD("txn_value", DataTypes.DOUBLE()),
        DataTypes.FIELD("txn_date", DataTypes.STRING())
    ])
)
t_env.create_temporary_view("txn_table", t_txns) ①

t_env.execute_sql(""" ②
    CREATE TABLE sink_table (
        txn_id INT,
        acc_id INT,
        txn_value DOUBLE,
        txn_date STRING
    ) WITH (
        'connector' = 'print' ③
    )
""")

t_env.execute_sql(""" ④
    INSERT INTO sink_table
    SELECT * FROM txn_table
    WHERE `acc_id` = 1002
""").wait()

t_env.create_temporary_table( ⑤
    'kafka_sink',
    TableDescriptor.for_connector('kafka')
    .schema(Schema.new_builder()
            .column('txn_id', DataTypes.INT())
            .column('acc_id', DataTypes.INT())
            .column('txn_value', DataTypes.DOUBLE())
            .column('txn_date', DataTypes.STRING())
            .build())
    .option('topic', 'transactions_test')
    .option('properties.bootstrap.servers', 'localhost:9092')
    .format(FormatDescriptor.for_format('json')
            .build())
    .build())

t_env.execute_sql("""
    INSERT INTO kafka_sink
    SELECT * FROM txn_table
    WHERE `acc_id` = 1002
""").wait()
```

① Creates a temporary view from a table.

② Creates a sink table.

③ Forwards all table data to STDOUT.

④ Invokes the `INSERT` query that triggers spilling data to the sink location. Since the `INSERT` operation runs asynchronously in a separate thread, `wait()` is used to wait for results forwarded to STDOUT.

⑤ Creates another sink table, which uses a Kafka connector to write table data to the Kafka topic.

DataStream API

PyFlink DataStream API provides a lower-level control over Flink features, like state management, processing time, checkpoints, recovery, etc.

The central API object is a DataStream, which can be thought of as an immutable collection that travels through transformation stages. A typical application comprises the following steps:

1. Create an execution environment.
2. Read data from a source and produce one or more DataStream objects.
3. Direct the DataStream(s) through one or more transformation operators. Each operator consumes an input DataStream and produces a new one with modified data.
4. Save the DataStream content to an external location using sinks.

Create execution environment

The entry point of a DataStream API application is the execution environment object (StreamExecutionEnvironment). The following example shows how to instantiate one.

```
menv = StreamExecutionEnvironment.get_execution_environment() ①
menv.set_parallelism(1) ②
menv.set_runtime_mode(RuntimeExecutionMode.STREAMING)

# data ingest, transformation, and sink logic

menv.execute() ③
```

① Creates a `StreamExecutionEnvironment` object.

② Sets configuration properties like parallelism, retries, serializer options, and so on.

③ Triggers the job execution. You can also use execute_async() to run a job asynchronously and retrieve job results later.

Once the environment object is created, you can use it to set job properties, define sources, sinks, operators, and trigger job execution.

Ingest data from a source

Using `StreamExecutionEnvironment`, you can define Flink sources to ingest data into the Flink app. Several ways of defining sources are shown below.

From a collection

The following example creates a `DataStream` using an in-memory collection as a source.

```
from pyflink.common import Types, Configuration
from pyflink.datastream import StreamExecutionEnvironment

config = Configuration()
config.set_string("execution.runtime-mode", "STREAMING")

menv = StreamExecutionEnvironment.get_execution_environment()
menv.set_parallelism(1)

mock_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-02"),
    (3, 1003, 70.00, "2025-01-03"),
]
mds = menv.from_collection(mock_data, type_info=Types.ROW_NAMED(
        ["txn_id", "acc_id", "txn_value", "txn_date"],
        [Types.INT(), Types.INT(), Types.DOUBLE(), Types.STRING()]))
mds.print()

menv.execute()
```

Sample output

```
+I[1,1001,100.0,2025-01-01]
+I[2,1002,20.0,2025-01-02]
+I[3,1003,70.0,2025-01-03]
```

TIP

Tokens like `+I`, `+U`, `-U`, and `-D` indicate changelog row operations, namely:

- `+I` — a new row was inserted.
- `+U` — the previous version of a row (to be removed).
- `-U` — the new version of the row (to be inserted).
- `-D` — a row was deleted.

From a Kafka topic

The following snippet creates a Kafka source by using a connector for Kafka.

```
from pyflink.common import SimpleStringSchema, WatermarkStrategy
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.datastream.connectors.kafka import KafkaSource, KafkaOffsetsInitializer

menv = StreamExecutionEnvironment.get_execution_environment()
menv.add_jars("file:///home/konstantin/pyflink_demo/flink-sql-connector-kafka-3.3.0-1.19.jar") ①

source_kafka = KafkaSource.builder() \ ②
    .set_bootstrap_servers("<host>:<port>") \
    .set_topics("m_topic") \
    .set_group_id("my_group") \
    .set_starting_offsets(KafkaOffsetsInitializer.earliest()) \
    .set_value_only_deserializer(SimpleStringSchema()) \
    .build()
mds = menv.from_source(source_kafka, ③
                      WatermarkStrategy.for_monotonous_timestamps(),
                      "Test Kafka source")

menv.execute()
```

① Adds the Kafka connector JAR to the PyFlink job context. To get an up-to-date Kafka connector version, see Flink documentation.

② Creates a Kafka source by connecting to a Kafka broker.

③ Creates a `DataStream` from the Kafka source.

From a file

The following snippet creates a `DataStream` from a CSV file, treating each line as a stream record.

```
from pyflink.common import WatermarkStrategy
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.datastream.connectors.file_system import FileSource, StreamFormat

menv = StreamExecutionEnvironment.get_execution_environment()
file_source = FileSource.for_record_stream_format(
    StreamFormat.text_line_format(), ①
    "test.csv"
).build()
wm_strategy = WatermarkStrategy.for_monotonous_timestamps()

ds = menv.from_source(source=file_source, ②
                     source_name="test_file_source",
                     watermark_strategy=wm_strategy,

)
ds.print()
menv.execute()
```

① Uses a built-in StreamFormat, which tells Flink to read files line by line, treating each line as a single record.

② Uses a built-in watermark, which assumes DataStream records (lines) never arrive out of order.

Sample output

```
10> 1,1001,20,2025-01-02
10> 2,1002,110,2025-01-01
10> 3,1003,23,2025-01-01
10> 4,1002,50,2025-01-03
10> 5,1001,78,2025-01-02
```

TIP

`10>` indicates a task ID or a subtask index.

Transformation operators

PyFlink provides numerous operators like `map()`, `filter()`, `reduce()`, etc. Each operator consumes a `DataStream`, applies transformations on the data, and emits a new `DataStream` object. Basic operations are covered below:

```
from pyflink.common import Types, Row
from pyflink.datastream import StreamExecutionEnvironment, MapFunction

menv = StreamExecutionEnvironment.get_execution_environment()
menv.set_parallelism(1)
mock_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-02"),
    (3, 1003, 70.00, "2025-01-03"),
]
ds = menv.from_collection(mock_data, type_info=Types.ROW_NAMED(
    ["txn_id", "acc_id", "txn_value", "txn_date"],
    [Types.INT(), Types.INT(), Types.DOUBLE(), Types.STRING()]))
print("Original DataStream:")
ds.print()

ds_filtered = ds.filter(lambda txn: txn["txn_value"] > 50.0) ①
print("Filtered DataStream:")
ds_filtered.print()

ds_mapped = ds.map( ②
    lambda txn: Row(
        txn["txn_id"],
        9999 if txn["acc_id"] == 1003 else txn["acc_id"],
        txn["txn_value"],
        txn["txn_date"]
    ), output_type=Types.ROW_NAMED(
        ["txn_id", "acc_id", "txn_value", "txn_date"],
        [Types.INT(), Types.INT(), Types.DOUBLE(), Types.STRING()]))
print("Lambda-mapped DataStream:")
ds_mapped.print()

class UpdateAccID(MapFunction): ③
    def map(self, value):
        txn_id, acc_id, txn_value, txn_date = value
        if acc_id < 2000:
            acc_id = acc_id * 2
        return (Row(txn_id, acc_id, txn_value, txn_date))

ds_mapped_class = ds.map(UpdateAccID(), ④
                         output_type=Types.ROW_NAMED(
                             ["txn_id", "acc_id", "txn_value", "txn_date"],
                             [Types.INT(), Types.INT(), Types.FLOAT(), Types.STRING()]
                         ))
print("MapFunction-mapped DataStream:")
ds_mapped_class.print()

menv.execute()
```

① Applies the filter() transformation on a DataStream.

② Applies the map() transformation using a lambda function.

③ A custom `MapFunction` implementation to be used by the map operator. The `map()` method is called for each element of the `DataStream`.

④ Applies the `map()` transformation using a custom `MapFunction` implementation.

Sample output

```
Original DataStream:
+I[1,1001,100.0,2025-01-01]
+I[2,1002,20.0,2025-01-02]
+I[3,1003,70.0,2025-01-03]

Filtered DataStream:
+I[1,1001,100.0,2025-01-01]
+I[3,1003,70.0,2025-01-03]

Lambda-mapped DataStream:
+I[1,1001,100.0,2025-01-01]
+I[2,1002,20.0,2025-01-02]
+I[3,9999,70.0,2025-01-03]

Custom MapFunction-mapped DataStream:
+I[1,2002,100.0,2025-01-01]
+I[2,2004,20.0,2025-01-02]
+I[3,2006,70.0,2025-01-03]
```

NOTE

The output lines will be shuffled since every `DataStream` operator prints to STDOUT from its own thread.

Write data to a sink

After the computation steps are complete, you can direct the results to an external location using sinks.

Write to a file

The following snippet spills the `DataStream` content to a file sink using the FileSystem connector.

```
from pyflink.common import Types, Encoder
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.datastream.connectors.file_system import FileSink, RollingPolicy

OUTPUT_PATH = "file:///home/konstantin/pyflink_demo/fs_sink"
menv = StreamExecutionEnvironment.get_execution_environment()
menv.set_parallelism(1)
mock_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-02"),
    (3, 1003, 70.00, "2025-01-03"),
]
mds = menv.from_collection(mock_data, type_info=Types.ROW_NAMED(
    ["txn_id", "acc_id", "txn_value", "txn_date"],
    [Types.INT(), Types.INT(), Types.DOUBLE(), Types.STRING()]))
fs_sink = FileSink.for_row_format(OUTPUT_PATH, ①
                                  Encoder.simple_string_encoder("UTF-8")) \
    .with_rolling_policy(RollingPolicy.default_rolling_policy( ②
    part_size=1024 ** 3,
    rollover_interval=15 * 60 * 1000,
    inactivity_interval=5 * 60 * 1000)) \
    .build()

mds.sink_to(fs_sink) ③
menv.execute()
```

① Creates a `FileSink` sink using the row-encoded format.

② Sets a rolling policy.

③ Flushes the `DataStream` records to the sink.

This Flink job creates a directory specified by the `OUTPUT_PATH` location, for example:

```
/home/konstantin/pyflink_demo/fs_sink
└─ /2025-04-08--13
    └─ part-4a8b2a20-187d-4ad6-ae7a-8f0e6dc880c5-0
```

Every `DataStream` record is written to a target file as a new line:

```
+I[1, 1001, 100.0, 2025-01-01]
+I[2, 1002, 20.0, 2025-01-02]
+I[3, 1003, 70.0, 2025-01-03]
```

Write to an ORC file

To write binary data (ORC, Parquet, Avro, etc.) to a file, use bulk-encoded formats. A sample job that writes ORC bytes using `FileSink` is below:

```
from pyflink.common import Types, Configuration
from pyflink.datastream import StreamExecutionEnvironment
from pyflink.datastream.connectors.file_system import FileSink
from pyflink.datastream.formats.orc import OrcBulkWriters
from pyflink.table import DataTypes

menv = StreamExecutionEnvironment.get_execution_environment()
menv.set_parallelism(1)
menv.add_jars("file:///home/konstantin/pyflink_demo/flink-sql-orc-1.19.2.jar") ①
OUTPUT_PATH = "file:///home/konstantin/pyflink_demo/fs_sink_orc"
mock_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-02"),
    (3, 1003, 70.00, "2025-01-03"),
]
mds = menv.from_collection(mock_data,
                           type_info=Types.ROW_NAMED(
                               ["txn_id", "acc_id", "txn_value", "txn_date"],
                               [Types.INT(), Types.INT(), Types.DOUBLE(), Types.STRING()]))

row_type = DataTypes.ROW([
    DataTypes.FIELD('txn_id', DataTypes.INT()),
    DataTypes.FIELD('acc_id', DataTypes.INT()),
    DataTypes.FIELD('txn_value', DataTypes.DOUBLE()),
    DataTypes.FIELD('txn_date', DataTypes.STRING()),
])

file_sink_orc = FileSink.for_bulk_format(
    OUTPUT_PATH,
    OrcBulkWriters.for_row_type( ②
        row_type=row_type,
        writer_properties=Configuration(),
        hadoop_config=Configuration(),
    )
).build()

mds.sink_to(file_sink_orc)
menv.execute()
```

① Adds a dependency JAR to work with ORC.

② Uses an ORC writer with `FileSink`.

Write to a Kafka topic

The following snippet forwards the content of a `DataStream` to a Kafka topic using the Kafka sink object. The connection to the server is provided by the Kafka connector.

```
menv = StreamExecutionEnvironment.get_execution_environment()
menv.set_parallelism(1)
menv.add_jars("file:///D:/dev/py/pyflink/flink-sql-connector-kafka-3.3.0-1.19.jar") ①
mock_data = [
    (1, 1001, 100.00, "2025-01-01"),
    (2, 1002, 20.00, "2025-01-02"),
    (3, 1003, 70.00, "2025-01-03"),
]
mds = menv.from_collection(mock_data, type_info=Types.ROW_NAMED(
        ["txn_id", "acc_id", "txn_value", "txn_date"],
        [Types.INT(), Types.INT(), Types.DOUBLE(), Types.STRING()]))

kafka_sink = KafkaSink.builder() \ ②
    .set_bootstrap_servers("<host>:<port>") \
    .set_record_serializer(
        KafkaRecordSerializationSchema.builder()
            .set_topic("m_topic")
            .set_value_serialization_schema(SimpleStringSchema())
            .build()
    ) \
    .set_delivery_guarantee(DeliveryGuarantee.AT_LEAST_ONCE) \
    .build()

mds.sink_to(kafka_sink) ③
menv.execute()
```

① Adds the Kafka connector JAR to the PyFlink job context. To get an up-to-date Kafka connector version, see Flink documentation.

② Creates a Kafka sink object that connects to a Kafka broker using the `"<host>:<port>"` values.

③ Writes `DataStream` items to the Kafka sink.

Dependency management

PyFlink jobs may require third-party dependencies, for example, Python libraries, connector JARs, ML frameworks, etc. The ways for importing these are slightly different for Table API and DataStream API and are shown below.

JAR dependencies

Table API

```
table_env.get_config().set("pipeline.jars", "file:///path/to/myJar1.jar;file:///path/to/myJar2.jar")
table_env.get_config().set("pipeline.classpaths", "file:///path/to/myJar1.jar;file:///path/to/myJar2.jar")
```

DataStream API

```
stream_ex_env.add_jars("file:///path/to/myJar1.jar;file:///path/to/myJar2.jar")
stream_ex_env.add_classpaths("file:///path/to/myJar1.jar", "file:///path/to/myJar2.jar")
```

As a CLI argument

```
$ flink run \
      --python test_flink_job.py \
      --jarfile flink-sql-connector-kafka-3.3.0-1.19.jar
```

Python dependencies

Table API

```
table_env.add_python_file("path/to/myScript.py")
```

DataStream API

```
stream_execution_environment.add_python_file("path/to/myScript.py")
```

As a CLI argument

```
$ flink run \
      --python test_flink_job.py \
      --pyFiles test.py
```

Flink SQL Gateway

Konstantin Alpashkin

Contents

To Table of Contents

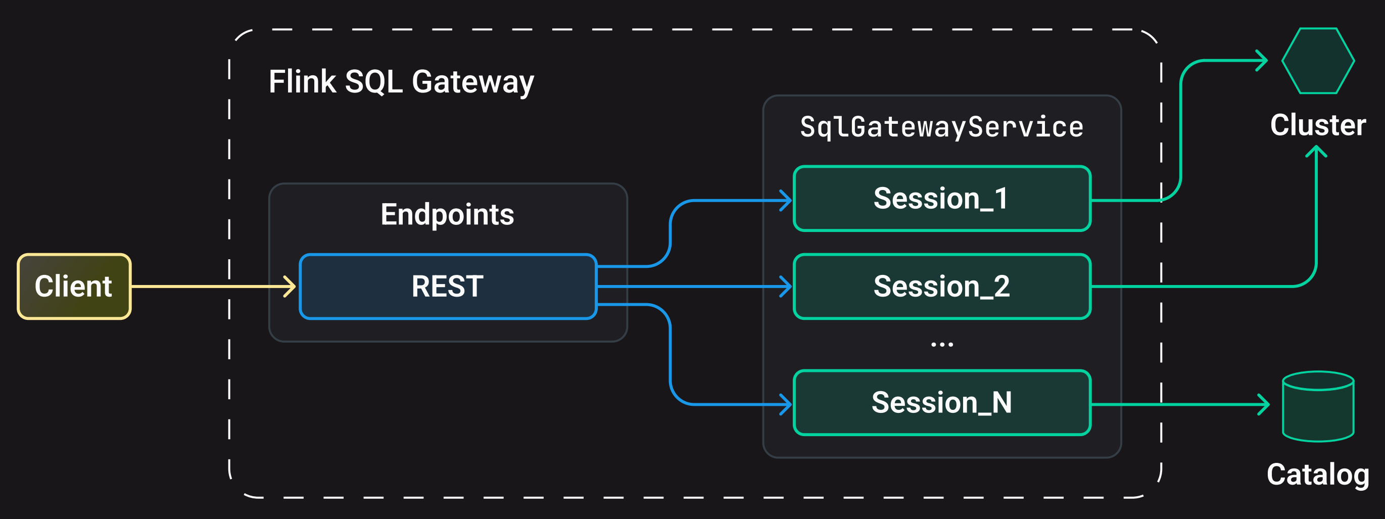
Work with REST endpoint

Usage example

Flink SQL Gateway is a service that allows multiple clients to run Flink SQL queries in parallel. With SQL Gateway, users can submit Flink jobs, look up the metadata, and analyze the data in real time.

SQL Gateway consists of the SqlGatewayService processor that handles SQL queries and pluggable endpoints through which users can submit the queries. As of ADH 3.1.2.b1, the supported endpoint is REST endpoint which assumes interaction with the service via HTTP.

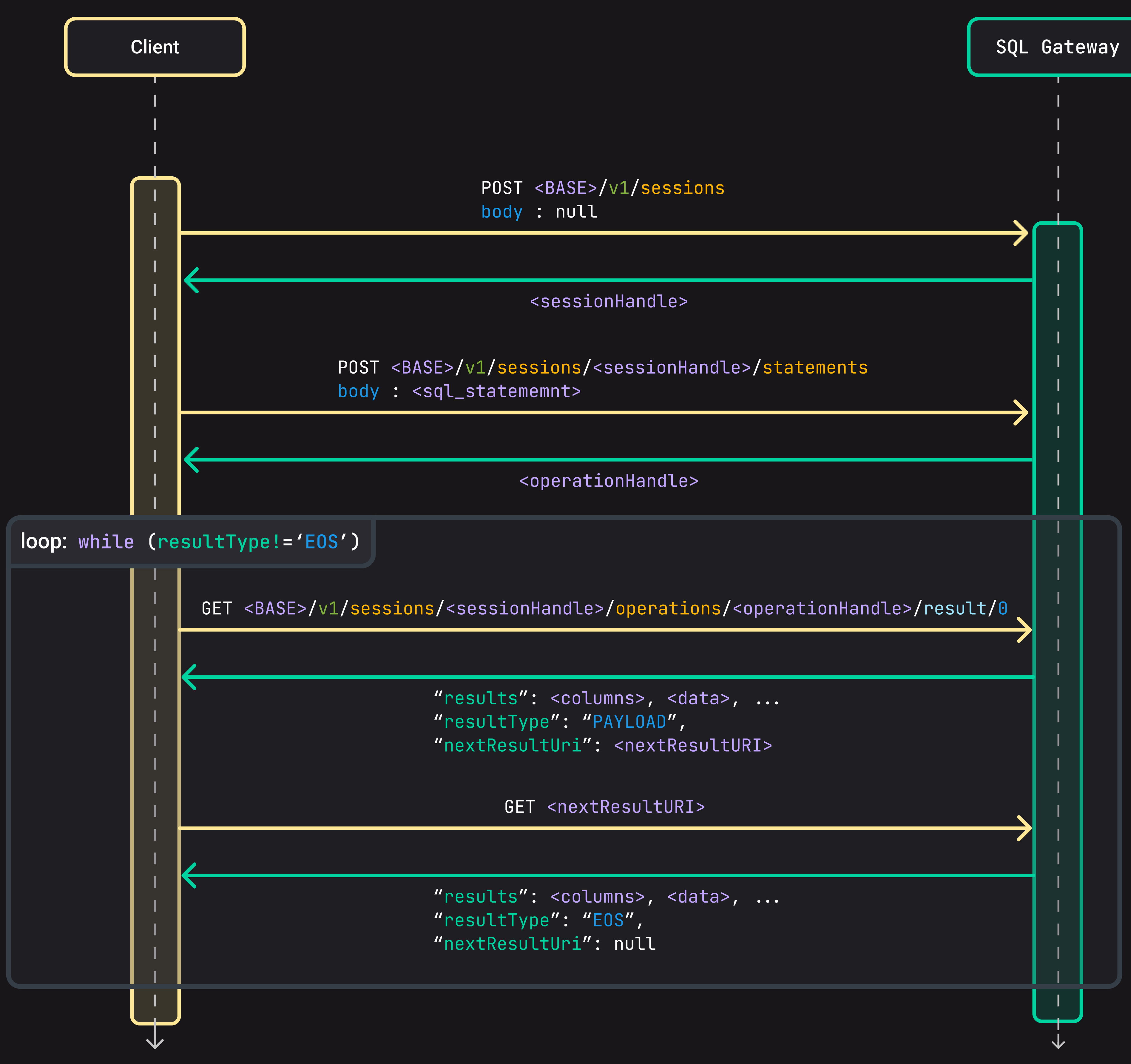
At a high level, the SQL Gateway architecture is illustrated below.



Flink SQL Gateway architecture

Work with REST endpoint

The interaction flow through the REST endpoint is presented in the following diagram.



Client
SQL Gateway
POST <BASE>/v1/sessions
body : null
<sessionHandle>
POST <BASE>/v1/sessions/<sessionHandle>/statements
body : <sql_statememnt>
<operationHandle>
loop: while (resultType!='EOS')
GET <BASE>/v1/sessions/<sessionHandle>/operations/<operationHandle>/result/0
"results": <columns>, <data>, ...
"resultType": "PAYLOAD",
"nextResultUri": <nextResultURI>
GET <nextResultURI>
"results": <columns>, <data>, ...
"resultType": "EOS",
"nextResultUri": null

Interaction via REST endpoint

The key interaction steps are as follows.

1. **Create session**. First, a client requests to create a new session. The SQL Gateway creates a new session object and returns the session identifier — `sessionHandle` that is used for further communication. The session lives for a configurable time period and its lifetime can be manually prolonged.
2. **Submit SQL**. When the session is created, the client submits SQL queries to the SQL Gateway server. For each query received, SQL Gateway creates a new Operation entity and returns an `operationHandle` ID that is used later to fetch query results. The Operation has its lifecycle and can be forcibly cancelled/closed to release the resources.
3. **Fetch results**. Using the `operationHandle`, the client fetches results from the previously created Operation. If the Operation's computations are ready, SQL Gateway returns a batch of result data and a URI pointing to the next batch. When all results are served, SQL Gateway returns a response with `"resultType": "EOS"`, indicating that all the results have been fetched.

The REST URL to access the SQL Gateway service is available on the ***Clusters*** → ***<YOUR_CLUSTER>*** → ***Services*** → ***Flink*** → ***Info*** page in ADCM, for example: *http://ka-adh-1.ru-central1.internal:8083/v1/info*. The complete reference on the SQL Gateway REST API is available in Flink documentation.

Usage example

The example below walks you through basic SQL Gateway operations using the REST endpoint. To run the operations using curl, follow the steps below. Alternatively, you can download and import the Postman collection to run the requests.

1. Check the SQL Gateway service availability.

```
$ curl http://<sql-gateway-host>:8083/v1/info
```

   This returns basic information about the service.

```
{"productName":"Apache Flink","version":"1.16.2"}
```

2. Create a new session.

```
$ curl -X POST http://<sql-gateway-host>:8083/v1/sessions
```

   The server returns a `sessionHandle` that uniquely identifies the created user session.

```
{"sessionHandle":"97fa59cf-6aa9-440b-b5cb-191f4167f0fb"}
```

3. Using the received `sessionHandle`, submit the SQL query.

```
$ curl -X POST http://<sql-gateway-host>:8083/v1/sessions/{sessionHandle}/statements/ -d @test_query.json
```

   In this example, *test_query.json* stores a sample word count SQL query and looks as follows:

```
{
    "statement": "SELECT word, SUM(frequency) AS `count` FROM (   VALUES ('Hello', 1), ('Ciao', 1), ('Hello', 2) ) AS
WordTable(word, frequency) GROUP BY word"
}
```

   The response contains an `operationHandle` that uniquely identifies the submitted query.

```
{"operationHandle":"b18e38b0-8a9b-4f36-9121-7d9f2eb46b81"}
```

4. Fetch the query results.

```
$ curl http://<sql-gateway-host>:8083/v1/sessions/{sessionHandle}/operations/{operationHandle}/result/0
```

   The response looks as follows.

```
{"results":{"columns":[{"name":"word","logicalType":{"type":"VARCHAR","nullable":false,"length":5},"comment":null},
{"name":"count","logicalType":{"type":"INTEGER","nullable":false},"comment":null}],"data":[{"kind":"INSERT","fields":
["Hello",1]},{"kind":"INSERT","fields":["Ciao",1]},{"kind":"UPDATE_BEFORE","fields":["Hello",1]},{"kind":"UPDATE_AFTER","fields":
["Hello",3]}]},"resultType":"PAYLOAD","nextResultUri":"/v1/sessions/b9ea29d4-e373-463f-bd9a-a943d853b093/operations/c5df0878-
e102-4e02-a37d-3293aa53c038/result/1"}
```

   Notice `"resultType":"PAYLOAD"` and `"nextResultUri":"<nextURI>"` — the latter field carries a URI to get the next batch with results. If calling `<nextURI>` returns a response that contains `"resultType": "EOS"`, this indicates that all the results have been served for the given query.

5. To prolong the session lifetime, submit a heartbeat so the session doesn't get destroyed after the timeout.

```
$ curl -X POST http://<sql-gateway-host>:8083/v1/sessions/{sessionHandle}/heartbeat
```

   The `200 OK` response indicates that the session lifetime has been extended.

Apart from the requests above there are other useful actions like get status/cancel/stop operation, etc. For more details on these, see Flink documentation.

Run Flink on YARN

Konstantin Alpashkin

Contents

To Table of Contents

Overview

Prepare a user for the YARN mode

Flink on YARN modes

- Application mode
- Session mode

This article describes the use of the Flink service on top of a YARN cluster.

Overview

Generally, the Flink service supports two deployment modes:

- **Standalone mode**. This mode assumes that the TaskManager and JobManager components are installed on the ADH cluster hosts. In this mode, a Flink cluster is started and stopped using ADCM and is continuously running, waiting for new jobs to be submitted. The amount of resources allocated to JobManager/TaskManager components is fixed and can be configured using the Flink service settings.
- **YARN mode**. In this mode, the Flink service setup does not require JobManager/TaskManager components to be installed on ADH hosts. Instead, to execute a Flink application, these components are created dynamically in YARN containers and get disposed of once the job is complete (the containers' lifecycle actually varies based on the YARN mode used). When a Flink application runs on YARN, it is transferred to the YARN ResourceManager first, which spawns YARN containers on ADH hosts managed by YARN NodeManagers. During the job execution phase, Flink dynamically allocates and de-allocates TaskManager resources to complete the job. More details on running Flink on YARN are described below.

An important difference between the Standalone and YARN mode is where the Flink application code is compiled to a JobGraph. In the Standalone mode, the Flink client does the compilation and sends the JobGraph to the JobManager for execution. This makes Flink clients heavy CPU/RAM consumers and requires additional bandwidth for moving JobGraphs over the network. In the YARN mode, the application's `main()` method runs in a dedicated YARN container with a JobManager, reducing the load on the hosts with Flink clients.

Prepare a user for the YARN mode

Before using Flink in the YARN mode, complete the steps below to prepare the local user for interaction with a YARN cluster.

1. Create a user directory in HDFS and set ownership. These directories are used by YARN to store temporary execution data.

   ```
   $ hadoop fs -mkdir /user/<username>
   $ hadoop fs -chown -R <username>:hadoop /user/<username>
   ```

   where `<username>` is the local user that runs commands like `flink run …`.

2. If Kerberos is enabled and the local user name differs from the principal name, specify the user-principal mapping using an `auth_to_local` rule. For example:

   ```
   RULE:[1:$1@$0](k_alpashkin_krb2@AD.RANGER-TEST)s/.*/admin/
   ```

3. Run the ***Update Core configuration*** action of the Core configuration service.
4. Restart HDFS, YARN, and Flink services.
5. If Ranger plugins are enabled for HDFS and YARN, grant permissions to the user.

Flink on YARN modes

A Flink cluster can run on YARN in two modes:

- Application mode
- Session mode

These modes are described below with usage examples.

Application mode

Running Flink on YARN in the application mode creates a short-lived, per-application Flink cluster that serves only one Flink application. This cluster exists only for the lifetime of the submitted application and gets automatically destroyed when the application exits. This mode provides better isolation for multiple Flink jobs, since every job runs in its own dedicated Flink cluster without any interference.

Here's a sequence of events that take place when you submit a Flink application to YARN in the application mode:

1. The processing starts with submitting a Flink JAR to the Flink client.
2. The Flink client requests YARN to create a dedicated container (Application Master) to deploy a JobManager. The application code with all the dependencies gets transferred to the JobManager's container. The `main()` method runs inside this container.
3. The JobManager requests additional containers from YARN to deploy TaskManagers, required for executing Flink tasks. The number of spawned TaskManagers depends on the application's startup parameters and the Flink service settings (parallelism, number of slots).
4. When the Flink job completes (successfully or not, or gets cancelled), all the containers allocated by YARN are shut down.

Below is an example of running Flink on YARN in the application mode:

1. Submit the application JAR to the Flink client:

   ```
   $ flink run-application \ ①
   -t yarn-application \ ②
   /usr/lib/flink/examples/batch/WordCount.jar
   ```

   ① The `run-application` command instructs Flink to run in the application mode.

   ② Specifies YARN as the deployment target.

   The output indicates the interaction with YARN components and contains the ID of the YARN Application Master:

   ```
   Setting HBASE_CONF_DIR=/etc/hbase/conf because no HBASE_CONF_DIR was set.
   SLF4J: Class path contains multiple SLF4J bindings.
   SLF4J: Found binding in [jar:file:/usr/lib/flink/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
   SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
   SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
   SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
   2025-06-24 15:37:27,169 INFO  org.apache.hadoop.yarn.client.DefaultNoHARMFailoverProxyProvider [] - Connecting to
   ResourceManager at adh-ka-2.ru-central1.internal/10.92.41.75:8032
   2025-06-24 15:37:27,411 INFO  org.apache.hadoop.yarn.client.AHSProxy                         [] - Connecting to Application
   History server at adh-ka-3.ru-central1.internal/10.92.43.237:10200
   2025-06-24 15:37:27,429 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - No path for the flink jar
   passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
   2025-06-24 15:37:27,668 INFO  org.apache.hadoop.conf.Configuration                           [] - resource-types.xml not found
   2025-06-24 15:37:27,669 INFO  org.apache.hadoop.yarn.util.resource.ResourceUtils            [] - Unable to find 'resource-types.
   xml'.
   2025-06-24 15:37:27,758 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - Cluster specification:
   ClusterSpecification{masterMemoryMB=2048, taskManagerMemoryMB=2048, slotsPerTaskManager=1}
   2025-06-24 15:37:28,517 WARN  org.apache.hadoop.hdfs.shortcircuit.DomainSocketFactory      [] - The short-circuit local reads
   feature cannot be used because libhadoop cannot be loaded.
   2025-06-24 15:37:29,279 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - Cannot use kerberos delegation
   token manager, no valid kerberos credentials provided.
   2025-06-24 15:37:29,304 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - Submitting application master
   application_1750778541795_0003
   2025-06-24 15:37:29,363 INFO  org.apache.hadoop.yarn.client.api.impl.YarnClientImpl        [] - Submitted application
   application_1750778541795_0003
   2025-06-24 15:37:29,363 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - Waiting for the cluster to be
   allocated
   2025-06-24 15:37:29,367 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - Deploying cluster, current
   state ACCEPTED
   2025-06-24 15:37:34,956 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - YARN application has been
   deployed successfully.
   2025-06-24 15:37:34,957 INFO  org.apache.flink.yarn.YarnClusterDescriptor                    [] - Found Web Interface adh-ka-2.
   ru-central1.internal:8081 of application 'application_1750778541795_0003'.
   ```

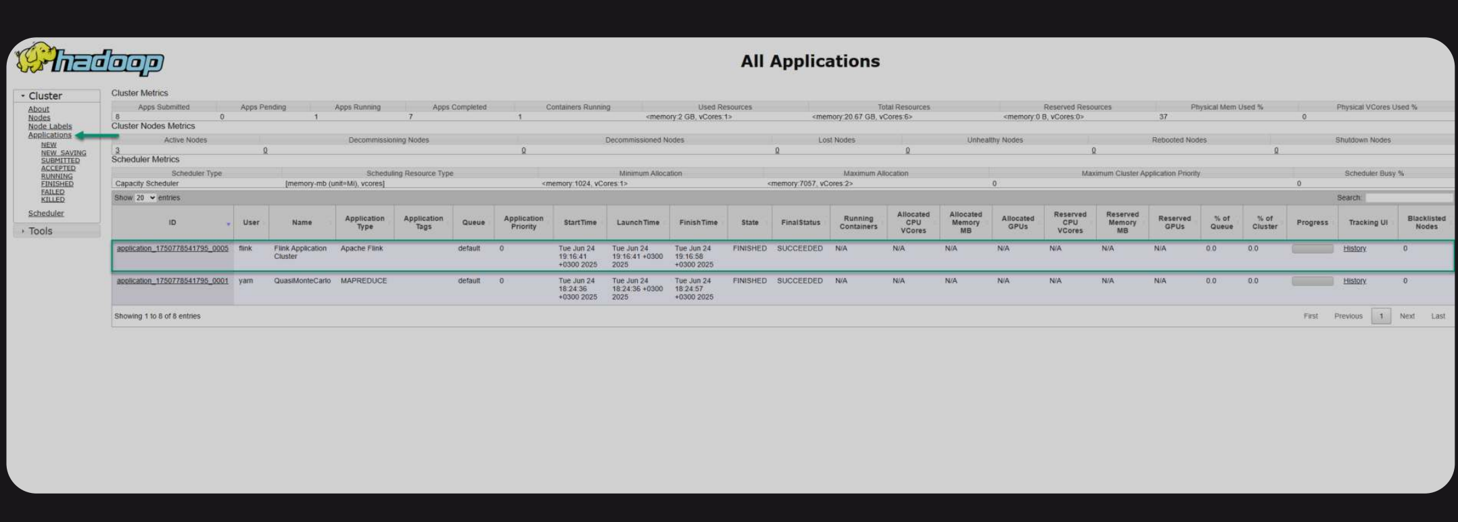
2. Once the application is submitted, you can invoke commands on it like cancelling, listing, or creating a savepoint. For example:

   ```
   $ flink list \ ①
     -t yarn-application \
     -Dyarn.application.id=application_1750778541795_0003

   $ flink cancel \ ②
     -t yarn-application \
     -Dyarn.application.id=application_1750778541795_0003 <jobId>
   ```

   ① Lists Flink jobs belonging to the `application_1750778541795_0003` YARN application.

   ② Cancels the Flink job identified by `<jobId>`. Cancelling a job stops the Flink cluster.

3. Verify that the application has succeeded in the YARN ResourceManager web UI. The up-to-date web UI link can be found in ADCM (***Clusters → <ADH_cluster> → Services → YARN → Info***).



YARN web interface

TIP

For more light-weight job submission, consider using the `yarn.provided.lib.dirs` parameter and upload your Flink JARs to an HDFS location accessible by all ADH hosts. In this case, the Flink client does not have to transfer heavy JARs to the JobManager as it can be picked from a remote location. For example:

```
$ flink run-application \
    -t yarn-application \
    -Dyarn.provided.lib.dirs="hdfs://user/admin/flink_demo/test_flink.jar" \
    hdfs://user/admin/flink_demo/test_flink.jar
```

For more information on running Flink on YARN in the application mode, see Flink documentation.

Session mode

In this mode, a single (typically long-running) Flink cluster is deployed in YARN containers to serve multiple Flink jobs. When a Flink job completes, the Flink cluster session remains alive.

Running Flink in this mode assumes the following sequence of events:

1. The processing starts with creating a Flink cluster session in YARN. This creates a YARN container (Application Master) with a JobManager. Containers for TaskManagers are initialized dynamically based on the startup parameters and the configuration of the Flink service.
2. The ID of the launched YARN application is retrieved (for example, from the YARN web UI, logs, or by using the `yarn application -list` command).
3. One or more Flink jobs are attached to the running Flink cluster session.
4. When required, the Flink cluster session is terminated manually.

Below is an example of running a Flink application on YARN in the session mode:

1. Launch a Flink cluster session on YARN:

   ```
   $ flink-yarn-session \
       -d \ ①
       -nm my_flink_job \ ②
       -s 2 \ ③
       -tm 2048 ④
       -yj /usr/lib/flink/lib/flink-dist-1.20.1.jar ⑤
   ```

   ① Starts a Flink cluster session in the detached mode.

   ② Sets a custom name for the YARN application.

   ③ Sets the number of slots per TaskManager.

   ④ Sets the amount of memory in megabytes to allocate per TaskManager container.

   ⑤ Overrides the `yarn.flink-dist-jar` configuration property. Without this, a warning may appear in logs: `No path for the flink jar passed…`.

2. Get the YARN application ID. For this, you can use the YARN web UI or use the command:

   ```
   $ yarn application -list
   ```

   The output indicates the running Flink cluster:

   ```
   WARNING: YARN_OPTS has been replaced by HADOOP_OPTS. Using value of YARN_OPTS.
   2025-06-26 14:24:07,737 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at adh-ka-2.ru-central1.
   internal/10.92.41.75:8032
   2025-06-26 14:24:07,974 INFO client.AHSProxy: Connecting to Application History server at adh-ka-3.ru-central1.internal/10.92.
   43.237:10200
   Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTED, RUNNING] and tags: []):1
                   Application-Id      Application-Name        Application-Type          User           Queue
   State             Final-State              Progress                        Tracking-URL
   application_1750778541795_0007          my_flink_job            Apache Flink         flink         default
   RUNNING              UNDEFINED                  100%            http://10.92.43.237:8081
   ```

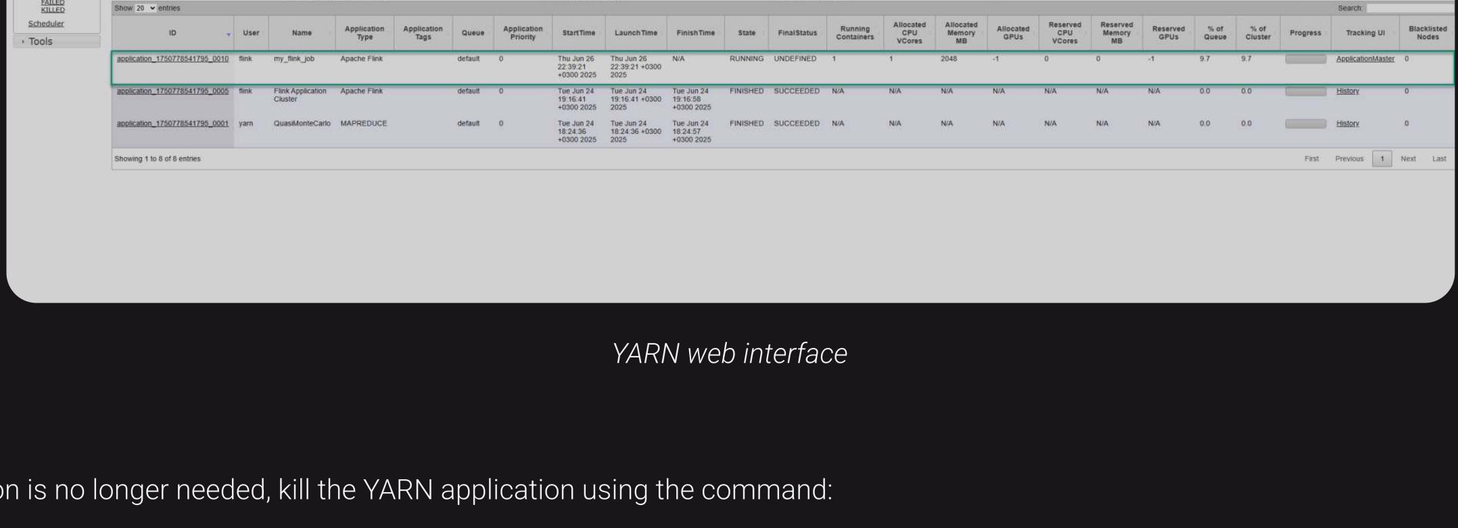
3. Attach a Flink job to the YARN session:

   ```
   $ flink run
       -t yarn-session
       -Dyarn.application.id=application_1750778541795_0007
       /usr/lib/flink/examples/batch/WordCount.jar
   ```

   Sample output

   ```
   Setting HBASE_CONF_DIR=/etc/hbase/conf because no HBASE_CONF_DIR was set.
   SLF4J: Class path contains multiple SLF4J bindings.
   SLF4J: Found binding in [jar:file:/usr/lib/flink/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
   SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
   SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
   SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
   2025-06-26 14:27:37,600 INFO  org.apache.flink.yarn.cli.FlinkYarnSessionCli                [] - Found Yarn properties file
   under /var/tmp/.yarn-properties-flink.
   2025-06-26 14:27:37,600 INFO  org.apache.flink.yarn.cli.FlinkYarnSessionCli                [] - Found Yarn properties file
   under /var/tmp/.yarn-properties-flink.
   Executing WordCount example with default input data set.
   Use --input to specify file input.
   Printing result to stdout. Use --output to specify output path.
   2025-06-26 14:27:38,422 INFO  org.apache.hadoop.yarn.client.DefaultNoHARMFailoverProxyProvider [] - Connecting to
   ResourceManager at adh-ka-2.ru-central1.internal/10.92.41.75:8032
   2025-06-26 14:27:38,659 INFO  org.apache.hadoop.yarn.client.AHSProxy                         [] - Connecting to Application
   History server at adh-ka-3.ru-central1.internal/10.92.43.237:10200
   2025-06-26 14:27:38,673 INFO  org.apache.flink.yarn.YarnClusterDescriptor                  [] - No path for the flink jar
   passed. Using the location of class org.apache.flink.yarn.YarnClusterDescriptor to locate the jar
   2025-06-26 14:27:38,815 INFO  org.apache.flink.yarn.YarnClusterDescriptor                  [] - Found Web Interface adh-ka-
   3.ru-central1.internal:8081 of application 'application_1750778541795_0007'.
   Job has been submitted with JobID 5fb3965a1ed2ee3ed5790a3fe4df84c1
   Program execution finished
   Job with JobID 5fb3965a1ed2ee3ed5790a3fe4df84c1 has finished.
   Job Runtime: 7027 ms
   Accumulator Results:
   - 180f560f3af3522f5cb4c5ba2d0f4297 (java.util.ArrayList) [170 elements]

   (a,5)
   (action,1)
   ...
   ```

4. View the application execution details in the YARN ResourceManager web UI. The up-to-date web UI link can be found in ADCM (***Clusters → <ADH_cluster> → Services → YARN → Info***).



YARN web interface

5. When the Flink cluster session is no longer needed, kill the YARN application using the command:

   ```
   $ yarn application -kill application_1750778541795_0007
   ```

   The output:

   ```
   2025-06-26 14:33:19,976 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at adh-ka-2.ru-central1.
   internal/10.92.41.75:8032
   2025-06-26 14:33:20,226 INFO client.AHSProxy: Connecting to Application History server at adh-ka-3.ru-central1.internal/10.92.
   43.237:10200
   Killing application application_1750778541795_0007
   2025-06-26 14:33:20,609 INFO impl.YarnClientImpl: Killed application application_1750778541795_0007
   ```

For more information on running Flink on YARN in the session mode, see Flink documentation.

Logging in Flink

Konstantin Alpashkin

Contents

To Table of Contents

Logs location and format
Log types
Configure logging with ADCM
  Default log4j.properties
View logs in Flink UI
Logging in Flink apps

The Flink service generates text logs that can be useful for analyzing the cause of different errors that may occur while working with the service. By default, Flink uses the SLF4J logging facade and Log4j 2 as a logging framework. You can also use any SLF4J-compliant logging framework, such as Log4j 1 or Logback.

Logs location and format

By default, Flink stores logs in the */var/log/flink* directory on ADH hosts where Flink components are installed. An example of the */var/log/flink/* directory contents is below.

```
$ ls -l /var/log/flink
```

```
-rw-rw-r-- 1 admin admin 37329 Mar 13 11:34 flink-admin-client-ka-adh-6.ru-central1.internal.log
-rw-r--r-- 1 flink flink 77921 Mar 13 11:34 flink-flink-standalonesession-0-ka-adh-6.ru-central1.internal.log
-rw-r--r-- 1 flink flink   457 Mar 13 11:07 flink-flink-standalonesession-0-ka-adh-6.ru-central1.internal.out
-rw-r--r-- 1 flink flink 70599 Mar 13 11:34 flink-flink-taskexecutor-0-ka-adh-6.ru-central1.internal.log
-rw-r--r-- 1 flink flink  3084 Mar 13 11:34 flink-flink-taskexecutor-0-ka-adh-6.ru-central1.internal.out
-rw-r--r-- 1 flink flink    73 Mar 13 11:07 flink-jobmanager.out
-rw-r--r-- 1 root  root  38489 Mar 13 11:34 flink-root-client-ka-adh-6.ru-central1.internal.log
-rw-r--r-- 1 flink flink    68 Mar 13 11:07 flink-taskmanager.out
```

Log types

There are several types of logs generated by Flink components. Their description is listed below.

Log name	Attributes	Comment
flink-<user_name>-client-<host_name>.log	▪ `<host_name>` — Flink host name; ▪ `<user_name>` — the Unix user that submitted a Flink job.	Flink client logs
flink-flink-standalonesession-<n>-<host_name>.log	▪ `<host_name>` — Flink host name; ▪ `<n>` — session index.	YARN logs
flink-flink-taskexecutor-<n>-<host_name>.log	▪ `<host_name>` — Flink host name; ▪ `<n>` — job index.	TaskExecutor job details
flink-jobmanager.out	—	Flink JobManager's output
flink-taskmanager.out	—	Flink TaskManager's output

Configure logging with ADCM

With the use of Log4j, configuring logging in Flink assumes editing Log4j configuration files. The Flink service uses two Log4j configuration files:

- *log4j.properties*. The default configuration file used by JobManager/TaskManager components.
- *log4j-cli.properties*. Used for logging events related to the Flink command line interface.

These files are located in the */etc/flink/conf* directory on the ADH hosts with Flink components installed. However, instead of editing the files manually on each host, ADCM provides a way to update the Log4j configuration on all ADH hosts at once. For this:

1. In ADCM, go to ***Clusters*** → ***<cluster_name>*** → ***Services*** → ***Flink*** → ***Primary Configuration*** and enable the ***Show advanced*** option.
2. Click ***log4j.properties***/***log4j-cli.properties*** and edit the configuration. The Log4j configuration specified via ADCM UI will overwrite the contents of the */etc/flink/conf/log4j*.properties* files during the service restart. For detailed reference on supported Log4j properties, see Log4j 2 documentation.

NOTE

Updates to Log4j configuration files do not require a restart of the Flink service. The configuration is scanned periodically (30 seconds by default) and can be refreshed on the fly.

Below are ADCM configuration properties of the Flink service related to logging.

Flink service logging parameters

Settings section	Configuration property	Default value	Description
flink-env.sh	FLINK_LOG_DIR	/var/log/flink	The location of Flink log files
flink-conf.yaml	Logging level	INFO	The root log level

Default log4j.properties

Below is the default *log4j.properties* file used by Flink. The major properties are highlighted to help you get started with Log4j configuration.

```
################################################################################
{% include 'apache_license.txt' %}
################################################################################

# Allows this configuration to be modified at runtime. The file will be checked every 30 seconds.
monitorInterval=30 ①

# This affects logging for both user code and Flink
rootLogger.level = {{ services.flink.config.flink_conf.logging_level }} ②
rootLogger.appenderRef.file.ref = MainAppender

# Uncomment this if you want to _only_ change Flink's logging
#logger.flink.name = org.apache.flink
#logger.flink.level = INFO

# The following lines keep the log level of common libraries/connectors on
# log level INFO. The root logger does not override this. You have to manually
# change the log levels here.
logger.akka.name = akka
logger.akka.level = INFO
logger.hadoop.name = org.apache.hadoop
logger.hadoop.level = INFO
logger.zookeeper.name = org.apache.zookeeper
logger.zookeeper.level = INFO
logger.shaded_zookeeper.name = org.apache.flink.shaded.zookeeper3
logger.shaded_zookeeper.level = INFO
{% raw %}
# Log all infos in the given file
appender.main.name = MainAppender
appender.main.type = RollingFile
appender.main.append = true
appender.main.fileName = ${sys:log.file} ③
appender.main.filePattern = ${sys:log.file}.%i ④
appender.main.layout.type = PatternLayout
appender.main.layout.pattern = %d{yyyy-MM-dd HH:mm:ss,SSS} %-5p %-60c %x - %m%n ⑤
appender.main.policies.type = Policies
appender.main.policies.size.type = SizeBasedTriggeringPolicy ⑥
appender.main.policies.size.size = 100MB
appender.main.policies.startup.type = OnStartupTriggeringPolicy ⑦
appender.main.strategy.type = DefaultRolloverStrategy
appender.main.strategy.max = ${env:MAX_LOG_FILE_NUMBER:-10} ⑧

# Suppress the irrelevant (wrong) warnings from the Netty channel handler
logger.netty.name = org.jboss.netty.channel.DefaultChannelPipeline
logger.netty.level = OFF
{% endraw %}
```

① An interval in seconds for scanning this Log4j configuration for updates.

② Sets the root log level. The possible severity levels are: `ALL`, `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF`. The default value is taken from the ***flink-conf.yaml*** → ***Logging level*** property in ADCM.

③ Defines a name for log files. The default pattern generates file names like *flink-flink-taskexecutor-<n>-<host_name>.log*.

④ Defines a file name pattern for rotated files.

⑤ Specifies a pattern to convert log events to strings.

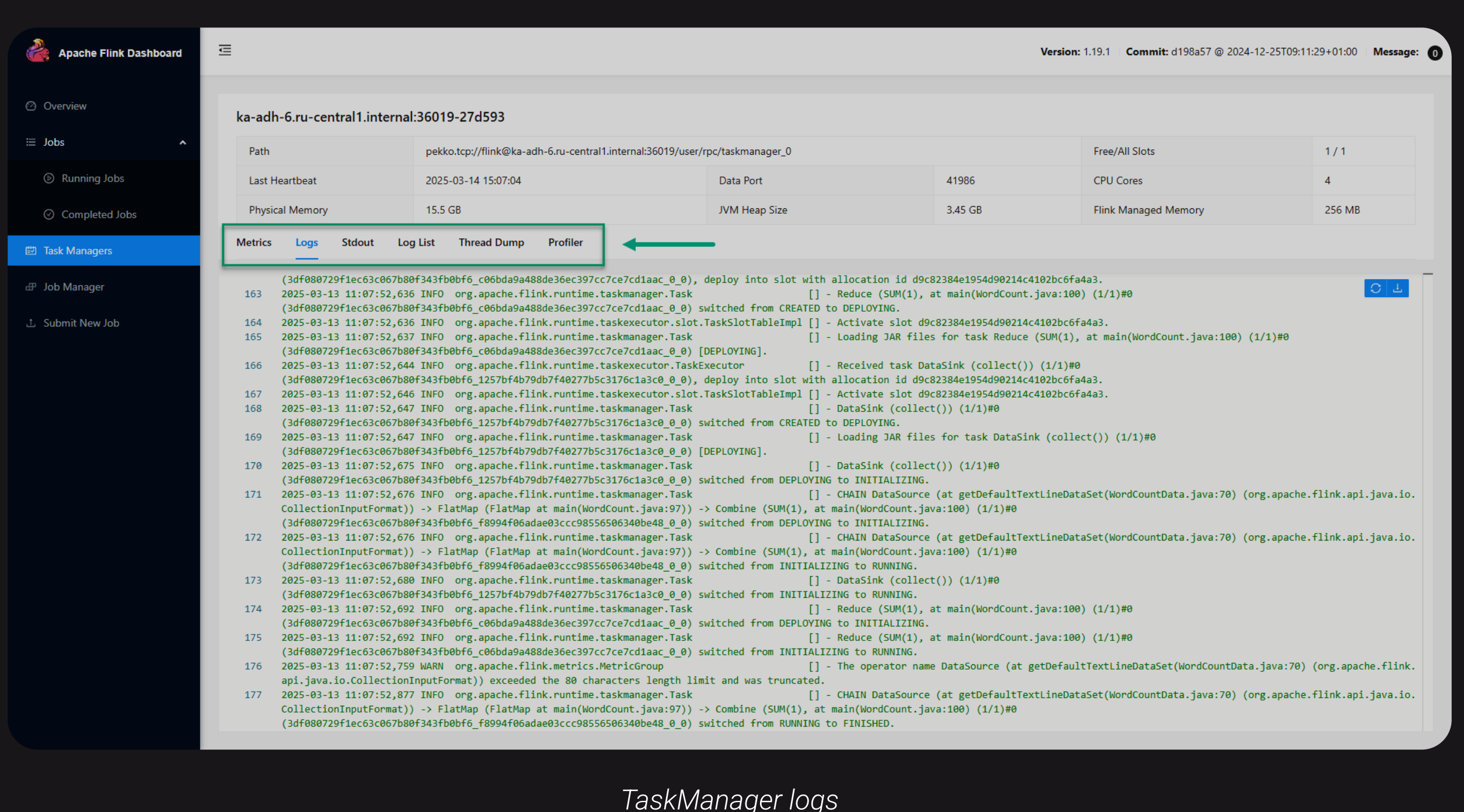
⑥ A rollover policy for rotating log files upon reaching a certain size. When a log file reaches the size limit set in the `appender.main.policies.size.size` parameter, the appender starts writing to a new log file.

⑦ Another policy that causes log files rollover on Flink startup (if a log file is older than the current JVM's startup time).

⑧ Sets the maximum number of log files. When exceeded, the oldest log files are deleted.

View logs in Flink UI

You can also view Flink log files generated by TaskManagers/JobManagers using Flink UI. To view TaskManager logs, in Flink UI go to the ***Task Managers*** page and select the Task Manager you need. Text logs, system output, and other metadata are available on tabs as shown in the image.



Version: 1.19.1 **Commit:** d198a57 @ 2024-12-25T09:11:29+01:00 **Message:** 0

ka-adh-6.ru-central1.internal:36019-27d593

Path	pekko.tcp://flink@ka-adh-6.ru-central1.internal:36019/user/rpc/taskmanager_0			Free/All Slots	1 / 1
Last Heartbeat	2025-03-14 15:07:04	Data Port	41986	CPU Cores	4
Physical Memory	15.5 GB	JVM Heap Size	3.45 GB	Flink Managed Memory	256 MB

```
(3df080729f1ec63c067b80f343fb0bf6_c06bda9a488de36ec397cc7ce7cd1aac_0_0), deploy into slot with allocation id d9c82384e1954d90214c4102bc6fa4a3.
2025-03-13 11:07:52,636 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - Reduce (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_c06bda9a488de36ec397cc7ce7cd1aac_0_0) switched from CREATED to DEPLOYING.
2025-03-13 11:07:52,636 INFO  org.apache.flink.runtime.taskexecutor.slot.TaskSlotTableImpl [] - Activate slot d9c82384e1954d90214c4102bc6fa4a3.
2025-03-13 11:07:52,637 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - Loading JAR files for task Reduce (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_c06bda9a488de36ec397cc7ce7cd1aac_0_0) [DEPLOYING].
2025-03-13 11:07:52,644 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutor           [] - Received task DataSink (collect()) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_1257bf4b79db7f40277b5c3176c1a3c0_0_0), deploy into slot with allocation id d9c82384e1954d90214c4102bc6fa4a3.
2025-03-13 11:07:52,646 INFO  org.apache.flink.runtime.taskexecutor.slot.TaskSlotTableImpl [] - Activate slot d9c82384e1954d90214c4102bc6fa4a3.
2025-03-13 11:07:52,647 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - DataSink (collect()) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_1257bf4b79db7f40277b5c3176c1a3c0_0_0) switched from CREATED to DEPLOYING.
2025-03-13 11:07:52,647 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - Loading JAR files for task DataSink (collect()) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_1257bf4b79db7f40277b5c3176c1a3c0_0_0) [DEPLOYING].
2025-03-13 11:07:52,675 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - DataSink (collect()) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_1257bf4b79db7f40277b5c3176c1a3c0_0_0) switched from DEPLOYING to INITIALIZING.
2025-03-13 11:07:52,676 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.
CollectionInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:97)) -> Combine (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_f8994f06adae03ccc98556506340be48_0_0) switched from DEPLOYING to INITIALIZING.
2025-03-13 11:07:52,676 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.
CollectionInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:97)) -> Combine (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_f8994f06adae03ccc98556506340be48_0_0) switched from INITIALIZING to RUNNING.
2025-03-13 11:07:52,680 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - DataSink (collect()) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_1257bf4b79db7f40277b5c3176c1a3c0_0_0) switched from INITIALIZING to RUNNING.
2025-03-13 11:07:52,692 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - Reduce (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_c06bda9a488de36ec397cc7ce7cd1aac_0_0) switched from DEPLOYING to INITIALIZING.
2025-03-13 11:07:52,692 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - Reduce (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_c06bda9a488de36ec397cc7ce7cd1aac_0_0) switched from INITIALIZING to RUNNING.
2025-03-13 11:07:52,759 WARN  org.apache.flink.metrics.MetricGroup                         [] - The operator name DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.
api.java.io.CollectionInputFormat)) exceeded the 80 characters length limit and was truncated.
2025-03-13 11:07:52,877 INFO  org.apache.flink.runtime.taskmanager.Task                    [] - CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.
CollectionInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:97)) -> Combine (SUM(1), at main(WordCount.java:100) (1/1)#0
(3df080729f1ec63c067b80f343fb0bf6_f8994f06adae03ccc98556506340be48_0_0) switched from RUNNING to FINISHED.
```

TaskManager logs

To view Job Manager logs, go to the ***Job Manager*** page and switch to the ***Logs*** tab.

Logging in Flink apps

The following snippet shows how to use an SLF4J logger in your Java applications for Flink.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class MyFlinkApp {
        private static final Logger logger = LoggerFactory.getLogger(MyFlinkApp.class); ①
        public static void main(String[] args) {
        logger.info("Log INFO event"); ②
        try {
            logger.debug("In-try with placeholder: {}", "foobar"); ③
        } catch (Exception e) {
            logger.error("An {} has occurred.", "error", e);
        }
        }
}
```

① Creates a logger object. It is recommended to keep the logger object in a `private static final` field.

② Logs an arbitrary event.

③ Logs an event using placeholders.

For Python apps, log events can be created as follows:

```
import logging

logging.basicConfig(stream=sys.stdout,
                    level=logging.INFO,
                    format="%(message)s")
logging.info("Log this INFO event")
```

Build a streaming ETL pipeline with Flink

Konstantin Alpashkin

Contents

To Table of Contents

Overview

Prepare Arenadata products

- ADPG
- ADS and ADS Control
- ADH

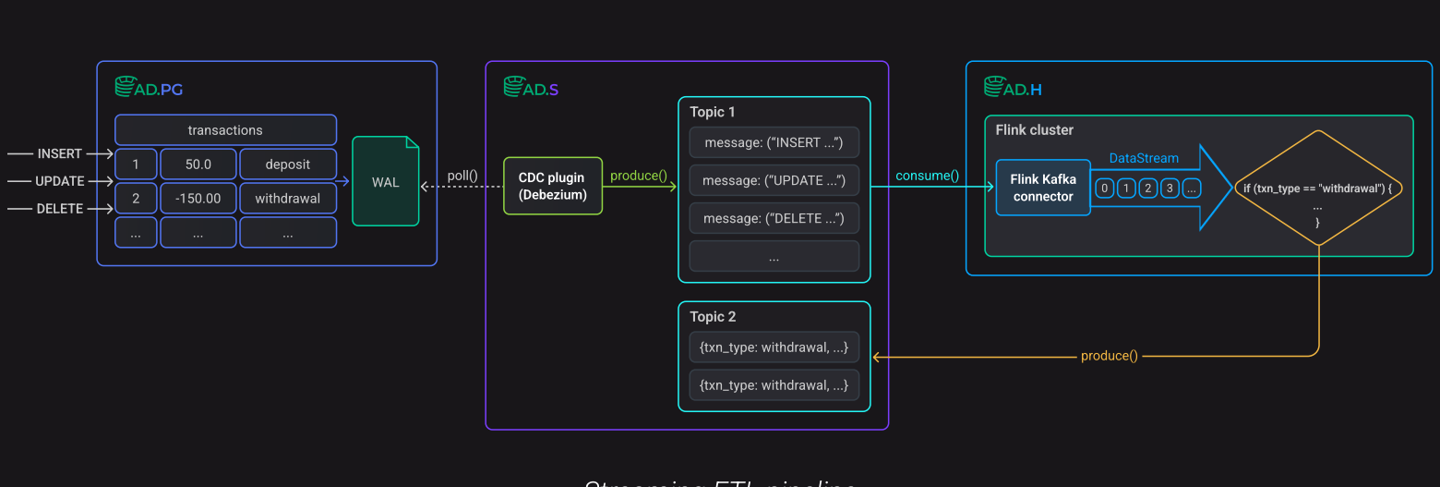
Create a Flink application

Run the pipeline

Overview

This section shows how to build a streaming ETL pipeline using the Flink service (ADH) and other Arenadata products. At the heart of the demo pipeline is a Flink application that analyzes a data stream (Kafka) in real time, extracts specific messages, and forwards them to another data stream based on a condition.

The pipeline flow is visualized in the following diagram.



Streaming ETL pipeline

The major steps are highlighted below:

1. A PostgreSQL (ADPG) server stores the `transactions` table like the one below:

```
 txn_id | acc_id |   txn_value  |      txn_date       |  txn_type
--------+--------+-----------+---------------------+------------
      1 |   1001 |       130.00 | 2025-07-01 10:15:00 | spb
      2 |   1002 |       -50.00 | 2025-07-02 09:45:00 | withdrawal
      3 |   1002 |       -50.25 | 2025-07-01 11:00:00 | withdrawal
     ...|    ... |          ... |                 ... |       ...
```

   The table continuously receives `INSERT`/`UPDATE`/`DELETE` commands from external clients.

2. An ADS cluster is hosting a Kafka server. The ADS cluster also has a CDC plugin (Debezium) configured to monitor the changes made to the `transactions` table. The plugin polls the ADPG's write-ahead log (WAL) for new entries and submits each change to the Kafka queue as a separate message.
3. In the ADH cluster, a Flink application subscribes to the Kafka topic using the Flink connector for Kafka. Flink analyzes each message from the topic, and if a specific condition is met (`txn_type=withdrawal`), sends the message to a special Kafka queue.

> **NOTE**
>
> In the example pipeline described in this article, Flink performs a basic check by looking for a specific substring in the message body. In real-life pipelines, Flink can be used for much deeper analysis of the data streams and is capable of revealing non-obvious patterns using stateful processing.

The pipeline demonstrated in this article is built using the following Arenadata software stack.

Product	Version
Arenadata Postgres (ADPG)	16.3.4
Arenadata Streaming (ADS)	3.7.2.1
Arenadata Streaming Control (ADS Control)	2.5.0
Arenadata Hyperwave (ADH)	4.0.0

Prepare Arenadata products

The sections below show how to install and configure the Arenadata products involved in the pipeline flow from scratch:

- ADPG
- ADS and ADS Control
- ADH

ADPG

1. Install an ADPG cluster according to the Online installation guide.
2. Ensure the ADPG service uses the default port number (`5432`) for incoming connections.
3. Create the `demo_user` role. For this, run `psql` on the host with the ADPG service:

```
$ sudo su - postgres
$ psql
```

   And execute the command:

```
CREATE ROLE demo_user WITH LOGIN SUPERUSER PASSWORD 'password';
```

4. In the installed ADPG cluster, create the `demo_db` database. For this, run in `psql`:

```
CREATE DATABASE demo_db OWNER demo_user;
```

5. In the `demo_db` database, create the `transactions` table in the `public` schema:

```
\c demo_db
```

```
CREATE TABLE transactions (
    txn_id     BIGINT,
    acc_id     INTEGER,
    txn_value  DOUBLE PRECISION,
    txn_date   TIMESTAMP,
    txn_type   TEXT
);
```

6. Modify the pg_hba.conf configuration to allow the access to ADPG. For this, add the following line to the ***PG_HBA*** field on the configuration page of the ADPG service:

```
host demo_db demo_user all trust
```

> **NOTE**
>
> This line allows TCP/IP connections to ADPG from any IP address, which is insecure for a production environment. For better security, specify explicit IP addresses of hosts that will connect to the ADPG server.

7. Set the `wal_level=logical` configuration parameter. This adds information required to support logical decoding to the log, which is required by Debezium. To set the parameter, add an entry in the ***postgresql.conf custom section*** field on the configuration parameters page of the ADPG service.

Debezium permissions

To allow Debezium to interact with ADPG, ensure that the corresponding PostgreSQL role used by Debezium has the following privileges:

- `REPLICATION`
- `LOGIN`
- `CREATE`
- `SELECT`

In this article, the role with the `SUPERUSER` permissions includes all the required privileges, but it is recommended that you use this role only for testing. For better security, it is recommended to grant only necessary privileges.

More information on setting up user privileges for Debezium can be found in the Debezium documentation.

ADS and ADS Control

1. Install an ADS cluster according to the Online installation guide. The cluster must include the ZooKeeper, Kafka, and Kafka Connect services.
2. Enable the ***auto.create.topics.enable*** Kafka configuration parameter in the ***server.properties*** group. This is required to allow Debezium to create Kafka topics automatically.
3. Install an ADS Control cluster according to the instructions.
4. Integrate ADS Control with the ADS cluster according to the instructions.
5. In the ADS Control web UI, create a Debezium connector for the ADPG server according to the instructions. The Debezium connector used in this scenario has the following JSON configuration:

```
{
  "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
  "publication.autocreate.mode": "filtered",
  "database.user": "demo_user", ①
  "database.dbname": "demo_db", ②
  "tasks.max": 1,
  "database.port": 5432,
  "plugin.name": "pgoutput",
  "topic.prefix": "postgres",
  "database.hostname": "10.92.43.174", ③
  "database.password": "password", ④
  "name": "PostgresConnector",
  "value.converter": "org.apache.kafka.connect.storage.StringConverter",
  "key.converter": "org.apache.kafka.connect.storage.StringConverter"
}
```

   ① The PostgreSQL role for Debezium to connect to ADPG.

   ② The ADPG database to monitor for changes.

   ③ The address of the ADPG server. Replace with your ADPG server IP address.

   ④ The password to access the ADPG server.

As a result, Debezium reads the ADPG WAL file and produces messages to the `postgres.public.transactions` Kafka topic. An example message is below:

```
Struct{after=Struct{txn_id=1,acc_id=1001,txn_value=250.0,txn_date=1751364900000000,txn_type=deposit},source=Struct{version=2.7.0.
Final,connector=postgresql,name=postgres,ts_ms=1752485024943,snapshot=first,db=demo_db,sequence=
[null,"29241720"],ts_us=1752485024943896,ts_ns=1752485024943896000,schema=public,table=transactions,txId=758,lsn=29241720},op=r,t
s_ms=1752485025133,ts_us=1752485025133518,ts_ns=1752485025133518357}
```

ADH

Install an ADH cluster according to the Online installation guide. The ADH cluster must include the following services:

- Core configuration
- HDFS
- YARN
- ZooKeeper
- Flink

Create a Flink application

The following steps show how to build a Flink application in Java using Maven.

> **NOTE**
>
> For more information on running Flink applications in Python, refer to the PyFlink usage examples article.

Install Maven (if not yet installed):

1. Download Maven:

```
$ wget https://dlcdn.apache.org/maven/maven-3/3.9.10/binaries/apache-maven-3.9.10-bin.tar.gz -P /tmp
```

2. Unpack the archive:

```
$ sudo tar -zxf /tmp/apache-maven-3.9.10-bin.tar.gz -C /opt
```

3. Check the Maven version:

```
$ /opt/apache-maven-3.9.10/bin/mvn -version
```

   The output:

```
Apache Maven 3.9.10 (5f519b97e944483d878815739f519b2eade0a91d)
Maven home: /opt/apache-maven-3.9.10
Java version: 1.8.0_412, vendor: Red Hat, Inc., runtime: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.412.b08-1.el7_9.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-1160.59.1.el7.x86_64", arch: "amd64", family: "unix"
```

Once Maven is installed, create the project:

1. Create a new Maven project with the following structure:

```
flink-app-demo
│   pom.xml
└───src
    └───main
        └───java
            └───io
                └───arenadata
                       └─KafkaFlinkApp.java
```

   The example *pom.xml* is presented below.

   pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>io.arenadata</groupId>
  <artifactId>flink-app-demo</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>Flink-app-demo</name>

  <properties>
    <flink.version>1.20.1</flink.version>
    <kafka.connector.version>3.3.0-1.20</kafka.connector.version>
    <java.version>1.8</java.version>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-streaming-java</artifactId>
      <version>${flink.version}</version>
    </dependency>
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-base</artifactId>
      <version>${flink.version}</version>
    </dependency>
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kafka</artifactId>
      <version>${kafka.connector.version}</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.17.1</version>
    </dependency>
  </dependencies>
</project>
```

> **NOTE**
>
> This scenario shows how to build and run a Flink JAR in the ADH cluster, which provides core libraries, mandatory for Flink operation. Depending on the Flink version, running the project in an IDE (outside of ADH cluster) may result in `java.lang.NoClassDefFoundError` exceptions due to lack of dependencies. To test your Flink application in IDE outside the ADH cluster, be sure to include core Flink dependencies in your *pom.xml*. An example of such "fat" *pom.xml* is below.

   pom.xml for running in IDE

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>io.arenadata</groupId>
  <artifactId>flink-app-demo</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>Flink-app-demo</name>

  <properties>
    <flink.version>1.20.1</flink.version>
    <kafka.connector.version>3.3.0-1.20</kafka.connector.version>
    <java.version>1.8</java.version>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
  </properties>


    <!-- This dependency is provided, because it should not be packaged into the JAR file. -->
  <dependencies>
    <!-- Flink Streaming API -->
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-streaming-java</artifactId>
      <version>${flink.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-base</artifactId>
      <version>${flink.version}</version>
    </dependency>
    <!-- Flink Kafka Connector -->
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kafka</artifactId>
      <version>${kafka.connector.version}</version>
    </dependency>

    <!-- Kafka client (optional, provided by connector) -->
    <dependency>
      <groupId>org.apache.kafka</groupId>
      <artifactId>kafka-clients</artifactId>
      <version>3.7.2</version>
    </dependency>

    <!-- Flink Core (optional) -->
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-core</artifactId>
      <version>${flink.version}</version>
    </dependency>

    <!-- Logging (required for output visibility) -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>1.7.36</version>
    </dependency>

    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.17.1</version>
    </dependency>

    <!-- To make it work in IntelliJ IDEA: -->

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-runtime</artifactId>
      <version>${flink.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-clients</artifactId>
      <version>${flink.version}</version>
    </dependency>

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-runtime-web</artifactId>
      <version>${flink.version}</version>
      <optional>true</optional>
    </dependency>

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-test-utils</artifactId>
      <version>${flink.version}</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

2. Paste the following code to *KafkaFlinkApp.java*:

```
package io.arenadata;

import org.apache.flink.api.common.eventtime.WatermarkStrategy;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.connector.kafka.sink.KafkaSink;
import org.apache.flink.connector.kafka.sink.KafkaRecordSerializationSchema;
import org.apache.flink.connector.kafka.source.KafkaSource;
import org.apache.flink.connector.kafka.source.enumerator.initializer.OffsetsInitializer;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;

public class KafkaFlinkApp {

    public static void main(String[] args) throws Exception {
        final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

        KafkaSource<String> src_kafka_transactions = KafkaSource.<String>builder() ①
                .setBootstrapServers("ka-ads-1.ru-central1.internal:9092,ka-ads-2.ru-central1.internal:9092,ka-ads-3.ru-
central1.internal:9092") ②
                .setTopics("postgres.public.transactions") ③
                .setGroupId("flink-consumer-group")
                .setStartingOffsets(OffsetsInitializer.earliest())
                .setValueOnlyDeserializer(new SimpleStringSchema())
                .build();

        DataStream<String> stream = env.fromSource(src_kafka_transactions, ④
                WatermarkStrategy.noWatermarks(),
                "Kafka Source");


        DataStream<String> filtered = stream.filter(msg -> { ⑤
            boolean isWithdrawal = msg.contains("txn_type=withdrawal");
            if (isWithdrawal) {
                System.out.println("Withdrawal detected in message: " + msg);
            }
            return isWithdrawal;
        });

        KafkaSink<String> sink_kafka_withdrawals = KafkaSink.<String>builder() ⑥
                .setBootstrapServers("ka-ads-1.ru-central1.internal:9092,ka-ads-2.ru-central1.internal:9092,ka-ads-3.ru-
central1.internal:9092")
                .setRecordSerializer(
                        KafkaRecordSerializationSchema.<String>builder()
                                .setTopic("postgres.public.transactions.withdrawals")
                                .setValueSerializationSchema(new SimpleStringSchema())
                                .build())
                .build();

        filtered.sinkTo(sink_kafka_withdrawals); ⑦

        env.execute("My Flink demo app"); ⑧
    }
}
```

   ① Specifies a Kafka server as a source of input data.

   ② A list of Kafka broker addresses to connect to the ADS cluster. The up-to-date connection string can be found in ADCM (***Clusters → <ADS_cluster> → Services → Kafka → Info***).

   ③ A Kafka topic to subscribe to. Debezium automatically creates this topic and pushes messages with information about the changes in the ADPG table.

   ④ Creates a Flink `DataStream` object based on the Kafka source.

   ⑤ Creates another DataStream that includes only the messages containing `txn_type=withdrawal`. In real-life pipelines, Flink can perform more complicated analysis of a DataStream using advanced Flink API tools (for example, detecting X subsequent transactions committed by the same account within a specific time window).

   ⑥ Specifies a Kafka sink to send the filtered stream.

   ⑦ Directs the filtered stream to ADS.

   ⑧ Runs the application.

3. Compile the application:

```
$ cd flink-app-demo
$ /opt/apache-maven-3.9.10/bin/mvn clean package
```

Run the pipeline

1. Restart the ADPG, ADS, ADS Control, and ADH clusters.
2. On an ADH host with the Flink client component, run the Flink application:

```
$ flink run flink-app-demo/target/flink-app-demo-1.0-SNAPSHOT.jar
```

   This submits the JAR to the Flink JobManager and further execution is performed by TaskManagers. The sample output:

```
Setting HBASE_CONF_DIR=/etc/hbase/conf because no HBASE_CONF_DIR was set.
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/flink/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Job has been submitted with JobID c917cc83500f4866e3aeb67d1d17dbf7
```

3. Verify that the application has started successfully by running `flink list` or by using Flink web UI.

   Sample output:

```
[konstantin@ka-adh-2 ~]$ flink list
------------------ Running/Restarting Jobs -------------------
14.07.2025 12:36:13 : 57ff8e66a5a76cf80da5b760b44ac0ad : My Flink demo app (RUNNING)
--------------------------------------------------------------
No scheduled jobs.
```

4. Write some data to the test ADPG table. For this, run `psql` on the ADPG server host:

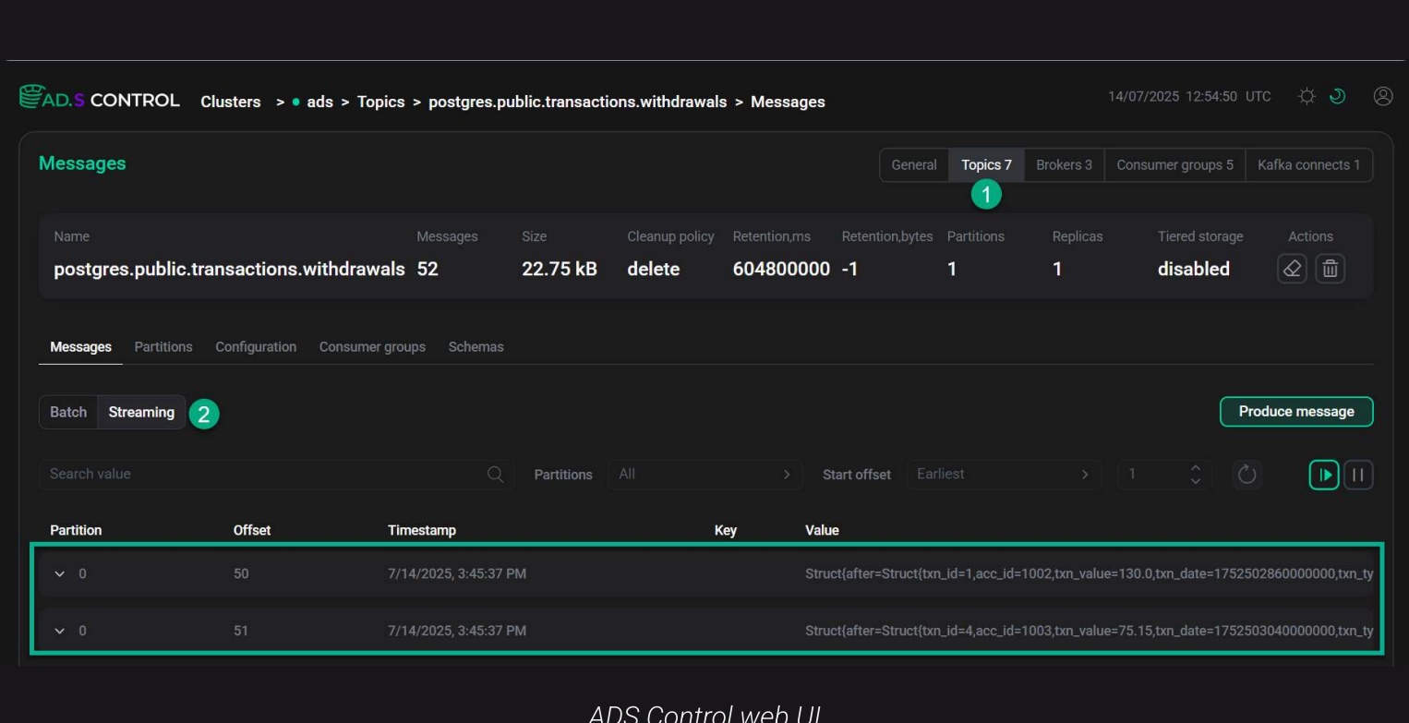
```
$ sudo su - postgres
$ psql
```

   Connect to the test database and insert some records:

```
\c demo_db
```

```
INSERT INTO public.transactions VALUES
(1, 1002,  130.00, '2025-07-14 14:21:00', 'withdrawal');
(2, 1001,  50.00, '2025-07-14 14:22:00', 'deposit');
(3, 1002,  50.25, '2025-07-14 14:23:00', 'spb');
(4, 1003,  75.15, '2025-07-14 14:24:00', 'withdrawal');
```

5. In the ADS Control web UI, go to ***Clusters → <ADS_cluster> → Topics → postgres.public.transactions.withdrawals → Messages*** as shown below.



ADS Control web UI

   After a short delay, two new messages filtered by Flink (`txn_type=withdrawal`) appear in the `postgres.public.transactions.withdrawals` topic. Verify the messages' content.

6. Stop the Flink application manually. For this, run the following commands on any ADH host with the Flink client component.

   Get the Flink application ID:

```
$ flink list
```

   Cancel the application:

```
$ flink cancel <application_id>
```

   Sample output:

```
Cancelling job c917cc83500f4866e3aeb67d1d17dbf7.
Cancelled job c917cc83500f4866e3aeb67d1d17dbf7.
```

Flink service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Flink service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Flink service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Flink components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

Flink components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Flink2 service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Flink2 service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Flink2 service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Flink2 components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

Flink2 components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Connect to HBase

Eugenia Kuzina

Contents

To Table of Contents

HBase shell

Native API

External API

Phoenix

The article describes possible ways to connect to HBase. You can use HBase shell, Phoenix, native and external HBase API. Additionally, it is possible to utilize a web interface to interact with HBase. See HBase UI overview.

HBase shell

HBase shell is a JRuby console available on each node of the HBase cluster. For more information, see Quick start with HBase shell and HBase shell commands articles.

Native API

HBase has a Java native API that provides programmatic access to Data Manipulation Language (DML). It allows you to perform CRUD operations on HBase tables. The article Use native Java API to connect HBase contains a code example. For full API member list, refer to the following link: User API Reference.

External API

You can access Apache HBase through non-Java languages and custom protocols in the same way as the native HBase version. ADH also includes HBase REST Server and HBase Thrift2 Server that can be added to a cluster as service components. For more information, see Use external API to connect HBase.

Phoenix

Apache Phoenix is a SQL query engine for Apache HBase. It is accessed as a JDBC driver, and allows you to query and manage HBase tables using SQL. For more information, see Connect to Phoenix.

Use native Java API to connect HBase

Eugenia Kuzina

To Table of Contents

Native API allows you to use Data Manipulation Language (DML). This way you can perform CRUD operations on HBase tables. This article contains an example of using several methods. For full API member list, refer to the following link: User API Reference.

You can utilize Maven to build the example below. If you do not familiar with this tool, the following tutorial should be helpful: Maven in 5 Minutes.

The example demonstrates how to create, modify, and delete an HBase table in ADH 3.1.2.1.

```java
package com.example.hbase.admin;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.HColumnDescriptor;
import org.apache.hadoop.hbase.HConstants;
import org.apache.hadoop.hbase.HTableDescriptor;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Admin;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.io.compress.Compression.Algorithm;

public class Example {

  private static final String TABLE_NAME = "MY_TABLE";
  private static final String CF_DEFAULT = "MY_COLUMN_FAMILY";


  public static void createOrOverwrite(Admin admin, HTableDescriptor table) throws IOException {
    if (admin.tableExists(table.getTableName())) {
      admin.disableTable(table.getTableName());
      admin.deleteTable(table.getTableName());
    }
    admin.createTable(table);
  }

  // Create a table
  public static void createSchemaTables(Configuration config) throws IOException {
    try (Connection connection = ConnectionFactory.createConnection(config);
         Admin admin = connection.getAdmin()) {

      HTableDescriptor table = new HTableDescriptor(TableName.valueOf(TABLE_NAME));
      table.addFamily(new HColumnDescriptor(CF_DEFAULT).setCompressionType(Algorithm.NONE));

      System.out.print("Creating table.");
      createOrOverwrite(admin, table);
      System.out.println(" Done.");
    }
  }

  public static void modifySchema (Configuration config) throws IOException {
    try (Connection connection = ConnectionFactory.createConnection(config);
         Admin admin = connection.getAdmin()) {

      TableName tableName = TableName.valueOf(TABLE_NAME);
      if (!admin.tableExists(tableName)) {
        System.out.println("Table does not exist.");
        System.exit(-1);
      }

      HTableDescriptor table = admin.getTableDescriptor(tableName);

      // Update existing table
      HColumnDescriptor newColumn = new HColumnDescriptor("NEW_COL_FAMILY");
      newColumn.setCompactionCompressionType(Algorithm.GZ);
      newColumn.setMaxVersions(HConstants.ALL_VERSIONS);
      admin.addColumn(tableName, newColumn);

      // Update existing column family
      HColumnDescriptor existingColumn = new HColumnDescriptor(CF_DEFAULT);
      existingColumn.setCompactionCompressionType(Algorithm.GZ);
      existingColumn.setMaxVersions(HConstants.ALL_VERSIONS);
      table.modifyFamily(existingColumn);
      admin.modifyTable(tableName, table);

      // Disable a table
      admin.disableTable(tableName);

      // Delete a column family
      admin.deleteColumn(tableName, CF_DEFAULT.getBytes("UTF-8"));

      // Delete a table. The table should be disabled first.
      admin.deleteTable(tableName);
    }
  }

  public static void main(String... args) throws IOException {
    Configuration config = HBaseConfiguration.create();

    //Add configuration files (hbase-site.xml, core-site.xml)
    config.addResource(new Path(System.getenv("HBASE_CONF_DIR"), "hbase-site.xml"));
    config.addResource(new Path(System.getenv("HADOOP_CONF_DIR"), "core-site.xml"));
    createSchemaTables(config);
    modifySchema(config);
  }
}
```

Add the following dependency to the *pom.xml* file:

```xml
<dependency>
    <groupId>org.apache.hbase</groupId>
    <artifactId>hbase-shaded-client</artifactId>
    <version>2.2.7</version>
</dependency>
```

Use external API to connect HBase

Eugenia Kuzina

Contents

To Table of Contents

HBase REST Server

HBase Thrift2 Server

You can access Apache HBase through non-Java languages and custom protocols. ADH 3.1.2.1 includes HBase REST Server and HBase Thrift2 Server that can be enabled as described below. It is also possible to use C/C++ Apache HBase Client, Java Data Objects (JDO), Scala, and Jython to connect to the ADH HBase service in the same way as to the native Apache HBase version.

HBase REST Server

ADH includes the HBase REST Server that can be enabled during installation. In an installed cluster, run the ***Add HBase REST Server(s)*** cluster action to add the ***HBase REST Server*** component. See HBase service management via ADCM.

By default, HBase REST Server operates on the `60080` port. To change this value, set the `hbase.rest.port` configuration parameter in ADCM UI. See Configuration parameters.

You can use the curl or wget tools to execute HTTP methods for accessing a REST server. Unless another method is specified for the current endpoint, utilize `GET` requests for select queries, `PUT` or `POST` requests for creation or modifying, and `DELETE` for deletion.

This article contains several examples of using endpoints. For information on all available endpoints, refer to Using REST Endpoints.

By default, the command output has a plain text format. To change the response format, add the `Accept` header as follows:

- for XML — `Accept: text/xml`;
- for JSON — `Accept: application/json`;
- for Protocol Buffers (data serialization protocol) — `Accept: application/x-protobuf`.

The following request returns cluster version in the XML format:

```
$ curl -vi -X GET -H "Accept: text/xml" "http://10.92.40.168:60080/version/cluster"
```

The result:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><ClusterVersion Version="2.2.7"/>
```

You can use endpoints to manage namespaces. The command below creates a namespace:

```
$ curl -vi -X POST "http://10.92.40.168:60080/namespaces/test_ns"
```

To delete the `test_ns` namespace, execute the following command:

```
$ curl -vi -X DELETE "http://10.92.40.168:60080/namespaces/test_ns"
```

The code below creates a new table or replaces an existing table's schema with the provided schema:

```
$ curl -vi -X POST \
-H "Accept: text/xml" \
-H "Content-Type: text/xml" \
-d '<?xml version="1.0" encoding="UTF-8"?><TableSchema name="users"><ColumnSchema name="cf" /></TableSchema>' \
"http://10.92.40.168:60080/users/schema"
```

Add a row to the `users` table created above:

```
$ curl -vi -X PUT \
         -H "Accept: text/xml" \
         -H "Content-Type: text/xml" \
         -d '<?xml version="1.0" encoding="UTF-8" standalone="yes"?><CellSet><Row key="cm93MQ=="><Cell
column="Y2Y6ZQo=">dmFsdWUx</Cell></Row></CellSet>' \
         "http://10.92.40.168:60080/users/testrow"
```

The row, column qualifier, and value must be Base-64 encoded.

The code below deletes the `users` table:

```
$ curl -vi -X DELETE -H "Accept: text/xml" "http://10.92.40.168:60080/users/schema"
```

HBase Thrift2 Server

You can add the ***HBase Thrift2 server*** component during cluster installation on the Add components step. It is also possible to use the ***Add Thrift2 Server(s)*** action to enable HBase Thrift2 server in the installed cluster. See HBase service management via ADCM.

Use configuration parameters listed in the table below to configure Thrift for secure authentication.

Parameter	Description
hbase.thrift.keytab.file	Thrift Kerberos keytab
hbase.thrift.kerberos.principal	Thrift Kerberos principal
hbase.thrift.security.qop	Defines authentication, integrity, and confidentiality checking. Supported values: ▪ `auth-conf` — authentication, integrity, and confidentiality checking; ▪ `auth-int` — authentication and integrity checking; ▪ `auth` — authentication checking only.

HBase Thrift2 server operates on the `9090` port.

HBase UI overview

Eugenia Kuzina

Contents

To Table of Contents

- Master Server UI
  - Home
  - Table Details
  - Procedures & Locks
  - HBCK Report
  - Process Metrics
  - Local Logs
  - Log Level
  - Debug Dump
  - Metrics Dump
  - Profiler
  - HBase Configuration
- Region Server UI
  - Home
  - Process Metrics
  - Local Logs
  - Log Level
  - Debug Dump
  - Metrics Dump
  - Profiler
  - HBase Configuration
- Thrift2 Server UI
  - Home
  - Local Logs
  - Log Level
  - Metrics Dump
  - Profiler
  - HBase Configuration

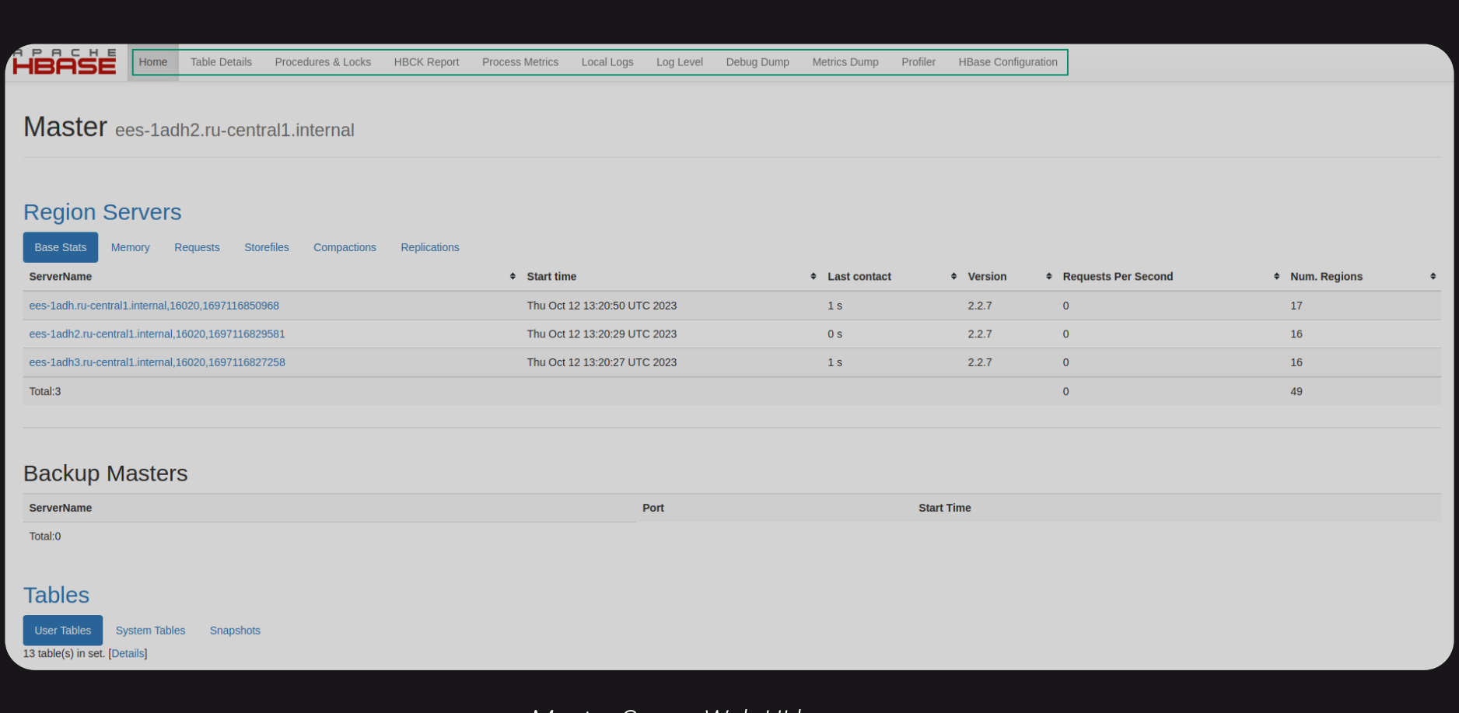
HBase web-based user interfaces allow you to view the current cluster state and change log level without service restart. HBase UI is supported for the Master, Region, and Thrift2 servers. URLs of HBase user interfaces are listed in the table below.

Web UI links

Server	Address
Master Server	http://<Master Server IP>:16010
Region Server	http://<Region Server IP>:16030
HBase Thrift2 Server	http://<HBase Thrift2 Server IP>:9095

Master Server UI

HBase Master Server UI works on the `16010` port.



Master ees-1adh2.ru-central1.internal

Region Servers

ServerName	Start time	Last contact	Version	Requests Per Second	Num. Regions
ees-1adh.ru-central1.internal,16020,1697116850968	Thu Oct 12 13:20:50 UTC 2023	1 s	2.2.7	0	17
ees-1adh2.ru-central1.internal,16020,1697116829581	Thu Oct 12 13:20:29 UTC 2023	0 s	2.2.7	0	16
ees-1adh3.ru-central1.internal,16020,1697116827258	Thu Oct 12 13:20:27 UTC 2023	1 s	2.2.7	0	16
Total:3				0	49

Backup Masters

ServerName	Port	Start Time
Total:0		

Tables

13 table(s) in set. [Details]

Master Server Web UI home page

HBase Master UI contains the following tabs:

- Home
- Table Details
- Procedures & Locks
- HBCK Report
- Process Metrics
- Local Logs
- Log Level
- Debug Dump
- Metrics Dump
- Profiler
- HBase Configuration

Home

The ***Home*** tab of the HBase Master Server UI includes the following sections with general information about an HBase cluster:

- ***Region Servers*** — contains base statistics, information about used memory, including memstore size, replication, and storefile statistics. The metric table contains links that refer to related HBase UIs of Region Servers.
- ***Backup Masters*** — provides information about Backup Master Servers if there are any in the cluster.
- ***Tables*** — lists user and system tables and snapshots.
- ***Peers*** — contains information on remote slave clusters where data is replicated to.
- ***Tasks*** — displays information on processed tasks. You can review non-RPC and RPC handler tasks separately. Also, the ***View as JSON*** link allows you to get task statistics in the JSON format.
- ***Software Attributes*** — contains general information on JVM, HBase, Hadoop, and ZooKeeper. It includes versions, checksums, dates of compilation, addresses of all registered ZK servers, the root node of this cluster in ZK, the key to adding the cluster as a peer for replication, the location of HBase home directory, and others.

Table Details

The ***Table Details*** tab contains the ***User Tables*** table with two columns:

- ***Table*** — displays links to a page with detailed table information that includes table attributes, schema, regions, region distribution by Region Servers, and table statistics.
- ***Description*** — contains table descriptions similar to the describe command output.

Procedures & Locks

The ***Procedures & Locks*** tab lists procedures that are completed with errors and locks. It also contains procedure WAL state statistics.

HBCK Report

The ***HBCK Report*** tab displays two reports: ***HBCK Chore Report*** and ***CatalogJanitor Consistency Issues***.

HBCK Chore Report checks the consistency of region information in the HMaster memory, RegionServer memory, and HDFS. It also checks whether the region exists and whether the deployment information of the region is consistent.

CatalogJanitor Consistency Issues Report determines whether the region information in the meta table is complete and whether there are overlap and hole issues.

Run the `hbck_chore_run` and catalogjanitor_run commands in hbase shell to generate new reports. Only report titles are shown if there are no problems to display.

Process Metrics

The ***Process Metrics*** tab displays a metric list for the HMaster process that includes the following items:

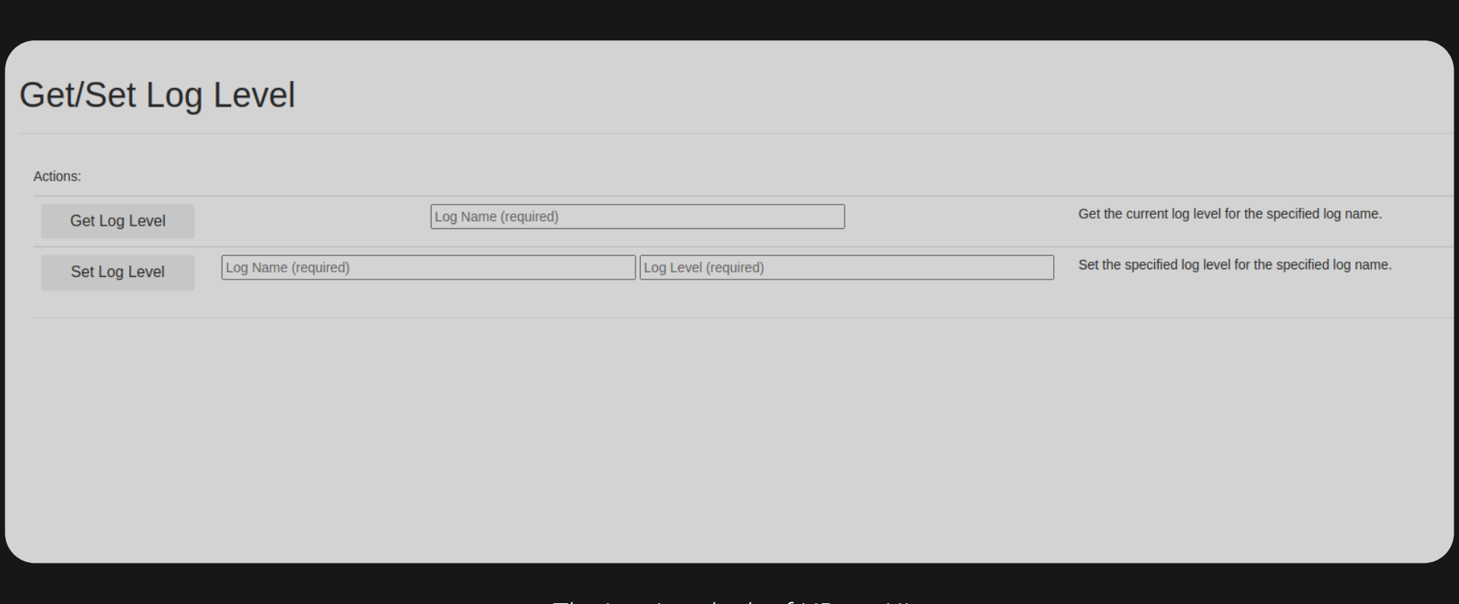
- ***Threads*** — statistics on process threads.
- ***GC Collectors*** — statistics on Java garbage collectors.
- ***Metaspace*** — statistics on `Metaspace`. The `Metaspace` region holds memory for your live class metadata and classloaders.
- ***Compressed Class Space*** — statistics on `Compressed Class Space` that is the region of compressed pointers used to store information about loaded classes.
- ***Par Eden Space*** — statistics on `Eden Space` where the most new Java objects are created. When the memory space for this region is filled up, reachable objects from `Eden Space` are moved to `Survivor Space`.
- ***Par Survivor Space*** — statistics on `Survivor Space`. When the memory space for this region is filled up, reachable objects from `Survivor Space` are moved to `CMS Old Gen`.
- ***CMS Old Gen*** — statistics on the `CMS Old Gen` memory region. This heap area is used for permanent objects.

Local Logs

The ***Local Logs*** tab contains a list of links to log files stored on the current host. When you click a link, the related log file opens in the browser window. In the file system, the log files are stored in the */var/log/hbase* directory. For more information, see Logging in HBase.

Log Level

The ***Log Level*** tab allows you to change the log level without service restart. The log level set on this tab is applied until the service is restarted. After restart, settings from the *log4j.properties* file are used.



Get the current log level for the specified log name.

Set the specified log level for the specified log name.

The Log Level tab of HBase UI

To check the log level, fill in the first ***Log Name*** field and click ***Get Log Level***. The log name should contain class hierarchy, for example, `org.apache.hadoop.hbase`.

To change the log level, type a log name in the corresponding field. Fill in the ***Log Level*** field with one of the following values: `TRACE`, `DEBUG`, or `INFO` and click ***Set Log Level***.

Debug Dump

The ***Debug Dump*** tab displays a report of the current state of the Master Server. It contains information on version, tasks, executors, the process thread dump, the server configuration in the XML format and logs. In case of problems in the HBase cluster, the debug dump can be used as a start point of debugging.

Metrics Dump

The ***Metrics Dump*** tab displays a report with metrics of the Master Server. It is the dump of the JMX bean in the JSON format.

Profiler

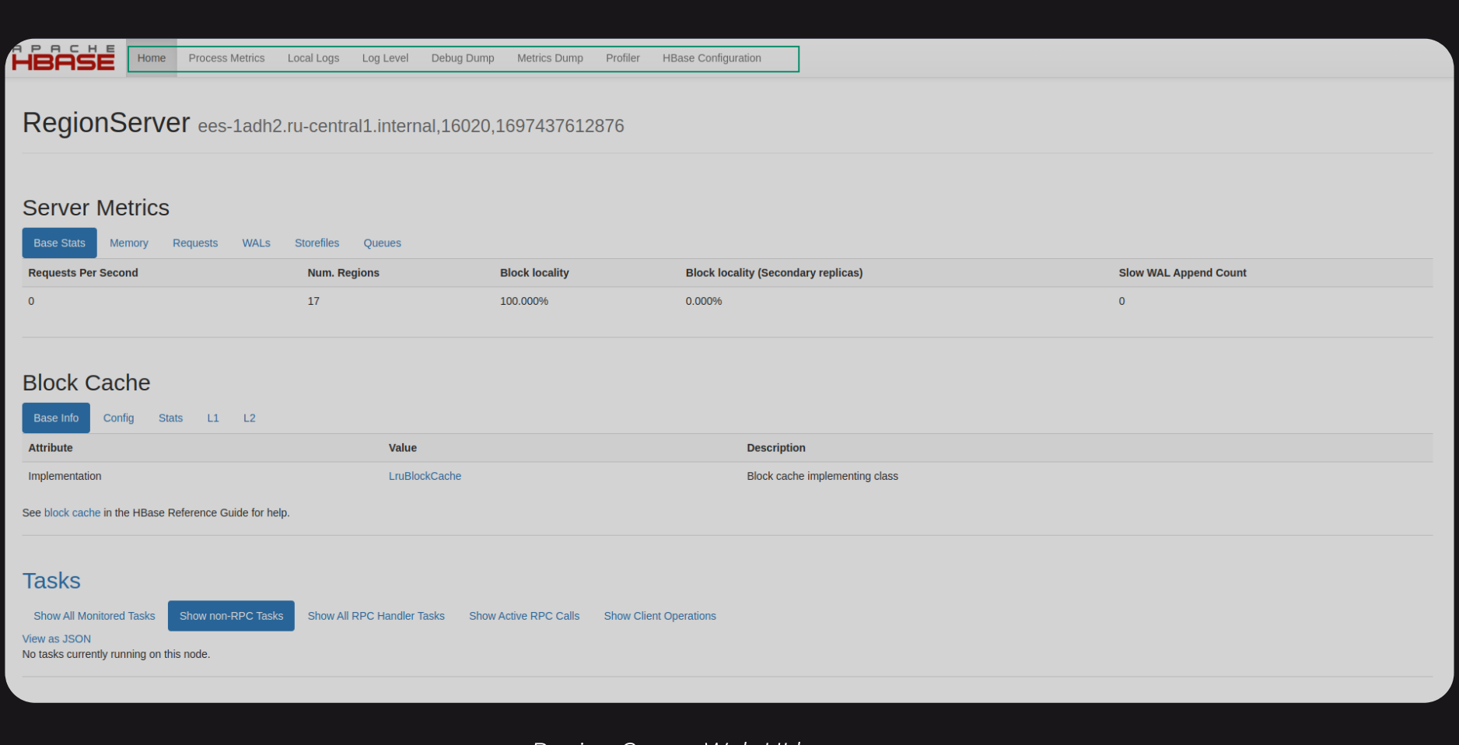
The profiler is disabled in the ADH cluster.

HBase Configuration

The ***HBase Configuration*** tab contains HBase Configuration settings in the XML format.

Region Server UI

HBase Region Server UI works on the `16030` port.



RegionServer ees-1adh2.ru-central1.internal,16020,1697437612876

Server Metrics

Requests Per Second	Num. Regions	Block locality	Block locality (Secondary replicas)	Slow WAL Append Count
0	17	100.000%	0.000%	0

Block Cache

Attribute	Value	Description
Implementation	LruBlockCache	Block cache implementing class

See block cache in the HBase Reference Guide for help.

Tasks

No tasks currently running on this node.

Region Server Web UI home page

HBase Region Server UI contains the following tabs:

- Home
- Process Metrics
- Local Logs
- Log Level
- Debug Dump
- Metrics Dump
- Profiler
- HBase Configuration

Home

The ***Home*** tab of the HBase Region Server UI includes general information about the Region Server:

- ***Server Metrics*** — contains base statistics for the current server, information about used memory including memstore size, statistics on requests, WAL, storefiles, and lengths of different types of queues.
- ***Block Cache*** — includes information on block cache statistics and configuration.
- ***Tasks*** — displays information on processed tasks. You can review non-RPC and RPC handler tasks separately. Also, the ***View as JSON*** link allows you to get task statistics in the JSON format.
- ***Regions*** — contains metrics for each region including statistics on requests, storefiles, compaction, and memstore size. A region name consists of a table name, a start key, and a randomly generated region id separated by commas. For example: `SYSTEM.LOG,\x01\x00\x00\x00,1695113080532.c4e4b11c5617462e641c9db3538a9c7d.`, `people_ages,F,1694507451861.b53a2f6692e0d4ab5a46a4def96676b0.`
- ***Replication Status*** — displays replication information and metrics if the replication is configured in the cluster.
- ***Software Attributes*** — contains general information on JVM, HBase, Hadoop, and ZooKeeper. It includes versions, checksums, dates of compilation, addresses of all registered ZK servers, address of HBase Master, and others.

Process Metrics

The ***Process Metrics*** tab displays a metric list for the HRegionServer process that includes the following items:

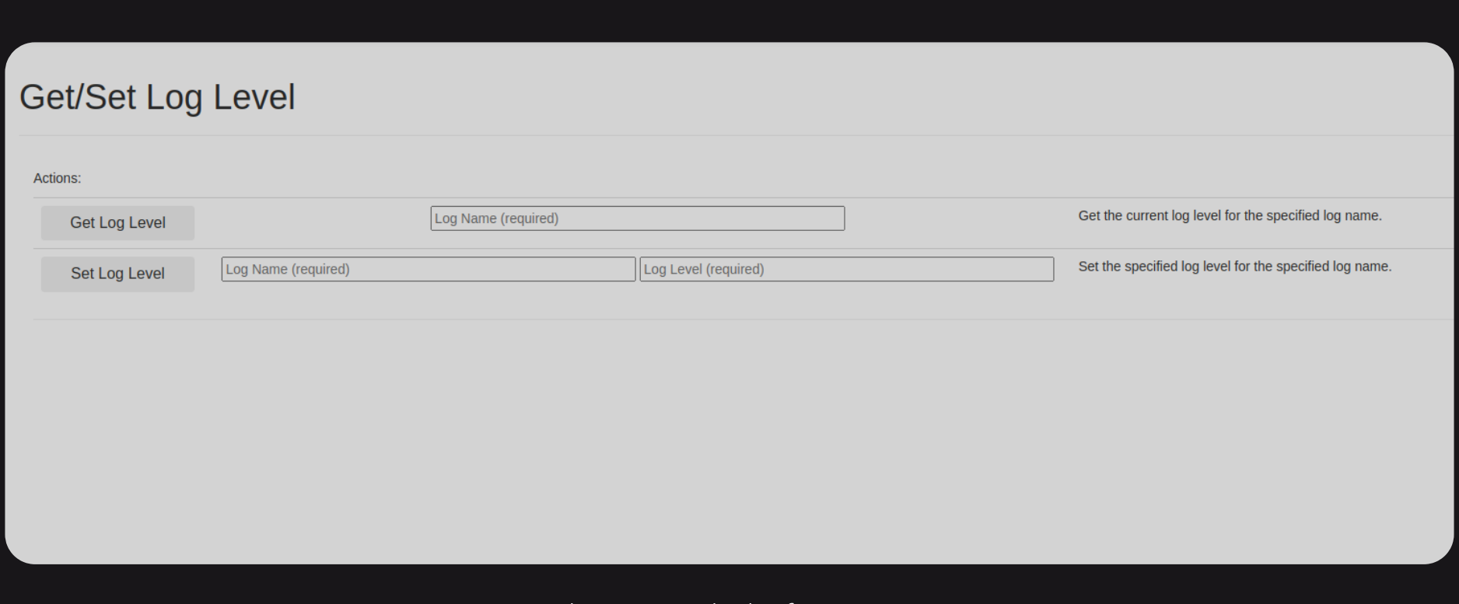
- ***Threads*** — statistics on process threads.
- ***GC Collectors*** — statistics on Java garbage collectors.
- ***Metaspace*** — statistics on `Metaspace`. The `Metaspace` region holds memory for your live class metadata and classloaders.
- ***Compressed Class Space*** — statistics on `Compressed Class Space` that is the region of compressed pointers used to store information about loaded classes.
- ***Par Eden Space*** — statistics on `Eden Space` where the most new Java objects are created. When the memory space for this region is filled up, reachable objects from `Eden Space` are moved to `Survivor Space`.
- ***Par Survivor Space*** — statistics on `Survivor Space`. When the memory space for this region is filled up, reachable objects from `Survivor Space` are moved to `CMS Old Gen`.
- ***CMS Old Gen*** — statistics on the `CMS Old Gen` memory region. This heap area is used for permanent objects.

Local Logs

The ***Local Logs*** tab contains a list of links to log files stored on the current host. When you click a link, the related log file opens in the browser window. In the file system, the log files are stored in the */var/log/hbase* directory. For more information, see Logging in HBase.

Log Level

The ***Log Level*** tab allows you to change the log level without service restart. The log level set on this tab is applied until the service is restarted. After restart, settings from the *log4j.properties* file are used.



Get the current log level for the specified log name.

Set the specified log level for the specified log name.

The Log Level tab of HBase UI

To check the log level, fill in the first ***Log Name*** field and click ***Get Log Level***. The log name should contain class hierarchy, for example, `org.apache.hadoop.hbase`.

To change the log level, type a log name in the corresponding field. Fill in the ***Log Level*** field with one of the following values: `TRACE`, `DEBUG`, or `INFO` and click ***Set Log Level***.

Debug Dump

The ***Debug Dump*** tab displays a report on the current state of the Region Server.

Metrics Dump

The ***Metrics Dump*** tab displays a report with metrics of the Region Server. It is the dump of the JMX bean in the JSON format.

Profiler

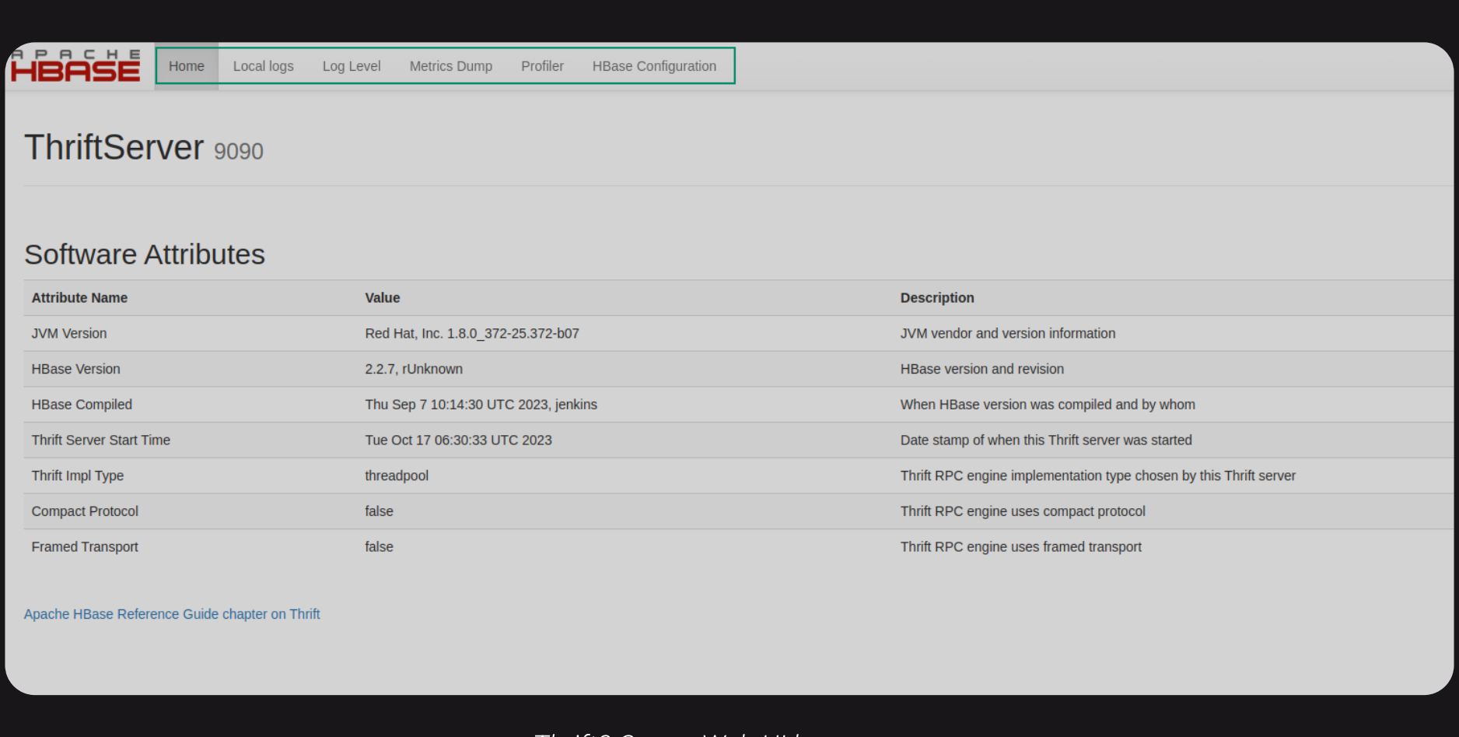
The profiler is disabled in the ADH cluster.

HBase Configuration

The ***HBase Configuration*** tab contains HBase Configuration settings in the XML format.

Thrift2 Server UI

HBase Thrift2 Server UI listens on the `9095` port.



ThriftServer 9090

Software Attributes

Attribute Name	Value	Description
JVM Version	Red Hat, Inc. 1.8.0_372-25.372-b07	JVM vendor and version information
HBase Version	2.2.7, rUnknown	HBase version and revision
HBase Compiled	Thu Sep 7 10:14:30 UTC 2023, jenkins	When HBase version was compiled and by whom
Thrift Server Start Time	Tue Oct 17 06:30:33 UTC 2023	Date stamp of when this Thrift server was started
Thrift Impl Type	threadpool	Thrift RPC engine implementation type chosen by this Thrift server
Compact Protocol	false	Thrift RPC engine uses compact protocol
Framed Transport	false	Thrift RPC engine uses framed transport

Apache HBase Reference Guide chapter on Thrift

Thrift2 Server Web UI home page

HBase Thrift2 Server UI contains the following tabs:

- Home
- Local Logs
- Log Level
- Metrics Dump
- Profiler
- HBase Configuration

Home

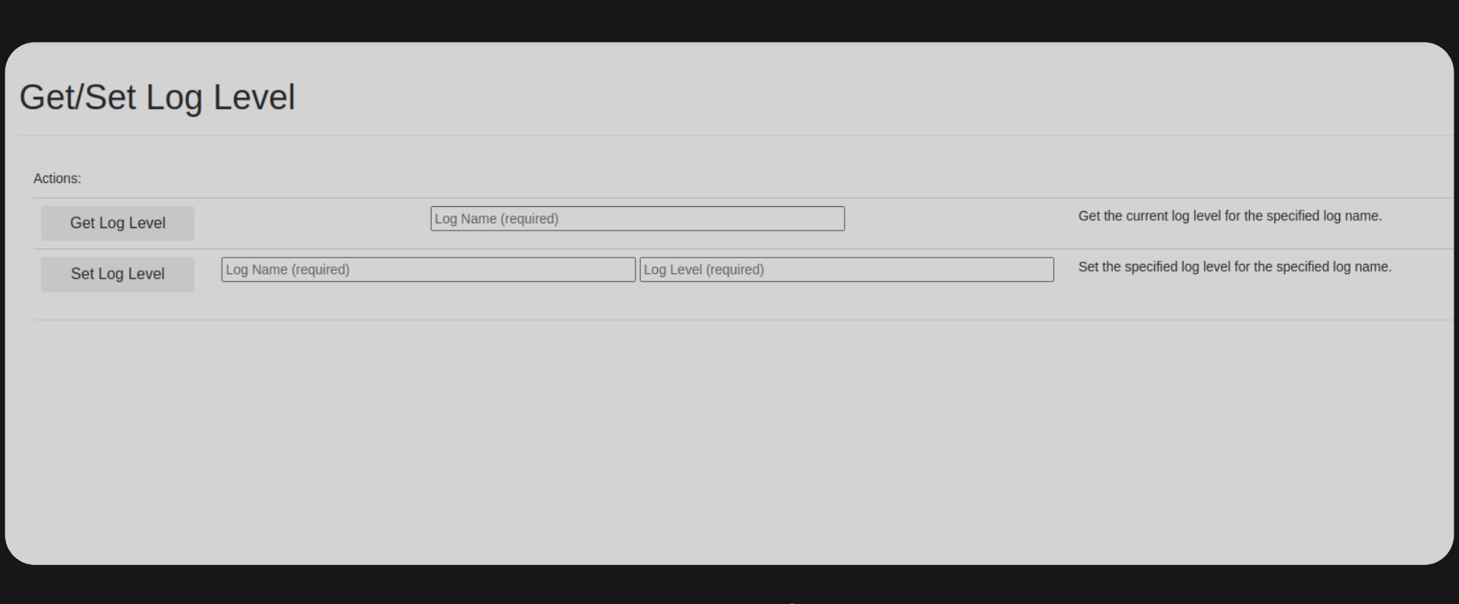
The ***Home*** tab of the Thrift2 Server UI displays the ***Software Attributes*** table with the JVM and HBase version information, Thrift2 Server start time, Thrift RPC engine implementation type chosen by the Thrift2 Server, and boolean values that show whether the Thrift RPC engine uses the compact protocol and framed transport.

Local Logs

The ***Local Logs*** tab contains a list of links to log files stored on the current host. When you click a link, the related log file opens in the browser window. In the file system, the log files are stored in the */var/log/hbase* directory. For more information, see Logging in HBase.

Log Level

The ***Log Level*** tab allows you to change the log level without service restart. The log level set on this tab is applied until the service is restarted. After restart, settings from the *log4j.properties* file are used.



Get the current log level for the specified log name.

Set the specified log level for the specified log name.

The Log Level tab of HBase UI

To check the log level, fill in the first ***Log Name*** field and click ***Get Log Level***. The log name should contain class hierarchy, for example, `org.apache.hadoop.hbase`.

To change the log level, type a log name in the corresponding field. Fill in the ***Log Level*** field with one of the following values: `TRACE`, `DEBUG`, or `INFO` and click ***Set Log Level***.

Metrics Dump

The ***Metrics Dump*** tab displays a report with metrics of the Thrift2 Server. It is the dump of the JMX bean in the JSON format.

Profiler

The profiler is disabled in the ADH cluster.

HBase Configuration

The ***HBase Configuration*** tab contains HBase Configuration settings in the XML format.

Quick start with HBase shell

Daria Barysheva

Contents

To Table of Contents

- Step 1. Create a table
- Step 2. Get information about the table
- Step 3. Put new data into the table
- Step 4. Get a specific table row
  - Read the whole row
  - Read a specific column
- Step 5. Scan the table for all data at once
  - Scan all columns
  - Scan a particular column
- Step 6. Delete cell values from the table
- Step 7. Alter the table
- Step 8. Drop the table

The simplest way to begin working with HBase is to use its utility HBase shell. It is a JRuby console available on each node of the HBase cluster immediately after its installation. To start working with the HBase shell, run the following command:

```
$ hbase shell
```

The shell prompt ends with a `>` character. All subsequent commands should be written after it.

The HBase shell prompt

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-hive.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-pig.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-thin-client.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.0.2, rUnknown, Thu Apr 15 20:20:26 UTC 2021
Took 0.0024 seconds
hbase(main):001:0>
```

To leave the HBase shell, use the following command:

```
exit
```

The main data operations available in HBase are listed below. They are based on the example described in HBase data model. The full list of these and other useful shell commands with description of their parameters you can find in HBase shell commands.

NOTE

Before executing HBase commands, we recommend you to read about the HBase data model.

Step 1. Create a table

To create a new table, write the `create` keyword, after it a table name in quotes, and then information about all column families of this table. In a column family definition, use one of the following options:

- Write only the name of the column families in quotes without curly brackets.
- Describe several attributes for the column family using key and value pairs in curly brackets.

The following command creates the `articles` table with two column families: `basic` and `tags`. Each column family is defined with two attributes: name and maximum number of stored value versions. So, all columns in these families will store five versions of each data value.

```
create 'articles', {NAME => 'basic', VERSIONS => 5}, {NAME => 'tags', VERSIONS => 5}
```

The output is similar to the following:

```
Created table articles
Took 1.6225 seconds
=> Hbase::Table - articles
```

Step 2. Get information about the table

To check whether any table exists and get information about it, use the following commands:

- **exists**. It checks whether the specified table exists:

```
exists 'articles'
```

The output is similar to the following:

```
Table articles does exist
Took 0.0705 seconds
=> true
```

If the specified table does not exist, the command returns the following message:

```
Table not_existed does not exist
Took 0.0076 seconds
=> false
```

- **list**. This command returns the table name if the specified table exists:

```
list 'articles'
```

The output is similar to the following:

```
TABLE
articles
1 row(s)
Took 0.3102 seconds
=> ["articles"]
```

If the specified table does not exist, the command returns the following message:

```
TABLE
0 row(s)
Took 0.0064 seconds
=> []
```

- **describe**. This command checks whether the table exists and is enabled and shows information about its column families:

```
describe 'articles'
```

The output is similar to the following:

```
Table articles is ENABLED
articles
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => '
false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false'
, IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE
=> 'true', BLOCKSIZE => '6553
6'}
{NAME => 'tags', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'f
alse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
 IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536
'}
2 row(s)
Took 0.1100 seconds
```

If the specified table does not exist, the command returns the following message:

```
ERROR: Table not_existed does not exist.

Describe the named table. For example:
  hbase> describe 't1'
  hbase> describe 'ns1:t1'

Alternatively, you can use the abbreviated 'desc' for the same thing.
  hbase> desc 't1'
  hbase> desc 'ns1:t1'

Took 0.0159 seconds
```

Step 3. Put new data into the table

To put new data into the table, use the `put` keyword and then write a comma-separated list of the table name, row key, column name, and data value. Each column name is a colon-separated combination of column family name and column qualifier. Column families are fixed, meaning their names are defined during table creation. Column qualifiers are not fixed, so that you can add them on the fly while putting new data into previously created tables.

The following commands put data into the `articles` table for the `article1` row key. We define columns `author` and `header` in the column family `basic`, and columns `arch`, `concepts`, and `tutorials` in the column family `tags`. The inserted data values occupy the last place in the commands. HBase generates timestamps for these data values automatically.

```
put 'articles', 'article1', 'basic:author', 'Test author'
put 'articles', 'article1', 'basic:header', 'Test article'
put 'articles', 'article1', 'tags:arch', true
put 'articles', 'article1', 'tags:concepts', true
put 'articles', 'article1', 'tags:tutorials', true
```

The following commands also put data into the `articles` table, but for another row key — `article2`. You can notice that it is not necessary to specify the previously used column qualifiers — you can enter only the necessary ones:

```
put 'articles', 'article2', 'basic:author', 'Test author2'
put 'articles', 'article2', 'basic:header', 'Test article2'
put 'articles', 'article2', 'tags:ref', true
```

The output is similar to the following:

```
Took 0.0051 seconds
```

Step 4. Get a specific table row

HBase provides various commands to read table rows as described in the following sections.

Read the whole row

To read a specific table row from a table, use the `get` keyword and then define the table name and the required row key. For example, the following command gets all values with the `article1` row key from the `articles` table:

```
get 'articles', 'article1'
```

The command returns data values of all the columns that the specified row contains. Each returned cell shows the value and timestamp of its creation:

```
COLUMN                                    CELL
 basic:author                             timestamp=1637054560096, value=Test author
 basic:header                             timestamp=1637054560118, value=Test article
 tags:arch                                timestamp=1637054560141, value=true
 tags:concepts                            timestamp=1637054560160, value=true
 tags:tutorials                           timestamp=1637054564066, value=true
1 row(s)
Took 0.0442 seconds
```

The output of the same command for the row key `article2` is as follows:

```
COLUMN                                    CELL
 basic:author                             timestamp=1637054576501, value=Test author2
 basic:header                             timestamp=1637054576516, value=Test article2
 tags:ref                                 timestamp=1637054577512, value=true
1 row(s)
Took 0.0099 seconds
```

By default, the `get` command shows only the latest versions for all data values even though the table can store up to five versions for each cell, as we have determined during table creation. If you add a new value, the `get` command will return this new value because it will have the latest timestamp.

The following example demonstrates how to add a new value for the `basic:header` column in the row with the `article1` key:

```
put 'articles', 'article1', 'basic:header', 'Test article. Version 2'
get 'articles', 'article1'
```

The output of the subsequent `get` command is similar to this:

```
COLUMN                                    CELL
 basic:author                             timestamp=1637054560096, value=Test author
 basic:header                             timestamp=1637055836875, value=Test article. Version 2
 tags:arch                                timestamp=1637054560141, value=true
 tags:concepts                            timestamp=1637054560160, value=true
 tags:tutorials                           timestamp=1637054564066, value=true
1 row(s)
Took 0.0066 seconds
```

The next example shows how to add the third value version for the `basic:header` column in the row with the `article1` key:

```
put 'articles', 'article1', 'basic:header', 'Test article. Version 3'
get 'articles', 'article1'
```

The output is similar to this:

```
COLUMN                                    CELL
 basic:author                             timestamp=1637054560096, value=Test author
 basic:header                             timestamp=1637056832082, value=Test article. Version 3
 tags:arch                                timestamp=1637054560141, value=true
 tags:concepts                            timestamp=1637054560160, value=true
 tags:tutorials                           timestamp=1637054564066, value=true
1 row(s)
Took 0.0064 seconds
```

Read a specific column

To read a specific column of the row, use the `get` command with the column name after the table name and the row key. The column name contains the column family name and the column qualifier, separated by a colon.

The following command gets the latest value, added to the `basic:header` column in the row with the `article1` key.

```
get 'articles', 'article1', 'basic:header'
```

The output is similar to this:

```
COLUMN                                    CELL
 basic:header                             timestamp=1637056832082, value=Test article. Version 3
1 row(s)
Took 0.0227 seconds
```

As you see, this command returns the latest added column value. If you want to get a particular value version, specify its timestamp along with the column name in curly brackets:

```
get 'articles', 'article1', {COLUMN => 'basic:header', TIMESTAMP => 1637054560118}
```

The output is similar to this:

```
COLUMN                                    CELL
 basic:header                             timestamp=1637054560118, value=Test article
1 row(s)
Took 0.0171 seconds
```

Step 5. Scan the table for all data at once

The `scan` command is used for getting all the table rows.

Scan all columns

To get all the columns for all table rows, use the `scan` keyword and the table name after it. The following command returns the content of the `articles` table:

```
scan 'articles'
```

The result is similar to this:

```
ROW                                       COLUMN+CELL
 article1                                 column=basic:author, timestamp=1637054560096, value=Test author
 article1                                 column=basic:header, timestamp=1637056832082, value=Test article. Version 3
 article1                                 column=tags:arch, timestamp=1637054560141, value=true
 article1                                 column=tags:concepts, timestamp=1637054560160, value=true
 article1                                 column=tags:tutorials, timestamp=1637054564066, value=true
 article2                                 column=basic:author, timestamp=1637054576501, value=Test author2
 article2                                 column=basic:header, timestamp=1637054576516, value=Test article2
 article2                                 column=tags:ref, timestamp=1637054577512, value=true
2 row(s)
Took 0.0204 seconds
```

To get a specific number of value versions, not only the latest one, specify the required number of versions using the following command:

```
scan 'articles', {VERSIONS => 5}
```

The output is similar to this:

```
ROW                                       COLUMN+CELL
 article1                                 column=basic:author, timestamp=1637054560096, value=Test author
 article1                                 column=basic:header, timestamp=1637056832082, value=Test article. Version 3
 article1                                 column=basic:header, timestamp=1637055836875, value=Test article. Version 2
 article1                                 column=basic:header, timestamp=1637054560118, value=Test article
 article1                                 column=tags:arch, timestamp=1637054560141, value=true
 article1                                 column=tags:concepts, timestamp=1637054560160, value=true
 article1                                 column=tags:tutorials, timestamp=1637054564066, value=true
 article2                                 column=basic:author, timestamp=1637054576501, value=Test author2
 article2                                 column=basic:header, timestamp=1637054576516, value=Test article2
 article2                                 column=tags:ref, timestamp=1637054577512, value=true
2 row(s)
Took 0.0138 seconds
```

Scan a particular column

To get values of a particular column for all table rows, specify the column name in curly brackets after the `scan` keyword and the table name. The following command gets all the values of the `basic:author` column in the `articles` table:

```
scan 'articles', {COLUMN => 'basic:author'}
```

The output is similar to this:

```
ROW                                       COLUMN+CELL
 article1                                 column=basic:author, timestamp=1637054560096, value=Test author
 article2                                 column=basic:author, timestamp=1637054576501, value=Test author2
2 row(s)
Took 0.0101 seconds
```

You can also define the number of required versions for a column using the following command:

```
scan 'articles', {COLUMN => 'basic:header', VERSIONS => 5}
```

The output is similar to this:

```
ROW                                       COLUMN+CELL
 article1                                 column=basic:header, timestamp=1637056832082, value=Test article. Version 3
 article1                                 column=basic:header, timestamp=1637055836875, value=Test article. Version 2
 article1                                 column=basic:header, timestamp=1637054560118, value=Test article
 article2                                 column=basic:header, timestamp=1637054576516, value=Test article2
2 row(s)
Took 0.0298 seconds
```

Step 6. Delete cell values from the table

To delete a particular cell value from the table, use the `delete` keyword and after it the table name, the row key, the column name, and, optionally, the timestamp. If timestamp is omitted, this command deletes only the latest version of the cell, leaving other versions intact. Defining a timestamp leads to deleting the cell having this timestamp.

The following command deletes the latest value of the `basic:header` column in the row with the `article1` key:

```
delete 'articles', 'article1', 'basic:header'
```

The output is similar to this:

```
Took 0.0122 seconds
```

After this, the `get` command returns the previous value version for the `basic:header` column in the `article1` row:

```
get 'articles', 'article1', {COLUMN => 'basic:header'}
```

The output looks similar to this:

```
COLUMN                                    CELL
 basic:header                             timestamp=1637055836875, value=Test article. Version 2
1 row(s)
Took 0.0239 seconds
```

Running this command with defined number of displayed versions can show that the `Test article. Version 3` value has already been deleted:

```
hbase(main):010:0> get 'articles', 'article1', {COLUMN => 'basic:header', VERSIONS => 5}
COLUMN                                    CELL
 basic:header                             timestamp=1637055836875, value=Test article. Version 2
 basic:header                             timestamp=1637054560118, value=Test article
1 row(s)
Took 0.0077 seconds
```

Step 7. Alter the table

Altering the table structure is performed in three stages:

1. Run the `disable` command to make sure that other operations can't be applied to this table.
2. Run the `alter` command to apply the necessary changes to your table.
3. Enable the table.

NOTE

In the latest HBase versions, tables can be altered without disabling them first. But, as altering enabled tables caused problems in the past, use this feature carefully and test it before moving to production.

The following commands disable the `articles` table, add the new `temp` column family into it, and enable the table again:

```
disable 'articles'
alter 'articles', {NAME => 'temp', VERSIONS => 3}
enable 'articles'
```

The output data for the `alter` command is similar to this:

```
Updating all regions with the new schema...
All regions updated.
Done.
Took 1.4006 seconds
```

Now if you apply the `describe` command to the `articles` table, you will see that its structure has been changed, that is, the `temp` column family has been added.

```
describe 'articles'
```

The output looks similar to this:

```
Table articles is ENABLED
articles
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => '
false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false'
, IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE
=> 'true', BLOCKSIZE => '6553
6'}
{NAME => 'tags', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'f
alse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
 IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536
'}
{NAME => 'temp', VERSIONS => '3', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'f
alse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
 IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536
'}
3 row(s)
Took 0.0224 seconds
```

Step 8. Drop the table

Dropping a table requires several stages:

1. Apply the `disable` command to the table to make sure that other operations can't be applied to this table.
2. Run the `drop` command to delete your table.

The following commands disable the `articles` table and drop it:

```
disable 'articles'
drop 'articles'
```

Now, if you try to apply the `exists` command to the `articles` table, you will see that it does not exist anymore:

```
hbase(main):008:0> exists 'articles'
Table articles does not exist
Took 0.0047 seconds
=> false
```

Use filters in HBase

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
- HBase filter syntax
  - Simple filter
  - Compound filter
- Individual filters in HBase
- Dynamic loading of custom filters

Overview

HBase in ADH offers a **Thrift filter language**. It allows you to filter the results of reading the data from HBase using the `get` or `scan` operations. Filtering occurs on the server side, so it doesn't reduce the load on the HBase. It does reduce, however, the load on the network as less data is transmitted to a client. Filters can be used both while using the Java API and in the HBase shell.

HBase filter syntax

Simple filter

A simple filter is specified by a string of the following kind:

```
"<FilterName> (<arg1>, <arg2>, ..., <argN>)"
```

where:

- `<FilterName>` is a name of one of the individual filters;
- `<arg1>`, `<arg2>` etc. are the arguments of that filter.

Example of a simple filter usage in HBase shell:

```
scan 'articles', { FILTER => "ColumnPaginationFilter (2, 1)"}
```

Full set of arguments is the filter condition. Different individual filters may require zero or more arguments.

Arguments may have the following types:

- string;
- integer;
- boolean;
- timestamp;
- comparison operator;
- comparator.

Timestamps

Timestamps in HBase shell table scan results are shown in human-readable format. However, when a filter requires a timestamp as an argument value, it should be specified in Unix epoch format. To convert timestamps from one format to another, use online tools like EpochConverter.

NOTE

HBase requires millisecond precision for timestamps. You need to manually add milliseconds to the timestamps after conversion to Unix epoch format.

Comparison operators and comparators

Comparison operators and comparators are used in filter arguments to compose conditions for lexicographic matches and comparisons.

The following comparison operators are used as filter arguments.

Comparison operators

Syntax	Description
<	Less
>	Greater
=	Equal
<=	Less or equal
>=	Greater or equal
!=	Not equal

The following comparators are used as filter arguments.

Comparators

Name	Description	Example
BinaryComparator	Lexicographically compares against the specified string	(>, 'binary:arc') This comparator will match everything lexicographically greater than `arc`
BinaryPrefixComparator	Lexicographically compares against a specified string. It only compares up to the length of this string	(=, 'binaryprefix:bot') This comparator will match everything that begins with `bot`
RegexStringComparator	Lexicographically compares against the specified regular expression for a string. Only `=` and `!=` comparisons can be used with this comparator	(!=, 'regexstring:be*ng') This comparator will match any string except those beginning with `be` and ending with `ng`
SubStringComparator	Searches for the given substring, case insensitive. Only `=` and `!=` comparisons can be used with this comparator	(!=, substring:con) This comparator will match any string except those containing `con`

Filter syntax rules

Filter syntax obeys the following rules:

- Filter expression must be enclosed in double quotes (`"`).
- Filter name must be a single word. All ASCII characters are suitable except parentheses, single quotes, and whitespaces.
- Arguments must be separated by commas and altogether enclosed in parentheses.
- String arguments must be each enclosed in single quotes (`'`).
- Arguments of other types (boolean, integer, comparison operator, etc.) should not be enclosed in quotes.
- Filter arguments can contain any ASCII characters. However, possible single quotes in the argument must be preceded with an extra single quote acting as an escape character.

Compound filter

A compound filter consists of individual filters united by logical operators. There are binary and unary logical operators. Binary operators unite the filters to the left and right of them:

- `AND` — the key/value is only returned if it satisfies both filters.
- `OR` — the key/value is returned if it satisfies either filter.

Unary operators precede the filter and modify its behavior:

- `SKIP` — if any of the key/value pairs does not satisfy the filter condition, the entire row is skipped.
- `WHILE` — for a particular row, the key/value pairs keep being emitted until one of them fails to satisfy the filter condition.

To build an order of individual filter processing in compound filters, use parentheses. Example:

```
(<Filter1> OR <Filter2>) AND (<Filter3> OR <Filter4>)
```

First, the result of `<Filter1>` and `<Filter2>` unification by the `OR` operator is processed. Next, the same happens to `<Filter3>` and `<Filter4>`. Finally, those processions are combined by the `AND` operator.

Logical operators and parentheses have the following order of precedence, highest to lowest:

1. parentheses;
2. `SKIP` and `WHILE` (share the same precedence);
3. `AND`;
4. `OR`.

Example of a compound filter usage in HBase shell:

```
scan 'articles', { FILTER => "ColumnPaginationFilter (2, 1) AND PrefixFilter ('co')"}
```

Individual filters in HBase

The individual filters presented in HBase are listed below along with their syntax. Example queries are executed against the test table loaded with values from test file people.csv. This file is an extended version of a test file used in the Bulk loading via built-in MapReduce jobs article. You can use instructions in that article to create the same test table which is used here. If you do, make sure you change the command for the `ImportTsv` job and use the `-Dimporttsv.columns=HBASE_ROW_KEY,basic:age,location:town,location:state,location:country` flag instead of `-Dimporttsv.columns=HBASE_ROW_KEY,basic:age`. Change the file and table names accordingly if necessary.

The test table's row keys are the names of the people. The first column family called `basic` has the only qualifier called `age`. The second column family called `location` has three qualifiers called `country`, `state`, and `town`. There are 997 rows in the table.

KeyOnlyFilter

Returns the key component of each key/value. Takes no arguments.

Syntax:

```
"KeyOnlyFilter ()"
```

Command example:

```
scan 'people', { FILTER => "KeyOnlyFilter ()" }
```

Result (the last eight keys):

```
 ...
 Zimmerman Gene                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Gene                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Gene                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Gene                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Madge                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Madge                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Madge                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=
 Zimmerman Madge                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=
997 row(s)
Took 0.5022 seconds
```

FirstKeyOnlyFilter

Returns the first key/value from each row. Takes no arguments.

Syntax:

```
"FirstKeyOnlyFilter ()"
```

Command example:

```
scan 'people', { FILTER => "KeyOnlyFilter ()" }
```

Result (the first and last five key/value pairs):

```
ROW                                           COLUMN+CELL
 Abbott Delia                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Abbott Howard                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Abbott Jack                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Adams Clyde                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Aguilar Myrtie                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=23
 ...
 Young Della                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=21
 Young Josephine                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Young Mattie                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=39
 Zimmerman Gene                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=35
 Zimmerman Madge                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
997 row(s)
Took 0.1058 seconds
```

PrefixFilter

Returns all key/value pairs from rows the keys of which begin with the prefix specified by the argument. Takes one argument: row key prefix.

Syntax:

```
"PrefixFilter ('Roy')"
```

Command example:

```
scan 'people', { FILTER => "PrefixFilter ('Roy')" }
```

Result:

```
ROW                                          COLUMN+CELL
 Roy Alfred                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=55
 Roy Alfred                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Roy Alfred                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 Roy Alfred                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Juarez
 Roy Belle                                   column=basic:age, timestamp=2024-07-30T08:10:19.297, value=35
 Roy Belle                                   column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Roy Belle                                   column=location:state, timestamp=2024-07-30T08:10:19.297, value=AZ
 Roy Belle                                   column=location:town, timestamp=2024-07-30T08:10:19.297, value=Nogales
 Roy Lora                                    column=basic:age, timestamp=2024-07-30T08:10:19.297, value=52
 Roy Lora                                    column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Roy Lora                                    column=location:state, timestamp=2024-07-30T08:10:19.297, value=Yucatan
 Roy Lora                                    column=location:town, timestamp=2024-07-30T08:10:19.297, value=Uman
 Roy Ronald                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=63
 Roy Ronald                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Roy Ronald                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=California
Baja
 Roy Ronald                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Mexicali
4 row(s)
Took 0.0252 seconds
```

ColumnPrefixFilter

Returns all key/value pairs from a column with a qualifier that begins with the prefix specified by the argument. Takes one argument: prefix of the column qualifier.

Syntax:

```
"ColumnPrefixFilter ('tow')"
```

Command example:

```
scan 'people', { FILTER => "ColumnPrefixFilter ('tow')" }
```

Since all rows of the test table have non-empty values for all columns, this filter returns either zero results if no column qualifier has the specified prefix, or all 997 of them if such column qualifier exists. The command given above results in 997 key/value pairs, the first and last five of which are shown below:

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Abbott Howard                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton Rouge
 Abbott Jack                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Juarez
 Adams Clyde                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Lafayette
 Aguilar Myrtie                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Uman
 ...
 Young Della                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Chetumal
 Young Josephine                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Redding
 Young Mattie                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=San Jose
 Zimmerman Gene                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Tijuana
 Zimmerman Madge                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Nogales
997 row(s)
Took 0.1360 seconds
```

MultipleColumnPrefixFilter

Returns all key/value pairs from columns with qualifiers that begin with any of the prefixes specified by the arguments. Takes one or more arguments: prefixes of the column qualifiers. In the case of a single argument, works the same as the `ColumnPrefixFilter` filter.

Syntax:

```
"MultipleColumnPrefixFilter ('sta', 'tow')"
```

Command example:

```
scan 'people', { FILTER => "MultipleColumnPrefixFilter ('sta', 'tow')" }
```

The same situation as in the previous example (the `ColumnPrefixFilter` filter) applies here: since all rows of the test table have non-empty values for all columns, this filter returns either zero results if no column qualifier has any of the specified prefixes, or all 997 of them per prefix specified if respective qualifiers exist. If one of the specified prefixes is the prefix of another (e.g. `to` and `tow`), the number of results does not multiply. The command given above results in 1994 key/value pairs, the first and last six of which are shown below:

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Abbott Delia                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Abbott Howard                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Abbott Howard                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton Rouge
 Abbott Jack                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 Abbott Jack                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Juarez
 ...
 Young Mattie                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=CA
 Young Mattie                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=San Jose
 Zimmerman Gene                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=California
Baja
 Zimmerman Gene                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Tijuana
 Zimmerman Madge                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=AZ
 Zimmerman Madge                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Nogales
997 row(s)
Took 0.1360 seconds
```

ColumnCountGetFilter

Returns columns of the row starting with the first and up to the argument value. This filter only works correctly with single rows, so it should be used with the `get` command. Using this filter with the `scan` command does not lead to an error, but provides an incorrect result. Takes one argument: number of the final column.

Syntax:

```
"ColumnCountGetFilter (3)"
```

Command example:

```
get 'people', 'Abbott Delia', { FILTER => "ColumnCountGetFilter (3)" }
```

Result:

```
COLUMN                                       CELL
 basic:age                                   timestamp=2024-07-30T08:10:19.297, value=62
 location:country                            timestamp=2024-07-30T08:10:19.297, value=USA
 location:state                              timestamp=2024-07-30T08:10:19.297, value=TX
1 row(s)
Took 0.6243 seconds
```

PageFilter

Returns rows of the table starting with the first and up to the argument value from each region. Takes one argument: number of the final row.

Syntax:

```
"PageFilter (2)"
```

Command example:

```
scan 'people', { FILTER => "PageFilter (2)" }
```

Since the test table was created with five regions split at `F`, `K`, `P`, and `W`, this command will return two rows with four key/value pairs apiece from each region. Result:

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Abbott Delia                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Delia                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Abbott Delia                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Abbott Howard                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Abbott Howard                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Howard                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Abbott Howard                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton Rouge
 Farmer Alan                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Farmer Alan                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Farmer Alan                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=Quintana Roo
 Farmer Alan                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Cancun
 Farmer Dean                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=65
 Farmer Dean                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Farmer Dean                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Farmer Dean                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton Rouge
 Keller Elmer                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Keller Elmer                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Keller Elmer                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=Quintana Roo
 Keller Elmer                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Chetumal
 Keller Flora                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=23
 Keller Flora                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Keller Flora                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Keller Flora                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Houston
 Padilla Ethan                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=61
 Padilla Ethan                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Padilla Ethan                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=NV
 Padilla Ethan                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Novac
 Padilla Scott                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=28
 Padilla Scott                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Padilla Scott                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Padilla Scott                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Beaumont
 Wade Janie                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=25
 Wade Janie                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Wade Janie                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=California
Baja
 Wade Janie                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Ensenada
 Wade Jerome                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=27
 Wade Jerome                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Wade Jerome                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Wade Jerome                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Houston
10 row(s)
Took 0.0138 seconds
```

ColumnPaginationFilter

Returns the number of columns specified by the first argument (limit) after the number of columns specified by the second argument (offset). Takes two arguments: limit and offset.

Syntax:

```
"ColumnPaginationFilter (2, 1)"
```

Command example:

```
scan 'people', { FILTER => "ColumnPaginationFilter (2, 1)" }
```

This command results in 1994 key/value pairs, the first and last six of which are shown below:

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Delia                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Abbott Howard                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Howard                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Abbott Jack                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Abbott Jack                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 ...
 Young Mattie                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Young Mattie                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=CA
 Zimmerman Gene                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Zimmerman Gene                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=California Baja
 Zimmerman Madge                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Zimmerman Madge                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=AZ
997 row(s)
Took 1.0474 seconds
```

InclusiveStopFilter

Scans the table row by row until it finds the row with the specified key, then returns all rows up to and including the one where the key was found. Takes one argument: row key.

Syntax:

```
"InclusiveStopFilter (3)"
```

Command example:

```
scan 'people', { FILTER => "InclusiveStopFilter ('Allen Austin')" }
```

Result:

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Abbott Delia                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Delia                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Abbott Delia                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Abbott Howard                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Abbott Howard                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Howard                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Abbott Howard                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton Rouge
 Abbott Jack                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Abbott Jack                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Abbott Jack                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 Abbott Jack                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Juarez
 Adams Clyde                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Adams Clyde                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Adams Clyde                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Adams Clyde                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Lafayette
 Aguilar Myrtie                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=23
 Aguilar Myrtie                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Aguilar Myrtie                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=Yucatan
 Aguilar Myrtie                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Uman
 Aguilar Terry                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=65
 Aguilar Terry                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Aguilar Terry                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=CA
 Aguilar Terry                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Fresno
 Alexander Derrick                           column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
 Alexander Derrick                           column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alexander Derrick                           column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Alexander Derrick                           column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Alexander Gregory                           column=basic:age, timestamp=2024-07-30T08:10:19.297, value=54
 Alexander Gregory                           column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alexander Gregory                           column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Alexander Gregory                           column=location:town, timestamp=2024-07-30T08:10:19.297, value=Lafayette
 Alexander Leon                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=42
 Alexander Leon                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alexander Leon                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=AZ
 Alexander Leon                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Kingman
 Allen Austin                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=34
 Allen Austin                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Allen Austin                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=NV
 Allen Austin                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Primm
10 row(s)
Took 0.7167 seconds
```

TimestampsFilter

Returns all key/value pairs with timestamps matching any of the timestamps specified by the arguments. Takes any number of arguments: timestamps.

Syntax:

```
"TimestampsFilter (1721203180857, 1721316861863)"
```

All the key/value pairs in the test table have the same timestamp, so for a more concise and illustrative output you might want to tweak a couple of them using the `put` command:

```
put 'people', 'Young Mattie', 'location:town', 'Carlsbad', 1722327020000
put 'people', 'Young Mattie', 'location:state', 'NM', 1722327020000
put 'people', 'Yates Douglas', 'location:state', 'NM', 1722327020000
put 'people', 'Yates Douglas', 'location:town', 'Albuquerque', 1722327020000
```

Command example:

```
scan 'people', { FILTER => "TimestampsFilter (1722327020000)" }
```

Result:

```
ROW                                          COLUMN+CELL
 Yates Douglas                               column=location:state, timestamp=2024-07-30T08:10:20, value=NM
 Yates Douglas                               column=location:town, timestamp=2024-07-30T08:10:20, value=Albuquerque
 Young Mattie                                column=location:state, timestamp=2024-07-30T08:10:20, value=NM
 Young Mattie                                column=location:town, timestamp=2024-07-30T08:10:20, value=Carlsbad
2 row(s)
Took 0.1001 seconds
```

RowFilter

Checks all rows and returns all key/value pairs in the row if the row key value matches the result of comparison specified by the arguments. This filter does not work with the `get` command. Takes two arguments: a comparison operator and a comparator.

Syntax:

```
"RowFilter (>=, 'binaryprefix:B')"
```

Command example:

```
scan 'people', { FILTER => "RowFilter (<, 'binaryprefix:B')" }
```

This filter should return all key/value pairs from the rows the keys of which begin with something lexicographically less than `B`. In case of our test table this basically means all rows about people whose last names begin with an `A`. Result:

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Abbott Delia                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Delia                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Abbott Delia                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Abbott Howard                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Abbott Howard                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Abbott Howard                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Abbott Howard                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton Rouge
 Abbott Jack                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Abbott Jack                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Abbott Jack                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 Abbott Jack                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Juarez
 Adams Clyde                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Adams Clyde                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Adams Clyde                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Adams Clyde                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Lafayette
 Aguilar Myrtie                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=23
 Aguilar Myrtie                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Aguilar Myrtie                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=Yucatan
 Aguilar Myrtie                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Uman
 Aguilar Terry                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=65
 Aguilar Terry                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Aguilar Terry                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=CA
 Aguilar Terry                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Fresno
 Alexander Derrick                           column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
 Alexander Derrick                           column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alexander Derrick                           column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Alexander Derrick                           column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Alexander Gregory                           column=basic:age, timestamp=2024-07-30T08:10:19.297, value=54
 Alexander Gregory                           column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alexander Gregory                           column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Alexander Gregory                           column=location:town, timestamp=2024-07-30T08:10:19.297, value=Lafayette
 Alexander Leon                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=42
 Alexander Leon                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alexander Leon                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=AZ
 Alexander Leon                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Kingman
 Allen Austin                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=34
 Allen Austin                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Allen Austin                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=NV
 Allen Austin                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Primm
 Allison Dustin                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=54
 Allison Dustin                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Allison Dustin                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=Yucatan
 Allison Dustin                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Merida
 Alvarado Dominic                            column=basic:age, timestamp=2024-07-30T08:10:19.297, value=63
 Alvarado Dominic                            column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Alvarado Dominic                            column=location:state, timestamp=2024-07-30T08:10:19.297, value=California Baja
 Alvarado Dominic                            column=location:town, timestamp=2024-07-30T08:10:19.297, value=Ensenada
 Alvarado Maria                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=58
 Alvarado Maria                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Alvarado Maria                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 Alvarado Maria                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Chihuahua
 Alvarado Melvin                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=34
 Alvarado Melvin                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alvarado Melvin                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Alvarado Melvin                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Austin
 Alvarado Timothy                            column=basic:age, timestamp=2024-07-30T08:10:19.297, value=27
 Alvarado Timothy                            column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alvarado Timothy                            column=location:state, timestamp=2024-07-30T08:10:19.297, value=CA
 Alvarado Timothy                            column=location:town, timestamp=2024-07-30T08:10:19.297, value=San Francisco
 Alvarez Bessie                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=34
 Alvarez Bessie                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alvarez Bessie                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NV
 Alvarez Bessie                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Las Vegas
 Alvarez Bruce                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=60
 Alvarez Bruce                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alvarez Bruce                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=CA
 Alvarez Bruce                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Modesto
 Alvarez Harvey                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=57
 Alvarez Harvey                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Alvarez Harvey                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NV
 Alvarez Harvey                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Fallon
 Alvarez Jacob                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=49
 Alvarez Jacob                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Alvarez Jacob                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=Yucatan
 Alvarez Jacob                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Merida
 Anderson Lester                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Anderson Lester                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Anderson Lester                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=Sonora
 Anderson Lester                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Hermosillo
 Anderson Lucile                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=33
 Anderson Lucile                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Anderson Lucile                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=Sonora
 Anderson Lucile                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Hermosillo
 Anderson Ora                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=26
 Anderson Ora                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Anderson Ora                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=Sonora
 Anderson Ora                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Nogales
 Andrews Caleb                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=22
 Andrews Caleb                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Andrews Caleb                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=Quintana Roo
 Andrews Caleb                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Cancun
 Andrews Lucy                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=33
 Andrews Lucy                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Andrews Lucy                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=AZ
 Andrews Lucy                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Kingman
 Andrews Noah                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=63
 Andrews Noah                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Andrews Noah                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=California Baja
 Andrews Noah                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Ensenada
 Andrews Susan                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=48
 Andrews Susan                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Andrews Susan                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=Yucatan
 Andrews Susan                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Merida
 Armstrong Isabella                          column=basic:age, timestamp=2024-07-30T08:10:19.297, value=49
 Armstrong Isabella                          column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Armstrong Isabella                          column=location:state, timestamp=2024-07-30T08:10:19.297, value=Sonora
 Armstrong Isabella                          column=location:town, timestamp=2024-07-30T08:10:19.297, value=Hermosillo
 Armstrong Marie                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=25
 Armstrong Marie                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Armstrong Marie                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Armstrong Marie                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Santa Fe
 Arnold Bettie                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=18
 Arnold Bettie                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Arnold Bettie                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=NV
 Arnold Bettie                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Primm
 Atkins Daisy                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=51
 Atkins Daisy                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Atkins Daisy                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=Quintana Roo
 Atkins Daisy                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Chetumal
 Atkins Gene                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=37
 Atkins Gene                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Atkins Gene                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Atkins Gene                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Austin Bertie                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=57
 Austin Bertie                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=MEX
 Austin Bertie                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=California Baja
 Austin Bertie                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Tijuana
 Austin Eugene                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=64
 Austin Eugene                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Austin Eugene                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=LA
 Austin Eugene                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Lafayette
 Austin Travis                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=53
 Austin Travis                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Austin Travis                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Austin Travis                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=El Paso
34 row(s)
Took 0.6767 seconds
```

FamilyFilter

Checks all columns and returns all key/value pairs in a column if its family name matches the result of comparison specified by the arguments. Takes two arguments: a comparison operator and a comparator.

Syntax:

```
"FamilyFilter (\<=, 'binaryprefix:c')"
```

Command example:

```
scan 'people', { FILTER => "FamilyFilter (<, 'binaryprefix:c')" }
```

This filter should return all key/value pairs the column families of which begin with something lexicographically less than `c`. In case of our test table this means the whole `basic:age` column. Result (the first and last five rows):

```
ROW                                               COLUMN+CELL
 Abbott Delia                                     column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Abbott Howard                                    column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Abbott Jack                                      column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Adams Clyde                                      column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Aguilar Myrtie                                   column=basic:age, timestamp=2024-07-30T08:10:19.297, value=23
 ...
 Young Della                                      column=basic:age, timestamp=2024-07-30T08:10:19.297, value=21
 Young Josephine                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Young Mattie                                     column=basic:age, timestamp=2024-07-30T08:10:19.297, value=39
 Zimmerman Gene                                   column=basic:age, timestamp=2024-07-30T08:10:19.297, value=35
 Zimmerman Madge                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
997 row(s)
Took 0.4836 seconds
```

QualifierFilter

Checks all columns and returns all key/value pairs in a column if its qualifier name matches the result of comparison specified by the arguments. Takes two arguments: a comparison operator and a comparator.

Syntax:

```
"QualifierFilter (=, 'binary:town')"
```

Command example:

```
scan 'people', { FILTER => "QualifierFilter (=, 'binary:town')" }
```

This filter should return all key/value pairs the column qualifier of which is exactly `town`. Result (first and last five rows):

```
ROW                                               COLUMN+CELL
 Abbott Delia                                     column=location:town, timestamp=2024-07-30T08:10:19.297, value=Dallas
 Abbott Howard                                    column=location:town, timestamp=2024-07-30T08:10:19.297, value=Baton
Rouge
 Abbott Jack                                      column=location:town, timestamp=2024-07-30T08:10:19.297, value=Juarez
 Adams Clyde                                      column=location:town, timestamp=2024-07-30T08:10:19.297,
value=Lafayette
 Aguilar Myrtie                                   column=location:town, timestamp=2024-07-30T08:10:19.297, value=Uman
 ...
 Young Della                                      column=location:town, timestamp=2024-07-30T08:10:19.297,
value=Chetumal
 Young Josephine                                  column=location:town, timestamp=2024-07-30T08:10:19.297,
value=Redding
 Young Mattie                                     column=location:town, timestamp=2024-07-30T08:10:20, value=Carlsbad
 Zimmerman Gene                                   column=location:town, timestamp=2024-07-30T08:10:19.297,
value=Tijuana
 Zimmerman Madge                                  column=location:town, timestamp=2024-07-30T08:10:19.297,
value=Nogales
997 row(s)
Took 0.2112 seconds
```

ValueFilter

Returns all key/value pairs the values of which match the result of comparison specified by the arguments. Takes two arguments: a comparison operator and a comparator.

Syntax:

```
"ValueFilter (=, 'binary:TX')"
```

Command example:

```
scan 'people', { FILTER => "ValueFilter (=, 'binary:TX')" }
```

This filter should return the list of texans from the test table, i.e. all key/value pairs which are exactly `TX`. Result (first and last five rows):

```
ROW                                               COLUMN+CELL
 Abbott Delia                                     column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Alexander Derrick                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Alvarado Melvin                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Atkins Gene                                      column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Austin Travis                                    column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 ...
 Watkins Julian                                   column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Welch Lela                                       column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Willis Travis                                    column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Wilson Grace                                     column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
 Wilson Nellie                                    column=location:state, timestamp=2024-07-30T08:10:19.297, value=TX
112 row(s)
Took 0.0535 seconds
```

DependentColumnFilter

Searches each row for the column identified by the two mandatory arguments: column family and qualifier. If such a column is found (**reference column**), returns all key/value pairs in that row that have the same timestamp as does the reference column. If not — nothing is returned.

If a third boolean argument (`dropDependentColumn`) is specified (optional), then the reference column is either returned as well (`false`) or not (`true`).

Two more arguments can be specified: a comparison operator and a comparator. In this case the column value must also match the result of comparison specified by these arguments for the column to be considered a reference one.

Takes two, three, or five arguments: column family (mandatory), column qualifier (mandatory), `dropDependentColumn` flag (boolean, optional), comparison operator, and a comparator (last two arguments can only be specified jointly).

Syntax:

```
"DependentColumnFilter ('location', 'town')"
```

```
"DependentColumnFilter ('location', 'town', true)"
```

```
"DependentColumnFilter ('location', 'town', true, =, 'binary:Carlsbad')"
```

Command example:

```
scan 'people', { FILTER => "DependentColumnFilter ('location', 'town', true, =, 'binary:Carlsbad')" }
```

This filter should search each row for the exact value `Carlsbad` in the column called exactly `location:town`, and in case of success return other key/value pairs that share the same timestamp, omitting the found one. Result:

```
ROW                                          COLUMN+CELL
 Young Mattie                                column=location:state, timestamp=2024-07-30T08:10:20, value=NM
1 row(s)
Took 0.0269 seconds
```

This example relies on the tweaking done in the example for the `TimestampsFilter`. Without it, the filter will return nothing as the comparison condition will not be met.

SingleColumnValueFilter

Searches each row for the reference column identified by four mandatory arguments: column family, column qualifier, comparison operator, and a comparator. If a column with the specified family and qualifier is found and its value matches the comparison result, all the key/value pairs in the row are returned. If a column with the specified family and qualifier is found but its value does not match the comparison result, no key/value pairs are returned. If no reference column is found, by default, all the key/value pairs in the row are returned.

Two more boolean arguments can be specified jointly: if the first (`setFilterIfMissing`) is set to `true` (default is `false`) and no reference column is found, then no key/value pairs in the row are returned. If the second (`setLatestVersionOnly`) is set to `false` (default is `true`), then all the reference column value versions are checked against the comparison result, and not only the latest one.

Takes four or six arguments: column family (mandatory), column qualifier (mandatory), comparison operator (mandatory), comparison value (mandatory), and `setFilterIfMissing` and `setLatestVersionOnly` flags (optional, jointly).

Syntax:

```
"SingleColumnValueFilter ('location', 'town', =, 'binaryprefix:Al')"
```

```
"SingleColumnValueFilter ('location', 'town', =, 'binaryprefix:Al', true, false)"
```

Command example:

```
scan 'people', { FILTER => "SingleColumnValueFilter ('location', 'town', =, 'binaryprefix:Al', true, false)" }
```

This filter should search each row for the values beginning with `Al` in the column called exactly `location:town`, checking all versions of those values. Only in case of success it returns all key/value pairs from that row. Result:

```
ROW                                          COLUMN+CELL
 Ball Nelle                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=43
 Ball Nelle                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Ball Nelle                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Ball Nelle                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Bell Leila                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=58
 Bell Leila                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Bell Leila                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Bell Leila                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Cohen John                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=28
 Cohen John                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Cohen John                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Cohen John                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Diaz Anne                                   column=basic:age, timestamp=2024-07-30T08:10:19.297, value=57
 Diaz Anne                                   column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Diaz Anne                                   column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Diaz Anne                                   column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Howard Florence                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=37
 Howard Florence                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Howard Florence                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Howard Florence                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Ingram Barbara                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=55
 Ingram Barbara                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Ingram Barbara                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Ingram Barbara                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Jefferson Charlie                           column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
 Jefferson Charlie                           column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Jefferson Charlie                           column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Jefferson Charlie                           column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Kennedy Todd                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=44
 Kennedy Todd                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Kennedy Todd                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Kennedy Todd                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 McGee Isabelle                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=48
 McGee Isabelle                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 McGee Isabelle                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 McGee Isabelle                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Page Victoria                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=30
 Page Victoria                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Page Victoria                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Page Victoria                               column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Phelps Lida                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=43
 Phelps Lida                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Phelps Lida                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Phelps Lida                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Phillips Helen                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=61
 Phillips Helen                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Phillips Helen                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Phillips Helen                              column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Reyes Marc                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=25
 Reyes Marc                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Reyes Marc                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Reyes Marc                                  column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Roberts Clayton                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=52
 Roberts Clayton                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Roberts Clayton                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Roberts Clayton                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Ryan Curtis                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=58
 Ryan Curtis                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Ryan Curtis                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Ryan Curtis                                 column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Spencer Lucinda                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Spencer Lucinda                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Spencer Lucinda                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Spencer Lucinda                             column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Woods Bessie                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=47
 Woods Bessie                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Woods Bessie                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Woods Bessie                                column=location:town, timestamp=2024-07-30T08:10:19.297, value=Albuquerque
 Yates Douglas                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=35
 Yates Douglas                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Yates Douglas                               column=location:state, timestamp=2024-07-30T08:10:20, value=NM
 Yates Douglas                               column=location:town, timestamp=2024-07-30T08:10:20, value=Albuquerque
18 row(s)
Took 0.0263 seconds
```

SingleColumnValueExcludeFilter

Works the same as the `SingleColumnValueFilter` filter except that the reference column value is never returned in the results.

Syntax:

```
"SingleColumnValueExcludeFilter ('location', 'town', =, 'binaryprefix:Al')"
```

```
"SingleColumnValueExcludeFilter ('location', 'town', =, 'binaryprefix:Al', true, false)"
```

Command example:

```
scan 'people', { FILTER => "SingleColumnValueExcludeFilter ('location', 'town', =, 'binaryprefix:Al', true, false)" }
```

This filter should search each row for the values beginning with `Al` in the column called exactly `location:town`, checking all versions of those values. Only in case of success it returns other key/value pairs from that row but not the found one. Result:

```
ROW                                          COLUMN+CELL
 Ball Nelle                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=43
 Ball Nelle                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Ball Nelle                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Bell Leila                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=58
 Bell Leila                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Bell Leila                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Cohen John                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=28
 Cohen John                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Cohen John                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Diaz Anne                                   column=basic:age, timestamp=2024-07-30T08:10:19.297, value=57
 Diaz Anne                                   column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Diaz Anne                                   column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Howard Florence                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=37
 Howard Florence                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Howard Florence                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Ingram Barbara                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=55
 Ingram Barbara                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Ingram Barbara                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Jefferson Charlie                           column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
 Jefferson Charlie                           column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Jefferson Charlie                           column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Kennedy Todd                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=44
 Kennedy Todd                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Kennedy Todd                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 McGee Isabelle                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=48
 McGee Isabelle                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 McGee Isabelle                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Page Victoria                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=30
 Page Victoria                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Page Victoria                               column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Phelps Lida                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=43
 Phelps Lida                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Phelps Lida                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Phillips Helen                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=61
 Phillips Helen                              column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Phillips Helen                              column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Reyes Marc                                  column=basic:age, timestamp=2024-07-30T08:10:19.297, value=25
 Reyes Marc                                  column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Reyes Marc                                  column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Roberts Clayton                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=52
 Roberts Clayton                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Roberts Clayton                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Ryan Curtis                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=58
 Ryan Curtis                                 column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Ryan Curtis                                 column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Spencer Lucinda                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Spencer Lucinda                             column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Spencer Lucinda                             column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Woods Bessie                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=47
 Woods Bessie                                column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Woods Bessie                                column=location:state, timestamp=2024-07-30T08:10:19.297, value=NM
 Yates Douglas                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=35
 Yates Douglas                               column=location:country, timestamp=2024-07-30T08:10:19.297, value=USA
 Yates Douglas                               column=location:state, timestamp=2024-07-30T08:10:20, value=NM
18 row(s)
Took 0.0646 seconds
```

ColumnRangeFilter

Returns only key/value pairs with column qualifier names in the range set by the arguments. The range ends can be definitive or empty. Each range end is followed by a boolean argument that defines whether the range end is included or not.

Takes four arguments: left range end, left inclusion flag, right range end, right inclusion flag. If a range end is empty, then its respective inclusion flag value does not matter.

Syntax:

```
"ColumnRangeFilter ('', true, 'c', false)"
```

Command example:

```
scan 'people', { FILTER => "ColumnRangeFilter ('', true, 'c', false)" }
```

This filter will only return the `basic:age` column, since the next column qualifier alphabetically is `country` and it does not fall into the specified range. Result (the first and last five rows):

```
ROW                                          COLUMN+CELL
 Abbott Delia                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=62
 Abbott Howard                               column=basic:age, timestamp=2024-07-30T08:10:19.297, value=24
 Abbott Jack                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Adams Clyde                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Aguilar Myrtie                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=23
 ...
 Young Della                                 column=basic:age, timestamp=2024-07-30T08:10:19.297, value=21
 Young Josephine                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=29
 Young Mattie                                column=basic:age, timestamp=2024-07-30T08:10:19.297, value=39
 Zimmerman Gene                              column=basic:age, timestamp=2024-07-30T08:10:19.297, value=35
 Zimmerman Madge                             column=basic:age, timestamp=2024-07-30T08:10:19.297, value=46
997 row(s)
Took 0.1788 seconds
```

Dynamic loading of custom filters

HBase in ADH supports dynamic loading of custom filters. To utilize this feature, you should specify the directory containing the custom filter JAR files:

1. Go to ADCM UI and select your ADH cluster.
2. Navigate to ***Services → HBase → Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
4. For the field name, enter `hbase.dynamic.jars.dir`. For the field value, enter a path of your preference. A good example is `${hbase.rootdir}/lib`. Click ***Apply***.
5. Save the configuration by clicking ***Save → Create*** and restart the service by clicking ***Actions → Reconfig and graceful restart***.

Provided there are JAR files with the custom filters in the specified location, you should be able to use them both in HBase shell and via Java applications using the HBase API.

Use coprocessors in HBase

Vladimir Adamenkov

Contents

To Table of Contents

- Coprocessors overview
  - Observer coprocessor
  - Endpoint coprocessor
- Coprocessor loading and unloading
  - Static coprocessor loading
  - Static coprocessor unloading
  - Dynamic coprocessor loading
  - Dynamic coprocessor unloading

Coprocessors overview

The HBase coprocessors feature is an extension mechanism (framework) that allows you to perform distributed computations by executing user code directly at where the data is stored.

Observer coprocessor

Observer coprocessor (or **observer**) is intended to create hooks allowing to monitor table operations from the client side. Those operations include `get`, `put`, `scan`, `delete`, and other requests. Three kinds of observers are presented in HBase:

- **RegionObserver** — provides hooks for data manipulation requests like `get`, `put`, `scan`, `delete`, and others. An instance of RegionObserver is allocated for every table region. The scope of each instance is restricted to its region.
- **WALObserver** — provides hooks for the operations of the write ahead log (WAL). WAL writing and reconstruction events can be observed or intercepted by the WALObserver. There is one WAL processing context per region server, with one WALObserver residing in it.
- **MasterObserver** — provides hooks for the operations related to data definition like `create table`, `delete table`, `modify table`, and others. A MasterObserver resides in the HBase master context.

An observer can observe and meddle in the actions performed by the client in the region. The following functions are available for an observer.

Observer functions

Function	Description
preGet, postGet	Called before and after the client makes a `get` request, respectively
preExists, postExists	Called before and after the client checks the existence of the data by using a `get` request, respectively
prePut, postPut	Called before and after the client writes the data by using a `put` request, respectively
preDelete, postDelete	Called before and after the client deletes the data by using a `delete` request, respectively
preOpen, postOpen	Called before and after the region is reported as online to the HBase master, respectively
preFlush, postFlush	Called before and after the memstore is flushed to the new store file, respectively
preScannerOpen, postScannerOpen	Called before and after the client invokes a new scanner by using a `scan` request, respectively
preScannerNext, postScannerNext	Called before and after the next row is requested by the client in an opened scanner, respectively
preScannerClose, postScannerClose	Called before and after the client closes a scanner, respectively
preCheckAndPut, postCheckAndPut	Called before and after the client calls the `checkAndPut()` function, respectively
preCheckAndDelete, postCheckAndDelete	Called before and after the client calls the `checkAndDelete()` function, respectively

Below is the code for a simple observer. It checks the user information for the `get` request by injecting code at certain `preGet` hooks. It will throw a CoprocessorException if the user is not allowed to access the resource, which will in turn deny the client request.

```
package org.apache.hadoop.hbase.coprocessor;

import org.apache.hadoop.hbase.client.Get;

public class AccessControlCoprocessor extends BaseRegionObserverCoprocessor {

  public Get preGet(CoprocessorEnvironment e, Get get)
      throws CoprocessorException {

    if (access_not_allowed)  {
      throw new AccessDeniedException("User access denied");
    }
    return get;
  }
}
```

Endpoint coprocessor

Endpoint coprocessor (or **endpoint**) allows you to run custom code at a region when triggered. For example, you can perform column aggregation at any given region server.

An endpoint allows you to define a custom dynamic RPC protocol for communication between the client and the region server. For example, it is possible to create a method that specifies custom request parameters and return types. RPC methods exposed by an endpoint can be triggered by calling client side dynamic RPC functions.

Coprocessor loading and unloading

A coprocessor needs to be loaded before the HBase can use it. Static loading is done through the HBase configuration and dynamic loading is done using the HBase shell or the Java API.

Static coprocessor loading

To statically load a coprocessor, do the following:

1. Place the JAR file with the coprocessor code into the *lib/* directory of the HBase installation path on all cluster hosts where the coprocessor is required.
2. Go to ADCM UI and select your ADH cluster.
3. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
4. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
5. Specify one of the following names:
   - `hbase.coprocessor.region.classes` — for a RegionObserver or an endpoint;
   - `hbase.coprocessor.wal.classes` — for a WALObserver;
   - `hbase.coprocessor.master.classes` — for a MasterObserver.
6. Specify the fully-qualified class name of the coprocessor implementation class as a property value. Multiple classes should have their names separated by commas.
7. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Statically loaded coprocessors are active on all regions of all tables. The first coprocessor has the priority `Coprocessor.Priority.SYSTEM`. All others have integer priorities in the order of their declaration. You can override the priority value by adding the new value to the class name separated by the pipe character. Example: `org.myname.hbase.coprocessor.endpoint.SumEndPoint|21`.

Static coprocessor unloading

To statically unload a coprocessor, do the following:

1. Delete the property from the ***Custom hbase-site.xml*** section that was created for the coprocessor.
2. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.
3. (Optionally) Delete the JAR file from the *lib/* directory of the HBase installation.

Dynamic coprocessor loading

To dynamically load a coprocessor, go to HBase shell and execute a command of the following kind:

```
alter '<table_name>', METHOD => 'table_att', 'coprocessor'=>'hdfs://<namenode>:<port>/<coprocessor.jar>|<class_name>|<priority>|
<args>'
```

where:

- `<table_name>` — the name of the table for which the coprocessor is being loaded;
- `<namenode>:<port>` — network address of the node containing the coprocessor code JAR file. If the file is stored on the same node, you can just drop this part, minding the triple slash character: `hdfs:///`;
- `<coprocessor.jar>` — full path to the JAR file on the filesystem;
- `<class_name>` — full class name of the coprocessor;
- `<priority>` — an integer representing the coprocessor priority;
- `<args>` — arguments taken by the coprocessor implementation.

Example:

```
alter 't1', METHOD => 'table_att', 'coprocessor'=>'hdfs://<namenode>:<port>/user/<hadoop-user>/coprocessor.jar|org.myname.hbase.
Coprocessor.RegionObserverExample|1037|arg1=1,arg2=2'
```

To verify that the coprocessor has loaded, use the `describe` command. The coprocessor info should appear in the output right after the table name.

Dynamic coprocessor unloading

To dynamically unload a coprocessor, go to HBase shell and execute a command of the following kind:

```
alter '<table_name>', METHOD => 'table_att_unset', NAME => 'coprocessor$1'
```

Scan over snapshots

Vladimir Adamenkov

Contents

To Table of Contents

Overview
- TableSnapshotScanner class
- TableSnapshotInputFormat class

Configuration

Overview

Scan over snapshot is a feature that allows you to scan data directly from HFiles without using HBase. This works regardless of whether the HBase is up and helps to save resources on the HBase server side.

Scan over snapshot is provided by the `TableSnapshotScanner` and `TableSnapshotInputFormat` classes.

TableSnapshotScanner class

The `TableSnapshotScanner` class is intended for single scans over the HFiles run from the client side. Those HFiles are copied into a temporary directory and deleted when the scanner is closed. This directory must not be a subdirectory of the path specified in the `hbase.rootdir` parameter in the *hbase-site.xml* configuration file. The user on the client side must also have write permissions for that directory.

Below is an example of the `TableSnapshotScanner` class usage:

```
Path restoreDir = new Path("<path>");
Scan scan = new Scan();
try (TableSnapshotScanner scanner = new TableSnapshotScanner(conf, restoreDir, snapshotName, scan)) {
    Result result = scanner.next();
    while (result != null) {
        ...
        result = scanner.next();
    }
}
```

where `<path>` is the path to the temporary directory.

TableSnapshotInputFormat class

The `TableSnapshotInputFormat` class is intended for scanning HFiles and use the results in MapReduce jobs.

Below is an example of the `TableSnapshotInputFormat` class usage:

```
Job job = new Job(conf);
Path restoreDir = new Path(",<path>");
Scan scan = new Scan();
TableMapReduceUtil.initTableSnapshotMapperJob(snapshotName, scan, MyTableMapper.class, MyMapKeyOutput.class,
MyMapOutputValueWritable.class, job, true, restoreDir);
```

Configuration

To be able to use the scan over snapshot feature, do the following:

1. Go to ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***HDFS*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hdfs-site.xml*** section and click ***Add property***.
4. For the property name, enter `dfs.namenode.acls.enabled` and set its value to `true`.
5. Open the ***Custom core-site.xml*** section and click ***Add property***.
6. For the property name, enter `fs.permissions.umask-mode` and set its value to `027`.
7. Save the configuration by clicking ***Save*** → ***Create***.
8. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
9. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
10. For the property name, enter `hbase.coprocessor.master.classes` and set its value to `"org.apache.hadoop.hbase.security.access.AccessController,org.apache.hadoop.hbase.security.access.SnapshotScannerHDFSAclController"`.
11. Add another property. For the property name, enter `hbase.acl.sync.to.hdfs.enable` and set its value to `true`.
12. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.
13. Go to HBase shell and modify the required table scheme to enable the snapshot scanning by executing the command of the following kind:

```
alter '<table>', CONFIGURATION => {'hbase.acl.sync.to.hdfs.enable' => 'true'}
```

where `<table>` is the name of the table.

Use medium-sized objects in HBase

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
  - Introduction
  - Architecture
  - Maintenance
  - Compaction
- Usage
  - Configure MOB columns
  - Configure a compaction policy
  - Configure an archiving policy
- Optimization
  - Configure MOB cache
  - Manual compaction

Overview

Introduction

Medium-sized object (**MOB**) is a type of data stored in HBase tables with the order of size from 100 KB to 10 MB or greater. Examples of such objects include images, documents, archives, and other binary files.

Technically, HBase is able to deal with values of any size, but objects larger than 100 KB can and will negatively affect the performance during MemStore flushes: splits and compactions will inevitably lead to write amplification. To avoid that, you can designate certain column families containing large objects in their cells as MOB column families. Table cells supposed to store MOBs only contain the references to the actual files. Those, in turn, are stored in special MOB HFiles which can be located at any specified location (HDFS or not).

Regular column family cells contain data as it is, regardless of size. MOB column family cells discriminate the data by its size according to the specified threshold: smaller data is stored as is, larger data is stored as MOBs.

Architecture

Initially, all data to be written into HBase for the first time is stored in the write-ahead log (WAL) and the MemStore. When a MemStore corresponding to a MOB column family flushes, it creates two HFiles. Cell values smaller than the specified threshold are written into a regular HFile. Cell values greater than (or equal to) the specified threshold are written into a special MOB HFile, and at the same time the references to the locations of those values are written into the corresponding column cells (MOB reference cells) of the regular HFile. After a MemStore is flushed and its corresponding regular HFile is closed, the region server appends the metadata to it. This metadata contains the list of all MOB HFiles referenced by the cells in the regular HFile.

The value of a MOB reference cell consists of two parts: the size of the MOB and the name of the corresponding MOB HFile. It also has two tags: the first one flags the cell as the reference one, and the second one contains the namespace and the table name for the MOB HFile. These tags can help when searching specifically for the MOB cells and to optimize the way the MOB system looks for values in MOB HFiles, especially after several HBase snapshot operations. They cannot be sent over using RPCs by default as they are only available on the HBase servers side.

Upon creation or compaction the MOB HFiles are put into a specific location in the HBase root directory, according to their namespace, table name, MOB logical region, and column family name. An example of path to such location: */hbase/mobdir/data/default/table_with_mobs/463b0c18d2eb1a65b4b940837d6d0a08/mob_column/*, where:

- `hbase` — HBase root directory;
- `mobdir/data` — default subfolder for MOB HFiles;
- `default` — namespace title;
- `table_with_mobs` — table name;
- `463b0c18d2eb1a65b4b940837d6d0a08` — logical region ID;
- `mob_column` — column family name.

Maintenance

Maintenance of the MOB HFiles is performed by chore routines (chores) on the HBase master server and on the region servers. They are responsible for enforcing the column families TTLs and for compacting the HFiles. When a MOB HFile becomes obsolete after a compaction procedure, a chore moves it to the archive exactly like it does with regular HFiles. The MOB region of a table is independent from the regular regions, so it can share their archive storage location. An example of path to such location: */hbase/archive/data/default/table_with_mobs/463b0c18d2eb1a65b4b940837d6d0a08/mob_column/*.

If there exists a snapshot of the table with MOB feature turned on (or a clone of such), the MOB HFiles will remain in the archive as long as those snapshots require them to. The cleaning chores delete the MOB HFiles from the archive when they are no longer needed.

Compaction

During the flush of a MemStore associated with a MOB column family, the HBase writes the values with sizes over the threshold into the MOB HFile. During the region compaction procedure the region server overwrites the data in the regular HFiles, but keeps the references to the MOB HFiles intact. The client can transparently access the original reference data, because the region server internal routines keep track of it and pull the required values from MOB HFiles. This allows you to accumulate a big amount of MOB HFiles without affecting the average speed of obtaining the MOB cells values. Compaction of MOB HFiles is required significatly less often.

If the MOB cells values are freqently deleted or modified, it may cause excessive consumption of space. In order to delete a MOB HFile and release space it is necessary to make sure that no cells have references to it, and its current values must be written into a new MOB HFile. Also, if the filesystem restricts the maximum number of files it can contain (e.g. HDFS), then even if the MOB cells values are not modified or deleted at all, the HFiles will pile up and require to be compacted.

Usage

Configure MOB columns

In HBase shell, two properties define the MOB designation of a column family:

- `IS_MOB` — boolean. Defines whether the values in the column can be considered MOBs.
- `MOB_THRESHOLD` — integer. Defines the number of bytes starting at which the value is considered a MOB.

The `MOB_THRESHOLD` property is optional. If left unspecified, it defaults to `102400` (100 KB).

HBase shell example for a command to create a new table with a column family designated to store MOBs:

```
create 'table_with_mobs', {NAME => 'mob_column', IS_MOB => true, MOB_THRESHOLD => 204800}
```

In this example the `table_with_mobs` table is created with the `mob_column` column family intended to store MOBs. All values larger than 200 KB will be treated as MOBs in this column family.

HBase shell example for a command to add a new column family to this table designated to store MOBs:

```
alter 'table_with_mobs', {NAME => 'anoter_mob_column', IS_MOB => true, MOB_THRESHOLD => 153600}
```

In this example the column family `anoter_mob_column` is added to the `table_with_mobs` table. It is also intended to store MOBs, and its threshold is set at 150 KB.

Java API example of a code snippet aimed to configure a MOB column family:

```
HColumnDescriptor mob_column = new HColumnDescriptor("f");
mob_column.setMobEnabled(true);
mob_column.setMobThreshold(102400L);
```

In this example the `mob_column` column family is created intended to store MOBs with the default threshold of 100 KB.

Configure a compaction policy

A specific chore on the HBase master server periodically launches a major compaction procedure for both regular and MOB HFiles. By default it starts once a week, but that can be changed:

1. Go to ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
4. For the field name, enter `hbase.mob.compaction.chore.period`. For the field value, enter a period length in seconds. For example, day is `86400` and month is `2592000`. Click ***Apply***.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

By default this chore tries to make all regions perform compactions simultaneously and maximize the load on the cluster. You can even the load by setting the limit of concurrently running compaction jobs on different regions:

1. Go to ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
4. For the field name, enter `hbase.mob.major.compaction.region.batch.size`. For the field value, enter an integer limit. Zero means the default behavior (all regions working at once). Click ***Apply***.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Configure an archiving policy

A specific chore on the HBase master server periodically launches an archivation procedure for MOB HFiles when they are no longer needed. A MOB HFile is sent to the archive in two cases:

- it is older than its column family's TTL;
- no regular HFiles in all regions have references to it.

By default this chore runs daily. This is good for most cases. However, if your MOB column families are updated rather often or have short TTLs, which causes frequent compactions, you may want to adjust the schedule of this chore:

1. Go to ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
4. For the field name, enter `hbase.master.mob.cleaner.period`. For the field value, enter the period value in seconds. Click ***Apply***.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Optimization

Configure MOB cache

The MOB HFiles are not always kept open because there can be a very large number of them compared to regular HFiles. The reader cache containing MOB HFiles keeps the most recently used ones (LRU cache). You can configure the following parameters:

- **Cache size** — limit to the number of opened file handlers to cache. Big cache size value may improve the performance by reducing the freqency of opening and closing the files. Too big cache size value may cause an error.
- **Eviction period** — timeout for an unused opened file to be closed and removed from the cache.
- **Eviction to retention ratio** — a factor between 0 and 1 determining how many files must be left in the cache when the cache size is exceeded. Upon reaching the cache size limit, the total number of cached file handlers is multiplied by this factor and the resulting number of most recently used remain in the cache; others are evicted.

To add these parameters, do the following:

1. Go to ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
   Use the following values to configure one or more of the required properties:
   - Cache size — for the field name, enter `hbase.mob.file.cache.size`. For the field value, enter an integer limit. Default is `1000`.
   - Eviction period — for the field name, enter `hbase.mob.cache.evict.period`. For the field value, enter an integer duration in seconds. Default is `3600` (one hour).
   - Eviction to retention ratio — for the field name, enter `hbase.mob.cache.evict.remain.ratio`. For the field value, enter a float factor between `0.0f` and `1.0f`. Default is `0.5f`.
4. Click ***Apply***.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Manual compaction

If you don't want to wait for the compaction chore to start on schedule, you can launch the compaction procedure manually. Go to the HBase shell and use one of the commands of the following kind:

```
major_compact 'table_with_mobs'
major_compact 'table_with_mobs', 'mob_column'
```

The first command will make all the MOB column families of the table to undergo the compaction process. The second one will only process the specified column family.

Example of the same request in the Java API:

```
Admin.majorCompact(table_with_mobs, mob_column);
```

Bulk loading in HBase

Daria Barysheva

To Table of Contents

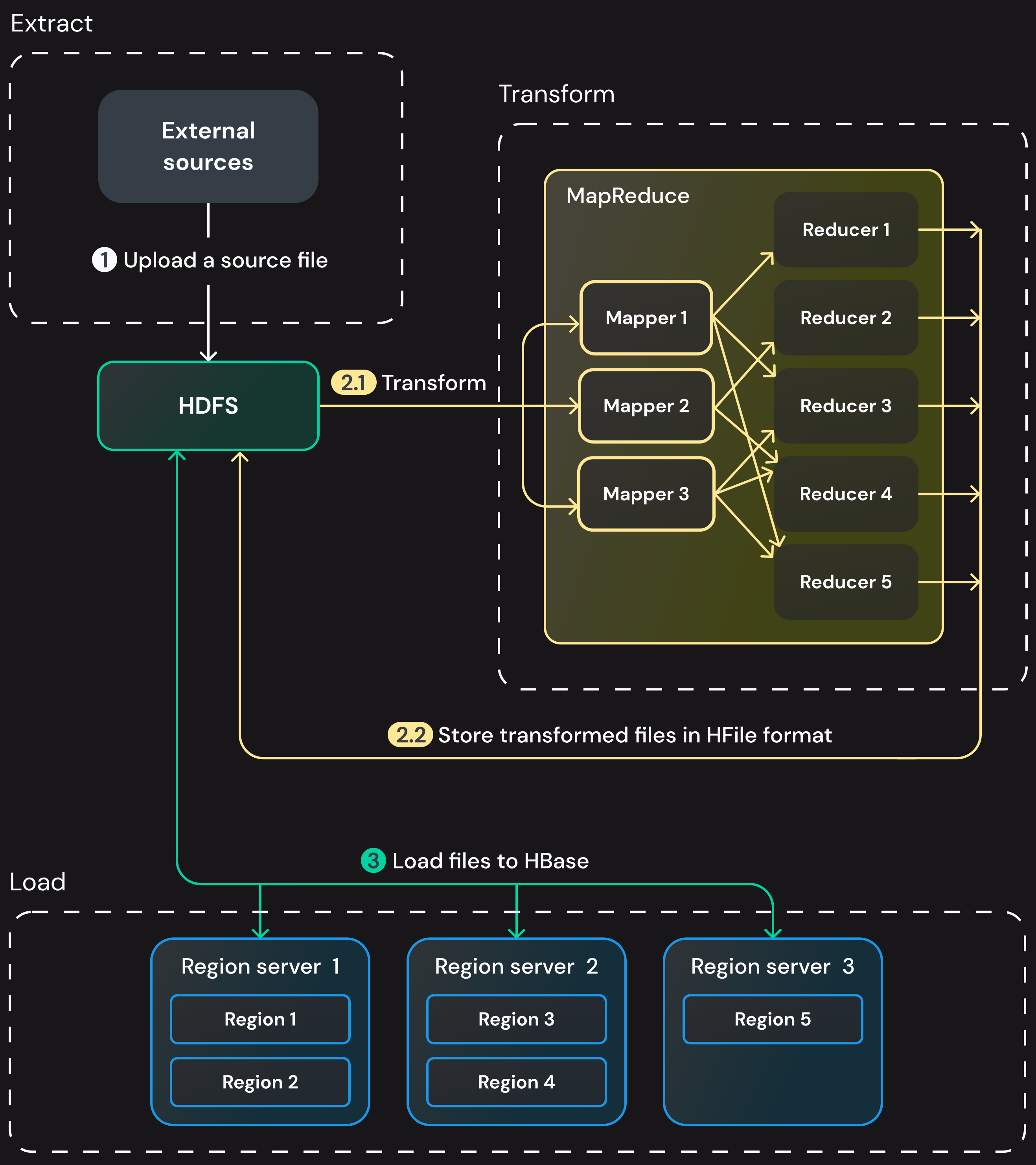
The typical write sequence in HBase includes the following steps: *WAL → MemStore → HFiles*. It works well in the normal mode, but when loading large amounts of data at the same time, this flow can lead to significant performance losses. Some of the possible problems are listed below:

- Increasing the volume requirements for MemStore and WAL.
- Expansion of compaction and flush queues.
- As a consequence, high latency, going out of SLA.

Bulk loading is designed to avoid these problems. It represents the process of preparing and loading data directly into HFiles in HDFS, bypassing the standard write path. This way allows you to insert the same amount of data more quickly. It can be useful for initial loading of datasets to HBase from external sources or for incremental insertion of large data batches, made, for example, during long night auto tasks.

Bulk loading is an ETL process that includes three main steps:

1. **Extract**. At this step, the process extracts data from the external source and puts it to HDFS. The input data can come from dumps of other databases or log files and presented in the format of plain text, CSV, or other. HBase does not manage this part of the process.
2. **Transform**. At this step, the source data is transformed into HFiles according to the structure of the predefined HBase table: one HFile is created per one region. The data is sorted alphabetically and distributed between files, according to the specified split points. This step usually requires a MapReduce job that is responsible for dividing data into the right number of partitions, sorting, generating a key/value class, matching HFile format requirements, and other.
3. **Load**. At this step, the HFiles generated earlier are moved into HBase. Each file is loaded into the relevant region through the Region Server that serves it.


Extract
External sources
① Upload a source file
HDFS
2.1 Transform
Transform
MapReduce
Mapper 1
Mapper 2
Mapper 3
Reducer 1
Reducer 2
Reducer 3
Reducer 4
Reducer 5
2.2 Store transformed files in HFile format
③ Load files to HBase
Load
Region server 1
Region 1
Region 2
Region server 2
Region 3
Region 4
Region server 3
Region 5

Bulk loading typical schema

There are many ways to configure bulk loading in HBase including built-in MapReduce jobs, custom MapReduce jobs, Spark jobs, and other.

Bulk loading via built-in MapReduce jobs

Daria Barysheva

Contents

To Table of Contents

Step 1. Extract

Step 2. Transform

Step 3. Load

This article describes the simplest way of HBase bulk loading based on the built-in ImportTsv and LoadIncrementalHFiles (also known as CompleteBulkLoad) utilities.

Step 1. Extract

In the first step, it is necessary to prepare sample data and upload it to HDFS. You can use the test file people_ages.csv. This file contains randomly generated names and ages of one thousand people. Notice, that there are three duplicates in the names: `McGee Isabelle`, `Reid Janie`, and `Summers Blanche` occur twice. Let’s leave these names intact to demonstrate the behavior of the system in such a case.

Do the following:

1. Download the file to one of the servers of your HBase cluster. Make sure that the file is uploaded successfully using the following command:

```
$ ls -la ~
```

The result looks similar to this:

```
total 56
drwx------. 5 dasha dasha   251 Nov 26 07:46 .
drwxr-xr-x. 3 root  root     19 Aug 31 11:54 ..
drwx------. 3 dasha dasha    17 Aug 31 15:06 .ansible
-rw-------. 1 dasha dasha 14913 Nov 25 16:07 .bash_history
-rw-r--r--. 1 dasha dasha    18 Apr  1  2020 .bash_logout
-rw-r--r--. 1 dasha dasha   193 Apr  1  2020 .bash_profile
-rw-r--r--. 1 dasha dasha   231 Apr  1  2020 .bashrc
drwxrwxrwx. 2 dasha dasha    64 Nov 26 07:45 dasha
-rw-rw-r--. 1 dasha dasha 17651 Nov 26 07:46 people_ages.csv
drwx------. 2 dasha dasha    29 Sep 23 17:35 .ssh
```

2. Create a directory for your user in HDFS if it does not exist yet:

```
$ sudo -u hdfs hdfs dfs -mkdir /user/dasha
$ hdfs dfs -ls /user
```

The output looks similar to the following:

```
Found 5 items
drwxr-xr-x   - hdfs   hadoop          0 2021-11-26 09:56 /user/dasha
drwx------   - hdfs   hadoop          0 2021-08-31 16:15 /user/hdfs
drwxr-xr-x   - mapred hadoop          0 2021-08-31 16:22 /user/history
drwxr-xr-x   - mapred mapred          0 2021-08-31 16:21 /user/mapred
drwxr-xr-x   - yarn   yarn            0 2021-09-01 06:57 /user/yarn
```

3. Expand access rights to your folder using the following command (see more information about file protection in Protect files in HDFS):

```
$ sudo -u hdfs hdfs dfs -chmod 777 /user/dasha
$ hdfs dfs -ls /user
```

The output looks similar to the following:

```
Found 5 items
drwxrwxrwx   - hdfs   hadoop          0 2021-11-26 09:57 /user/dasha
drwx------   - hdfs   hadoop          0 2021-08-31 16:15 /user/hdfs
drwxr-xr-x   - mapred hadoop          0 2021-08-31 16:22 /user/history
drwxr-xr-x   - mapred mapred          0 2021-08-31 16:21 /user/mapred
drwxr-xr-x   - yarn   yarn            0 2021-09-01 06:57 /user/yarn
```

4. Copy the file from the local file system to HDFS:

```
$ hdfs dfs -copyFromLocal ~/people_ages.csv /user/dasha
```

5. Make sure that the file is located in HDFS:

```
$ hdfs dfs -ls /user/dasha
```

The result looks similar to this:

```
Found 1 items
-rw-r--r--   3 dasha hadoop      17651 2021-11-26 09:57 /user/dasha/people_ages.csv
```

Step 2. Transform

In the second step, you should create a table in HBase and then use the built-in MapReduce job ImportTsv to transform the previously uploaded file into multiple HFiles according to the preconfigured table:

1. Run HBase shell:

```
$ hbase shell
```

2. Create a new table `people_ages` with one column family `basic`. Four split points will be used for generating five regions depending on the row keys values:

```
create 'people_ages', {NAME => 'basic', VERSIONS => 5}, {SPLITS => ['F', 'K', 'P', 'W']}
```

The output looks similar to this:

```
Created table people_ages
Took 1.6675 seconds
=> Hbase::Table - people_ages
```

3. Log out from HBase shell and run the MapReduce job ImportTsv:

```
$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator=',' -Dimporttsv.columns=HBASE_ROW_KEY,basic:age -
Dimporttsv.bulk.output=test_output people_ages /user/dasha/people_ages.csv
```

The arguments of this utility are listed in the following table.

Arguments

Dimporttsv.separator	A file separator. In our example, it is a comma
Dimporttsv.columns	A full list of columns contained in the imported text. In our example, using `HBASE_ROW_KEY,basic:age` means that the first column of the input text contains unique row keys, the second one — the values of the table column `basic:age`
Dimporttsv.bulk.output	A relative path to the HDFS directory where HFiles should be written to. In our example, using `test_output` means the following path: */user/dasha/test_output*

The last positions in the command arguments indicate the name of the table (`people_ages`) and the path to the source file in HDFS (`/user/dasha/people_ages.csv`).

TIP

The input argument `Dimporttsv.bulk.output` is optional. If you do not use it, data is loaded to HBase according to the typical write sequence. So, you can also use the MapReduce job ImportTsv for simple reading from CSV files (without making the subsequent steps). But if you need to load bulk data directly to HFiles, this argument must be defined.

4. Check the contents of the *test_output* directory:

```
$ hdfs dfs -ls /user/dasha/test_output
```

It contains a folder named *basic* according to the column family name in our table:

```
Found 2 items
-rw-r--r--   3 dasha hadoop          0 2021-11-26 09:59 /user/dasha/test_output/_SUCCESS
drwxr-xr-x   - dasha hadoop          0 2021-11-26 09:59 /user/dasha/test_output/basic
```

5. Check the contents of the *basic* folder:

```
$ hdfs dfs -ls /user/dasha/test_output/basic
```

It contains five HFiles — one file per region. These files, generated by ImportTsv, include data from the source file sorted alphabetically and stored in the HFile format:

```
Found 5 items
-rw-r--r--   3 dasha hadoop      16383 2021-11-26 09:59 /user/dasha/test_output/basic/0c17ccdbda7a465281ee063fa806ee44
-rw-r--r--   3 dasha hadoop      15511 2021-11-26 09:59 /user/dasha/test_output/basic/13d31777206148dab3fbf3bf3c2c3ad0
-rw-r--r--   3 dasha hadoop      14111 2021-11-26 09:59 /user/dasha/test_output/basic/5018aca2586840b5a983eff7a9b58e3f
-rw-r--r--   3 dasha hadoop      17110 2021-11-26 09:59 /user/dasha/test_output/basic/c5f4176da2764378826652cb24b6cb05
-rw-r--r--   3 dasha hadoop       9313 2021-11-26 09:59 /user/dasha/test_output/basic/de11b176c73a4aa29133c0064c1e5d52
```

NOTE

In this step, data is not loaded to the table. If you run HBase shell and apply the `scan` command to the table, it will return the empty result.

Step 3. Load

In this step, it is necessary to upload the prepared HFiles into HBase using the built-in LoadIncrementalHFiles utility:

1. In order to give HBase the permission to move the prepared files, change their owner to `hbase`:

```
$ sudo -u hdfs hdfs dfs -chown -R hbase:hbase /user/dasha/test_output
```

Make sure the owner of the *test_output* folder and all files in it is changed:

```
$ hdfs dfs -ls /user/dasha/
```

The output is similar to the following:

```
Found 4 items
drwx------   - dasha hadoop          0 2021-11-26 09:59 /user/dasha/.staging
drwxrwxrwx   - dasha hadoop          0 2021-11-26 09:59 /user/dasha/hbase-staging
-rw-r--r--   3 dasha hadoop      17651 2021-11-26 09:57 /user/dasha/people_ages.csv
drwxr-xr-x   - hbase hbase           0 2021-11-26 09:59 /user/dasha/test_output
```

The owner of the folder is changed as expected. Let’s take a look at the files in this folder:

```
$ hdfs dfs -ls /user/dasha/test_output/basic
```

The output is similar to the following:

```
Found 5 items
-rw-r--r--   3 hbase hbase      16383 2021-11-26 09:59 /user/dasha/test_output/basic/0c17ccdbda7a465281ee063fa806ee44
-rw-r--r--   3 hbase hbase      15511 2021-11-26 09:59 /user/dasha/test_output/basic/13d31777206148dab3fbf3bf3c2c3ad0
-rw-r--r--   3 hbase hbase      14111 2021-11-26 09:59 /user/dasha/test_output/basic/5018aca2586840b5a983eff7a9b58e3f
-rw-r--r--   3 hbase hbase      17110 2021-11-26 09:59 /user/dasha/test_output/basic/c5f4176da2764378826652cb24b6cb05
-rw-r--r--   3 hbase hbase       9313 2021-11-26 09:59 /user/dasha/test_output/basic/de11b176c73a4aa29133c0064c1e5d52
```

2. Run the LoadIncrementalHFiles utility under the `hbase` user. Use the path to the previously created folder (`test_output`) and the table name (`people_ages`) as input parameters:

```
$ sudo -u hbase hbase org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles /user/dasha/test_output people_ages
```

3. Run HBase shell and get the number of rows in the `people_ages` table:

```
count "people_ages"
```

As you can see, the table contains only 997 rows, not 1000, as in the uploaded file. The reason for this is the presence of three duplicates in the file, which we discussed earlier.

```
997 row(s)
Took 0.3057 seconds
=> 997
```

CAUTION

Be careful with selection of the source file field for the table row keys. This field must contain unique values. Duplicates are not uploaded, even if the column family supports several value versions.

4. Scan the `people_ages` table to check its content:

```
scan 'people_ages'
```

The first ten rows are listed below:

```
ROW                             COLUMN+CELL
 Abbott Delia                   column=basic:age, timestamp=1637918350575, value=62
 Abbott Howard                  column=basic:age, timestamp=1637918350575, value=24
 Abbott Jack                    column=basic:age, timestamp=1637918350575, value=29
 Adams Clyde                    column=basic:age, timestamp=1637918350575, value=29
 Aguilar Myrtie                 column=basic:age, timestamp=1637918350575, value=23
 Aguilar Terry                  column=basic:age, timestamp=1637918350575, value=65
 Alexander Derrick              column=basic:age, timestamp=1637918350575, value=46
 Alexander Gregory              column=basic:age, timestamp=1637918350575, value=54
 Alexander Leon                 column=basic:age, timestamp=1637918350575, value=42
 Allen Austin                   column=basic:age, timestamp=1637918350575, value=34
```

5. Check distribution of the table data between five regions using the following command:

```
list_regions 'people_ages'
```

The output looks similar to this:

```
                                         SERVER_NAME |                                                    REGION_NAME |  START_KEY
|    END_KEY |  SIZE |   REQ |   LOCALITY |
 --------------------------------------------------- | -------------------------------------------------------------- | ----------
| ---------- | ----- | ----- | ---------- |
 bds-adh-1.ru-central1.internal,16020,1637912629824 |  people_ages,,1637920713477.bf509f13aa0dd59a5f4e9bcfc6cfbfab. |            |
F |     0 |   953 |        1.0 |
 bds-adh-3.ru-central1.internal,16020,1637912629738 | people_ages,F,1637920713477.b6693f3894a0feddba80ff15c06e2fbf. |          F |
K |     0 |   873 |        1.0 |
 bds-adh-3.ru-central1.internal,16020,1637912629738 | people_ages,K,1637920713477.50dcc6b90c412dab1f830a1b963d21e1. |          K |
P |     0 |   781 |        1.0 |
 bds-adh-2.ru-central1.internal,16020,1637912628845 | people_ages,P,1637920713477.80f4f1f471c9129f00d8f73701159145. |          P |
W |     0 |  1019 |        1.0 |
 bds-adh-2.ru-central1.internal,16020,1637912628845 | people_ages,W,1637920713477.ef6dfb19328842ce5c4ce883378329b0. |          W |
|     0 |   369 |        1.0 |
 5 rows
Took 0.2080 seconds
hbase(main):012:0> 'people_ages'
=> "people_ages"
```

6. To check the presence of HFiles in HBase, log out from HBase shell and run the following command:

```
$ hdfs dfs -ls /hbase/data/default/people_ages
```

The output contains five folders with HFiles and two service folders:

```
Found 7 items
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/.tabledesc
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/.tmp
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/50dcc6b90c412dab1f830a1b963d21e1
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/80f4f1f471c9129f00d8f73701159145
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/b6693f3894a0feddba80ff15c06e2fbf
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/bf509f13aa0dd59a5f4e9bcfc6cfbfab
drwxr-xr-x   - hbase hbase          0 2021-11-26 09:58 /hbase/data/default/people_ages/ef6dfb19328842ce5c4ce883378329b0
```

All HBase data in HDFS is stored in the */hbase/data/* folder and sorted by namespaces and then by table names.

Use snapshots for data backup and restore in HBase

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
- Usage
  - Create a snapshot
  - Restore data from a snapshot
  - Export a snapshot
  - Rotate snapshots

Overview

HBase snapshots is a feature that allows you to back up data with minimal possible impact on the HBase performance. Making of a table snapshot neither involves any data copying nor requires the table to be disabled. A snapshot export to another cluster is done without the Master or Region Servers participation by using the utility based on distcp. When required, the table can be quickly restored from the snapshot.

Snapshot is basically a collection of metadata: table schema and links to the HFiles containing the actual data of the table. HFiles, once created, cannot be changed in any way except deleted. At the moment of a snapshot creation, the HFiles containing the data of a table become linked to that snapshot. If an HFile linked to a snapshot undergoes compaction, it is not deleted but rather archived. HFiles linked to a snapshot exist all the time the snapshot exists, so that the information could be extracted and the corresponding table restored. Same principle applies to snapshot clones, which are tables restored from a snapshot but independent from the original table: while a clone exists, the corresponding HFile or HFiles exist as well. This is why a rotation policy for the snapshots and their clones must be in place in order to avoid the disk space depletion.

> **NOTE**
>
> Detailed information on the HBase shell commands used for working with snapshots is provided in the HBase shell commands → Snapshots commands section.

Usage

For the purposes of examples shown below, a small test table named `articles` will be used, the creation process of which is provided in the Quick start with HBase shell article. Complete Step 1 and Step 3 from it if you wish to follow the examples.

The examples below are consistent with the following life cycle of a snapshot:

1. Creation of a snapshot and verification of its existence.
2. Restoration of a table from the snapshot after it being changed.
3. Export of a snapshot to another HBase cluster and recreation of a table from it.
4. Removal of a snapshot.

All commands are executed in the HBase shell if not specified otherwise.

Create a snapshot

To create a table snapshot, use the snapshot command. Example:

```
snapshot 'articles', 'articles_snp'
```

Use the list_snapshots command if you want to see all the snapshots or the list_table_snapshots command if you want to see the snapshots of the specified table. Example:

```
list_table_snapshots 'articles'
```

The output should look the following way:

```
SNAPSHOT                                     TABLE + CREATION TIME
 articles_snp                                articles (2024-11-01 08:11:25 UTC)
1 row(s)
Took 0.0528 seconds
=> ["articles_snp"]
```

Restore data from a snapshot

After creating a snapshot, alter the data in the table in any way you want. For example, you can delete one row entirely and add another one. Example:

```
deleteall 'articles', 'article2'
put 'articles', 'article3', 'basic:author', 'King'
put 'articles', 'article3', 'basic:header', 'Test article3'
put 'articles', 'article3', 'tags:ref', true
```

Scan the table to make sure that the changes took effect. Example:

```
scan 'articles'
```

The output should look the following way:

```
ROW                                          COLUMN+CELL
 article1                                    column=basic:author, timestamp=2024-11-01T08:03:55.730, value=Test author
 article1                                    column=basic:header, timestamp=2024-11-01T08:03:55.758, value=Test article
 article1                                    column=tags:arch, timestamp=2024-11-01T08:03:55.776, value=true
 article1                                    column=tags:concepts, timestamp=2024-11-01T08:03:55.794, value=true
 article1                                    column=tags:tutorials, timestamp=2024-11-01T08:03:58.771, value=true
 article3                                    column=basic:author, timestamp=2024-11-01T08:19:36.078, value=King
 article3                                    column=basic:header, timestamp=2024-11-01T08:19:36.093, value=Test article3
 article3                                    column=tags:ref, timestamp=2024-11-01T08:19:37.856, value=true
2 row(s)
Took 0.0638 seconds
```

To restore the table, use the restore_snapshot command, but first you need to disable the table being restored. Example:

```
disable 'articles'
restore_snapshot 'articles_snp'
```

Enable the table back again and scan it to make sure that the restoration worked. Example:

```
enable 'articles'
scan 'articles'
```

The output should look the following way:

```
ROW                                          COLUMN+CELL
 article1                                    column=basic:author, timestamp=2024-11-01T08:03:55.730, value=Test author
 article1                                    column=basic:header, timestamp=2024-11-01T08:03:55.758, value=Test article
 article1                                    column=tags:arch, timestamp=2024-11-01T08:03:55.776, value=true
 article1                                    column=tags:concepts, timestamp=2024-11-01T08:03:55.794, value=true
 article1                                    column=tags:tutorials, timestamp=2024-11-01T08:03:58.771, value=true
 article2                                    column=basic:author, timestamp=2024-11-01T08:04:19.531, value=Test author2
 article2                                    column=basic:header, timestamp=2024-11-01T08:04:19.554, value=Test article2
 article2                                    column=tags:ref, timestamp=2024-11-01T08:04:20.802, value=true
2 row(s)
Took 0.0387 seconds
```

Export a snapshot

You can export the snapshot to another cluster and restore the table there. To do this, first connect to an HBase source cluster node via SSH and change user to `hbase`, then export the snapshot. The export command has the following syntax:

```
$ hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot <snapshot_name> -copy-to hdfs://<namenode_active>:<port>
<hbase_dir> -mappers <map_number>
```

where:

- `<snapshot_name>` — name of the snapshot being exported;
- `<namenode_active>` — network address of the active NameNode server in the destination cluster (use the `hdfs haadmin -getAllServiceState` command to find out which server is active);
- `<port>` — port number of the NameNode server (`8020` by default);
- `<hbase_dir>` — HBase home directory: value of the `zookeeper.znode.parent` parameter in the *hbase-site.xml* configuration file of the destination cluster (`/hbase` by default);
- `<map_number>` — number of mapper jobs to perform the export.

Example:

```
$ sudo -u hbase bash
$ hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot articles_snp -copy-to hdfs://av-adh-backup-1.ru-central1.
internal:8020/hbase -mappers 4
```

Log in to the HBase shell of the destination cluster and restore the table from the snapshot:

```
restore_snapshot 'articles_snp'
```

> **NOTE**
>
> In this case the restored table is already enabled.

Rotate snapshots

It is important that old and unneeded snapshots do not pile up and are deleted in a timely manner. HFiles linked by snapshots (or their clones) cannot be erased as long as those snapshots exist, even if the HFiles are archived. When you no longer need a snapshot, delete it using the delete_snapshot command. Example:

```
delete_snapshot 'articles_snp'
```

You can also delete all snapshots of a certain table using the delete_table_snapshots command or delete all snapshots with the names matching a certain regular expression using the delete_all_snapshot command.

HBase replication

Vladimir Adamenkov

Contents

To Table of Contents

Overview

Replication setup

- From one cluster to another
- From one cluster to several others

Overview

Replication in HBase is a mechanism that allows you to copy the contents of a table (or its separate column families) to another ADH cluster. This may serve the purposes of backing up and recovering data, geographic data distribution, data aggregation, and others.

The cluster that contains the original data is the source cluster, and the cluster that receives the data is the destination cluster. Any cluster can be source, destination, or both. A source cluster can send data to any number of destination clusters. The replication process is incremental: as soon as any changes are made in the source data, they are transmitted to the destination. If the destination cluster is unavailable for some reason, the changes made since the last successful transaction are stored on the source side and await the destination cluster to be available back again. For the replication to work, the names of the source and destination tables and their respective column families must be the same.

> NOTE
>
> Detailed information on the HBase shell commands used for the replication process setup is provided in the HBase shell commands → Replication commands section.

Replication setup

For the replication of a table you can have any number of clusters for the one to be the source cluster and the others to be destination clusters. When your clusters are configured and running, do the following:

1. Go to the ADCM UI and select your source ADH cluster.
2. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***Custom hbase-site.xml*** section and click ***Add property***.
4. Enter `hbase.replication` for the property name and set its value to `true`.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Now you can set up the replication in various scenarios.

From one cluster to another

Test table used to illustrate the replication setup process is obtained by importing the people.csv file. Import procedure is provided in the Bulk loading via built-in MapReduce jobs article.

To set up the table replication from the source cluster to the destination cluster, do the following:

1. Log in to the HBase shells on both clusters.
2. On the destination cluster, create the table with the same structure and properties as the one being replicated:

```
create 'people', {NAME => 'basic', VERSIONS => 5}, {SPLITS => ['F', 'K', 'P', 'W']}, {NAME => 'location', VERSIONS => 5}
```

   You can use the `describe` command in the source cluster HBase shell to obtain the properties of the table.

3. On the source cluster, create a peer record using the add_peer command:

```
add_peer '1', CLUSTER_KEY => 'av-adh-backup-1.ru-central1.internal,av-adh-backup-2.ru-central1.internal,av-adh-backup-3.ru-
central1.internal:2181:/hbase'
```

4. Disable the table using the disable command:

```
disable 'people'
```

5. Set the column families up for replication using the alter command:

```
alter 'people', {NAME => 'basic', REPLICATION_SCOPE => '1'}, {NAME => 'location', REPLICATION_SCOPE => '1'}
```

6. Enable the table back again using the enable command:

```
enable 'people'
```

This should be enough for the replication process to start. You can make some changes in the table (e.g. add a value) and then check the replication process status using the status command:

```
status 'replication'
```

The output will look like this:

```
version 2.5.10
3 live servers
    av-adh-1.ru-central1.internal:
       SOURCE: PeerID=1
         Normal Queue: 1
           No Ops shipped since last restart, SizeOfLogQueue=1, EditsReadFromLogQueue=0, OpsShippedToTarget=0, No edits for this
source since it started, Replication Lag=0
      SINK: TimeStampStarted=1729231941314, Waiting for OPs...
    av-adh-2.ru-central1.internal:
       SOURCE: PeerID=1
         Normal Queue: 1
           No Ops shipped since last restart, SizeOfLogQueue=1, EditsReadFromLogQueue=0, OpsShippedToTarget=0, No edits for this
source since it started, Replication Lag=0
      SINK: TimeStampStarted=1729231977710, Waiting for OPs...
    av-adh-3.ru-central1.internal:
       SOURCE: PeerID=1
         Normal Queue: 1
           AgeOfLastShippedOp=799, TimeStampOfLastShippedOp=Fri Oct 18 12:27:53 UTC 2024, SizeOfLogQueue=1,
EditsReadFromLogQueue=1, OpsShippedToTarget=1, TimeStampOfNextToReplicate=Fri Oct 18 12:27:53 UTC 2024, Replication Lag=0
```

The `OpsShippedToTarget` property indicates how many change operations have been transmitted to the destination cluster table. Nonzero value on at least one server means the replication is happening.

If the source cluster table was non-empty before the replication setup, you may need to copy its contents to the destination cluster. To do so, connect to an HBase source cluster node via SSH and execute the commands of the following kind:

```
$ cd /bin
./hbase org.apache.hadoop.hbase.mapreduce.CopyTable --peer.adr=av-adh-backup-1.ru-central1.internal,av-adh-backup-2.ru-central1.
internal,av-adh-backup-3.ru-central1.internal:2181:/hbase people
```

where:

- `--peer.adr` — the cluster key you previously used composing the `add_peer` command;
- `people` — the name of the table being copied.

From one cluster to several others

To set up the replication to several destination clusters, do the same as for one cluster, but when adding peers, make a separate peer record for each destination cluster, paying attention to different IDs.

Example:

```
add_peer '1', CLUSTER_KEY => 'av-adh-backup1-1.ru-central1.internal,av-adh-backup1-2.ru-central1.internal,av-adh-backup1-3.ru-
central1.internal:2181:/hbase'
add_peer '2', CLUSTER_KEY => 'av-adh-backup2-1.ru-central1.internal,av-adh-backup2-2.ru-central1.internal,av-adh-backup2-3.ru-
central1.internal:2181:/hbase'
add_peer '3', CLUSTER_KEY => 'av-adh-backup3-1.ru-central1.internal,av-adh-backup3-2.ru-central1.internal,av-adh-backup3-3.ru-
central1.internal:2181:/hbase'
```

Data backup and restore in HBase

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
  - Key concepts
  - Architectural deployment strategies
- Configuration
- Limitations
- Basic commands
- Examples
  - HBase
  - Phoenix

Overview

The backup functionality for HBase has been introduced in ADH version 4.1.0 (HBase 2.6).

This functionality is designed to create "cold" copies of data, which is a critical element of the business continuity and disaster recovery strategy. Unlike replication, which creates "hot" copies, backups are stored in external storage and require manual recovery procedures, providing protection against catastrophic failures.

Key concepts

The backup functionality in HBase uses the following terms and concepts:

- **Full backup** — contains the full state of the table at the time of creation. It is the basis for subsequent incremental backups.
- **Incremental backup** — contains only changes from WAL relative to the last full or incremental backup. It is effective in terms of volume and allows you to restore data at a specific point in time.
- **Backup set** — a named group of tables to simplify the management of backup or recovery operations. It is recommended to save backups for each set in a dedicated path.
- **Backup ID** — a unique identifier (based on Unix epoch time) assigned to each backup session.
- **Increment merge** — the ability to combine several successful backups into one.

Architectural deployment strategies

The following architectural deployment strategies can be used for backing up data in HBase:

- **Inside the cluster** — backups are stored on the same HDFS cluster. It is only suitable for testing and does not provide fault tolerance.
- **Dedicated cluster** — backups are copied to a separate HDFS cluster, often with a cheaper hardware configuration. Recommended for production environments, it provides geographical separation.
- **Cloud Storage** — backups are saved to the cloud (for example, S3) or an HDFS-compatible storage system (via S3A, WebHDFS). The most fault-tolerant option.

Configuration

To configre the backup feature in HBase, do the following:

1. Go to the ADCM UI and select your ADH cluster.
2. Navigate to ***Services*** → ***HBase*** → ***Primary configuration*** and toggle ***Show advanced***.
3. Open the ***hbase-site.xml*** section and set the ***hbase.backup.enable*** parameter to `true`.
4. Review the following parameters and make sure they have the corresponding values:
   - ***hbase.master.logcleaner.plugins*** — `org.apache.hadoop.hbase.backup.master.BackupLogCleaner`. You can also add other backup procedure manager classes to this parameter value, separating them by commas. You can do the same for all the following parameters in this list.
   - ***hbase.procedure.master.classes*** — `org.apache.hadoop.hbase.backup.master.LogRollMasterProcedureManager`.
   - ***hbase.procedure.regionserver.classes*** — `org.apache.hadoop.hbase.backup.regionserver.LogRollRegionServerProcedureManager`.
   - ***hbase.coprocessor.region.classes*** — `org.apache.hadoop.hbase.backup.BackupObserver`.
   - ***hbase.coprocessor.master.classes*** — `org.apache.hadoop.hbase.backup.BackupMasterObserver`.
   - ***hbase.master.hfilecleaner.plugins*** — `org.apache.hadoop.hbase.backup.BackupHFileCleaner`.
5. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Limitations

The backup functionality in HBase has the following limitations:

- **Sequential execution**. You cannot run multiple backup or restore operations at the same time.
- **No cancellation**. Running backup or recovery operations cannot be canceled.
- **Superuser rights**. All operations are performed on behalf of the HBase superuser.
- **Restoring to online cluster only**. The target cluster for recovery must be up and running.
- **One destination**. Backup can only be saved to one location. Copying requires manual actions.
- **WAL volume growth**. Incremental backups prevent the deletion of WAL files before the next backup, which can lead to an increase in the amount of data in HDFS. It is necessary to closely monitor the schedule and storage.
- **Transparent data encryption (TDE)**. The functionality has not been tested on clusters with data encryption enabled.
- **Restoring in another cluster**. When you copy a backup to another cluster, it will be possible to restore, but after that you will need a full backup to create a history.

Basic commands

HBase backup functionality exploits the following basic commands:

1. Create a set of tables for granular backup configuration:

```
$ hbase backup set add <set_name1> <table1>,<table2>
$ hbase backup set add <set_name2> <table3>
```

2. Create full and incremental backups for different sets:

```
$ hbase backup create full hdfs://<nn>:8020/tmp/backups-path1/ -w 3 -s <set_name1> -d
$ hbase backup create full hdfs://<nn>:8020/tmp/backups-path2/ -w 3 -s <set_name2> -d
$ hbase backup create incremental hdfs://<nn>:8020/tmp/backups-path1/ -w 3 -s <set_name1> -d
$ hbase backup create incremental hdfs://<nn>:8020/tmp/backups-path2/ -w 3 -s <set_name2> -d
```

3. View backup history:

```
$ hbase backup history
```

4. Merge backups:

```
$ hbase backup merge <backup_increment_id1>,<backup_increment_id2>,<backup_increment_id3>
```

   IDs can be found in the history or in the root directory of backups on the filesystem.

5. Restore:

```
$ hbase restore hdfs://<nn>:8020/tmp/backups-path2/ <backup_id> -o -s <set_name2> -d
```

Examples

HBase

This example demonstrates the backup functionality with pure HBase, without Phoenix.

1. Create the original table:

```
$ hbase shell
```

```
create 'user_activity', 'cf1', 'cf2'
put 'user_activity', 'user1', 'cf1:name', 'Alice'
put 'user_activity', 'user1', 'cf1:email', 'alice@email.com'
put 'user_activity', 'user1', 'cf2:last_login', '2024-01-15'
put 'user_activity', 'user1', 'cf2:login_count', '5'
put 'user_activity', 'user2', 'cf1:name', 'Bob'
put 'user_activity', 'user2', 'cf1:email', 'bob@email.com'
put 'user_activity', 'user2', 'cf2:last_login', '2024-01-14'
put 'user_activity', 'user2', 'cf2:login_count', '3'
put 'user_activity', 'user3', 'cf1:name', 'Charlie'
put 'user_activity', 'user3', 'cf1:email', 'charlie@email.com'
put 'user_activity', 'user3', 'cf2:last_login', '2024-01-13'
put 'user_activity', 'user3', 'cf2:login_count', '7'
```

2. Exit `hbase` shell and create a set of backups:

```
exit
```

```
$ hbase backup set add user_backup_set user_activity
```

3. Create a full backup:

```
$ hbase backup create full hdfs://tmp/hbase-backup -s user_backup_set -w 3
```

4. Add new data:

```
$ hbase shell
```

```
put 'user_activity', 'user4', 'cf1:name', 'Diana'
put 'user_activity', 'user4', 'cf1:email', 'diana@email.com'
put 'user_activity', 'user4', 'cf2:last_login', '2024-01-16'
put 'user_activity', 'user4', 'login_count', '2'
put 'user_activity', 'user1', 'cf2:login_count', '6'
put 'user_activity', 'user2', 'cf2:last_login', '2024-01-16'
```

5. Exit `hbase` shell and make an incremental backup:

```
exit
```

```
$ hbase backup create incremental hdfs://tmp/hbase-backup -s user_backup_set -w 3
```

6. Add more data and create a second incremental backup:

```
$ hbase shell
```

```
put 'user_activity', 'user5', 'cf1:name', 'Eve'
put 'user_activity', 'user5', 'cf1:email', 'eve@email.com'
deleteall 'user_activity', 'user3'
put 'user_activity', 'user1', 'cf2:login_count', '7'
exit
```

```
$ hbase backup create incremental hdfs://tmp/hbase-backup -s user_backup_set -w 3
```

7. Check history and description:

```
$ hbase backup history -s user_backup_set
$ hbase backup describe <backup_id>
```

8. Simulate data loss:

```
$ hbase shell
```

```
disable 'user_activity'
drop 'user_activity'
```

9. Exit `hbase` shell and restore the data:

```
exit
```

```
$ hbase restore hdfs://tmp/hbase-backup <backup_id> -s user_backup_set
```

   You can restore the data from any point in time when one of the previous backups was made.

Phoenix

This example demonstrates the backup functionality with Phoenix.

1. Run the */usr/lib/phoenix/bin/sqlline.py* script and create the original table:

```
CREATE TABLE USER_ACTIVITY (
    USER_ID VARCHAR PRIMARY KEY,
    NAME VARCHAR,
    EMAIL VARCHAR,
    LAST_LOGIN DATE,
    LOGIN_COUNT INTEGER
) COMPRESSION='SNAPPY', SALT_BUCKETS=4;
```

2. Load values into the table:

```
UPSERT INTO USER_ACTIVITY VALUES ('user1', 'Alice', 'alice@email.com', TO_DATE('2024-01-15'), 5);
UPSERT INTO USER_ACTIVITY VALUES ('user2', 'Bob', 'bob@email.com', TO_DATE('2024-01-14'), 3);
UPSERT INTO USER_ACTIVITY VALUES ('user3', 'Charlie', 'charlie@email.com', TO_DATE('2024-01-13'), 7);
```

3. Press `Ctrl+D` and create a set of backups. Additional system tables are required for Phoenix tables. You may need more of them if, for example, indexes are used. For this reason, it is better to manage backups for Phoenix tables in one set:

```
$ hbase backup set add phoenix_backup_set USER_ACTIVITY,SYSTEM.CATALOG,SYSTEM.SEQUENCE,SYSTEM.STATS
```

4. Create a full backup:

```
$ hbase backup create full hdfs://tmp/hbase-backup -s phoenix_backup_set -w 3
```

5. Run the */usr/lib/phoenix/bin/sqlline.py* script again and add new data:

```
UPSERT INTO USER_ACTIVITY VALUES ('user4', 'Alice', 'alice@email.com', TO_DATE('2024-01-15'), 5);
UPSERT INTO USER_ACTIVITY VALUES ('user5', 'Bob', 'bob@email.com', TO_DATE('2024-01-14'), 3);
UPSERT INTO USER_ACTIVITY VALUES ('user6', 'Charlie', 'charlie@email.com', TO_DATE('2024-01-13'), 7);
```

6. Press `Ctrl+D` and make an incremental backup:

```
$ hbase backup create incremental hdfs://tmp/hbase-backup -s phoenix_backup_set -w 3
```

7. Run the */usr/lib/phoenix/bin/sqlline.py* script again and add more new data:

```
UPSERT INTO USER_ACTIVITY VALUES ('user7', 'Alice', 'alice@email.com', TO_DATE('2024-01-15'), 5);
UPSERT INTO USER_ACTIVITY VALUES ('user8', 'Bob', 'bob@email.com', TO_DATE('2024-01-14'), 3);
UPSERT INTO USER_ACTIVITY VALUES ('user9', 'Charlie', 'charlie@email.com', TO_DATE('2024-01-13'), 7);
```

8. Press `Ctrl+D` and make a second incremental backup:

```
$ hbase backup create incremental hdfs://tmp/hbase-backup -s phoenix_backup_set -w 3
```

9. Check history:

```
$ hbase backup history -s user_backup_set
$ hbase backup describe <backup_id>
```

10. Run the */usr/lib/phoenix/bin/sqlline.py* script again and simulate data loss:

```
DROP TABLE USER_ACTIVITY;
```

11. Press `Ctrl+D` and recover the data:

```
$ hbase restore hdfs://tmp/hbase-backup <backup_id> -s phoenix_backup_set
```

   You can restore the data from any point in time when one of the previous backups was made.

HBase performance tuning

Vladimir Adamenkov

Contents

To Table of Contents

- HDFS settings
- HBase settings
- Other
  - Table schema
  - Rowkey design
  - Major compaction equalization
  - Compression
  - Bloom filters
  - Bulk load

This article contains some advice on how to improve the performance of your HBase cluster. Some of the approaches cannot be defined exactly for all cases and need to be fine-tailored to each certain configuration.

HDFS settings

HDFS parameters

Parameter	Default and description	Recommendation
dfs.datanode.max.transfer.threads	Default: `4096`. This parameter defines the upper limit of files that an HDFS DataNode can service simultaneously	Set the value to `8192`. This parameter has to be added manually in the ***Custom hdfs-site.xml*** section of the HDFS service configuration page

HBase settings

RegionServer settings

Parameter	Default and description	Recommendation
zookeeper.session.timeout	Default: `90000` (in milliseconds). This parameter defines the timeout before the Master server considers a RegionServer inactive. The default value is irrelevant if the ZooKeeper service retains its default value for the tick time (2 seconds). The ZooKeeper service has its own timeouts for inactive services: minimum 2 tick times and maximum 20 tick times. By default, this means that the ZooKeeper service will wait no longer than 40 seconds before removing the inactive RegionServer ethereal znode while the Master server would be still waiting	Lower the HBase timeout parameter or increase the ZooKeeper tick time parameter — whichever is lower takes effect. Mind the following: - Do not lower the `zookeeper.session.timeout` parameter too much: there will always be lags and pauses due to network traffic, garbage collection, or other temporary problems. Your RegionServers should withstand those without exceeding the timeout. - Do not increase the ZooKeeper tick time higher than 4 or 5 seconds: this can negatively affect the ZooKeeper quorum health. The `zookeeper.session.timeout` parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page, the tick time is defined by the `tickTime` parameter of the ***zoo.cfg*** section in the ZooKeeper service configuration page. Both parameters are specified in milliseconds
hbase.regionserver.handler.count	Default: `30`. This parameter defines the number of open threads for responses to the user tables incoming queries	Each thread consumes CPU time and RAM, so you need to assess the average query payload size before tweaking this parameter. If the queries carry high payload, then this parameter is better kept at a low value. But if the queries are mostly light, then increasing this parameter value might be a good idea: begin with the number of the CPU cores doubled, and then increase it little by little until you reach the peak performance. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.master.wait.on.regionservers.mintostart	Default: `1`. This parameter defines the minimum number of RegionServers that have to be registered with the Master server before it begins to assign regions to them. The default value of `1` means that the Master server will begin assigning regions to the very first RegionServer that registers with it	If your cluster has many regions (e.g. several thousands), it is recommended to increase this parameter value so that the assignment of the regions would be delayed and hence more evenly distributed. Otherwise, the first registered RegionServer may constitute a bottleneck and slow down the cluster startup and even lead to the loss of access to itself. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

Heap memory settings

Parameter	Default and description	Recommendation
HBASE_MASTER_OPTS	Default: `-Xms700m -Xmx9G`. This parameter defines the amount of RAM allocated to the Master server	The default value is enough for most cases, as Master server does not require a lot of RAM to work. Nevertheless, if your cluster is really big and has an enormous number of tables and/or regions, you may need to increase the upper memory limit, since the Master server retains the regions statuses in its memory. This parameter is located in the ***hbase-env.sh*** section of the HBase service configuration page
HBASE_REGIONSERVER_OPTS	Default: `-Xms700m -Xmx9G`. This parameter defines the size of the RegionServer heap memory, which contains the MemStores and the block cache	Generally, the more this parameter value is the better. Use the following considerations: - RegionServer heap memory should be proportional to the number of regions on the node. - If memory consumption peaks at 60 or 70 percent — increase the heap. - If the JVMs use the CMS garbage collection algorithm, which is provided by default in Arenadata HBase distribution, the RegionServer heap memory must not be bigger than 36 GB. Otherwise, the garbage collector stop-the-world pauses can render the HBase unusable and heavily complicate the data preservation. If the JVMs use the G1 (garbage first) garbage collection algorithm, then there is no limit to the RegionServer heap memory size. This parameter is located in the ***hbase-regionserver-env.sh*** section of the HBase service configuration page

Block cache settings

Parameter	Default and description	Recommendation
hfile.block.cache.size	Default: `0.4`. This parameter defines the share of the block cache in the RegionServer heap memory	The default value is good for a situation when the table data is often heavily read and it is more or less the same data all the time. If the situation is not exactly that (e.g. the requested data is different all the time, or the data requisition is rather light in an average request), you may want to reduce this parameter value down to `0.2` or even `0.1`. The general rule is to keep this value no bigger than it really needs to be. Exact value should be picked based on performance, by trial and error. It also must be balanced with the `hbase.regionserver.global.memstore.size` parameter value. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

MemStore and flush settings

Parameter	Default and description	Recommendation
hbase.regionserver.global.memstore.size	Default: `0.4`. This parameter defines the share of the MemStore in the RegionServer heap memory	Specify a value so that it combined with the `hfile.block.cache.size` parameter value would add up to `0.7` or `0.75`. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.regionserver.global.memstore.size.lower.limit	Default: `0.95`. This parameter defines a threshold percentage in MemStores of a region. When the MemStores of a region are filled with new data up to this threshold, updates are blocked and flushes are forced	Leave the default value as is
hbase.hregion.memstore.flush.size	Default: `134217728` (128 MB). This parameter defines the size of data in a MemStore when it is flushed and an HFile is created from its contents. When a MemStore is flushed, all the other MemStores in the same region are flushed too. The less this value is, the more often flushes occur, the bigger is the I/O overhead, the larger is the number of HFiles, and the more often the comaction processes are run	It is recommended to double this parameter value (up to `268435456`, meaning 256 MB). If your HBase has a heavy write load, you may want to increase the value up to `536870912` (512 MB), but it is not always possible, especially if you have a rather small RegionServer heap memory size and a very big number of regions and column families. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page. Possible values are `134217728`, `268435456`, and `536870912`
hbase.hstore.flusher.count	Default: `2`. This parameter defines the number of threads for the flush operations. The more this value is, the more effective the flush process is, but at the same time the more the load on the HDFS is, and potentially the bigger is the number of compactions	You can start with increasing this parameter value up to `4` and then try to increase it more and see if that affects the performance. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hregion.memstore.block.multiplier	Default: `4`. This parameter defines the multiplier of possible MemStore size (defined by the `hbase.hregion.memstore.flush.size` parameter) extension when the update traffic is very high	It is recommended to increase this value up to `8`, if the RAM size allows it. When a MemStore size reaches its maximum defined by the multiplication of this parameter value and the `hbase.hregion.memstore.flush.size` parameter value, the corresponding column family is blocked for all write operations until the MemStore is flushed. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

Region split settings

Parameter	Default and description	Recommendation
hbase.regionserver.regionSplitLimit	Default: `1000`. This parameter defines the maximum number of regions on a RegionServer after which region splitting can no longer occur. This is not a limit on the number of the regions itself, but rather a stopping point in creating more regions	The default value of `1000` effectively means no limits, since predominantly there are much less regions per region server in HBase clusters. You can calculate this limit based on your particular situation. For example, your cluster has the following settings: - `HBASE_REGIONSERVER_OPTS` — `-Xms700m -Xmx32G`; - `hbase.regionserver.global.memstore.size` — `0.55`; - `hbase.hregion.memstore.flush.size` — `268435456` (256 MB); - 2 column families in the table. To find the number of regions, you need to multiply the region server heap memory size by the MemStore share quotient in it (the first two parameters), and then consecutively divide the result by the `hbase.hregion.memstore.flush.size` parameter value and the number of column families. In this case, we have the following formula: $\frac{32\mathrm{GB} \cdot 0.55}{256\mathrm{MB} \cdot 2} = 35.2$ Round the result up if it is fractioned and you will have your optimal number of regions per RegionServer. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hregion.max.filesize	Default: `10737418240` (10 GB). This parameter defines the maximum total size of HFiles in a region before it is split	The default value is generally good. You can increase this value based on performance (for example, up to 20 GB), but not too much: much higher values can negatively affect the compaction process. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page

Compaction settings

Parameter	Default and description	Recommendation
hbase.hstore.compactionThreshold	Default: `3`. This parameter defines the number of HFiles in a region before they undergo a minor compaction	It is recommended to up this value to `5`. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hstore.blockingStoreFiles	Default: `16`. This parameter defines the maximum number of HFiles in a region before the updates to this region are blocked. The block remains until a minor compaction is forcefully run or the timeout specified by the `hbase.hstore.blockingWaitTime` parameter expires	If your cluster has a heavy write load and flushes are delayed due to too many HFiles, increase this parameter value. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hstore.blockingWaitTime	Default: `90000` (in milliseconds). This parameter defines the timeout for the updates block of a region due to too many HFiles (see the previous parameter). When this time passes, the block is lifted regardless of whether the minor compaction has completed	Adjust this parameter according to the requirements of your cluster. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hstore.compaction.min	Default: `3`. This parameter defines the minimum number of HFiles designated for compaction before such compaction can be executed	It is recommended to increase this value to `5`. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hstore.compaction.max	Default: `10`. This parameter defines the maximum number of HFiles that can participate in a single compaction operation	The default value is good for most cases. If your cluster has a heavy write load, increase this value. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.regionserver.thread.compaction.small	Default: `1`. This parameter defines the number of threads used for the minor compaction process	If your cluster has a heavy write load and minor compactions are executed too often and take too much time, increase this value. However, it should not exceed half the number of disks available for the HBase. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.regionserver.thread.compaction.large	Default: `1`. This parameter defines the number of threads used for the major compaction process	If your cluster has a heavy write load, you can increase this value, but making it higher than the `hbase.regionserver.thread.compaction.small` parameter value is not recommended. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.hstore.compaction.min.size	Default: `134217728` (128 MB). This parameter defines the maximum HFile size for it to be automatically designated for compaction	If your cluster has a heavy write load, you may have many small HFiles that, when compacted, form an HFile with a size still under this value (especially in case of a low `hbase.hstore.compaction.max` parameter value) and require a compaction again. Lowering this parameter value may address this problem: specify a value that is a little lower than the multiplication of an average MemStore flush size and the `hbase.hstore.compaction.max` parameter value. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.hstore.compaction.ratio	Default: `1.2F`. This parameter defines a quotient for the previous parameter (`hbase.hstore.compaction.min.size`). Values of these two parameters multiplied give the upper limit of the HFile size for it to be designated for compaction	The higher this value is, the more often the compaction occurs, but less HFiles need to be read, and vice versa. Lower value is preferrable for clusters with heavy write load, and higher value is preferrable for clusters with heavy read load. Usually the default value is good; if you choose to adjust it — it is recommended to stay within the `1.0F` — `1.4F` range. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page

WAL settings

Parameter	Default and description	Recommendation
hbase.regionserver.logroll.multiplier	Default: `0.95`. This parameter defines the size of a WAL file relative to the HDFS block size (which is specified by the `dfs.blocksize` parameter in the ***hdfs-site.xml*** section of the HDFS service configuration page). For example, if the HDFS block size is 134217728 (128 MB), then the WAL file size will be 127506842 (approximately 121.6 MB)	It is recommended to leave this parameter as is in all cases. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

hbase.regionserver.maxlogs	Default: `32`.
	This parameter defines the maximum number of WAL files in a region server
		You need to calculate the value for this parameter yourself by doing the following:
		1. Take the upper limit from the value of the `HBASE_REGIONSERVER_OPTS` parameter and multiply it by the value of the `hbase.regionserver.global.memstore.size` parameter. The result is the MemStore size.
		2. Calculate the size of the WAL file as described above for the `hbase.regionserver.logroll.multiplier` parameter.
		3. Divide the MemStore size by the size of the WAL file. Round the result up if necessary.
		If your cluster has a heavy write load, when the number of WAL files reaches its limit, region servers might require to flush all MemStores until all WAL files are archived and there is enough space for writing operations. This is called flush storm, and can be recognized by the following line in the log:
		`INFO[regionserver//10.22.100.5:16020.`
		`logRoller] wal.FSHLog:Too many wals: logs=35,`
		`maxlogs=32; forcing flush of 20 regions(s):`
		`d4kjnfnkf34335666d03cb1f`
		If this happens, increase the parameter value.
		This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

MSLAB settings

Parameter	Default and description	Recommendation
hbase.hregion.memstore.mslab.chunksize	Default: `2097152` (2 MB). This parameter defines the size of a chunk in the MemStore Allocation Buffer	Double or quadruple this parameter value if you use large-sized records. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

Client settings

Parameter	Default and description	Recommendation
hbase.client.write.buffer	Default: `2097152` (2 MB). This parameter defines the size of the write buffer. The bigger this value is, the more memory is used on both server and client sides, but the less the number of RPCs is	It is recommended to set this value to `8388608` (8 MB). To estimate the size of the server memory used, multiply this parameter value by the value of the `hbase.regionserver.handler.count` parameter. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.client.pause	Default: `100`. This parameter defines the duration of the client pause. It is used as a timeout before retrying a failed operation	It is recommended to significantly reduce this value (e.g. down to `3` or `5`) in order to avoid data loss. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.client.retries.number	Default: `15`. This parameter defines the maximum number of retries of a failed operation	It is recommended to reduce this value to `3` or `5` in order to avoid data loss. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.rpc.timeout	Default: `300000` (in milliseconds; 5 minutes). This parameter defines the timeout for all RPCs that the client sends to the HBase	If your client requests tend to take heavy toll on RAM, increase this parameter value. Avoid increasing it too much, because the client might wait for too long and then fail anyway. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.client.scanner.timeout.period	Default: `60000` (in milliseconds; 1 minute). This parameter defines the timeout for the RPCs related to scanning operations	This parameter value becomes useless if it is set higher than the `hbase.rpc.timeout` parameter value, because the operation is aborted after whichever timeout expires first. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
hbase.client.scanner.caching	Default: `2147483647`. This parameter defines the maximum number of rows that the client scanner can fetch from the HBase in one sitting	If the value is too high, the scanner may become overloaded by fetching and transferring too much data to the client, or even exceed the timeout, failing the job; also this will lead to using too much heap memory even in case of success. A value too low will lead to ineffective use of cluster resources, causing idling. This parameter value must be chosen carefully according to the characteristics of disks, RAM, CPUs, and the average read load. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.client.scanner.max.result.size	Default: `2097152` (2 MB). This parameter defines the maximum number of bytes returned when calling a next scanner method. If a single row size is larger than this value, it is still returned in its entirety	The default value is good for 1G Ethernet networks. If the network capacity in your cluster is better, increase this value, but mind the following: - Review the value specified in the `HBASE_MASTER_OPTS` parameter. It is recommended to set the value of `hbase.client.scanner.max.result.size` at one-eighth of the lower limit. - The set value multiplied by the value of `hbase.regionserver.handler.count` parameter must not exceed the upper limit. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

Other

Table schema

It is recommended to use no more than two column families per table.

Rowkey design

It is recommended to use salting/hashing techniques for even row distribution in regions.

Major compaction equalization

For more smooth performance, consider scheduling major comactions for different tables and column families using `cron` or similar utilities.

Compression

In some cases, using compression can boost performance by downsizing the HFiles and hence lowering the number of I/O operations.

Bloom filters

It is recommeded to use the Bloom filters for boosting the search operations in HFiles. The Bloom filters allow detecting whether an HFile contains a certain key by reading only a small amount of data of that HFile. If it does not, then the next HFile is searched.

Bulk load

Use bulk load whenever possible, as it is the fastest way to upload data to the HBase. If bulk load is not possible, consider uniting singular requests (like `GET`, `PUT`, or `DELETE`) in lists, as they are quite costly when executed one by one.

Logging in HBase

Daria Barysheva

Contents

To Table of Contents

[Using a file system]
[Using Web UI]

Both Master and Region Servers in HBase write text logs, which are useful while analyzing the causes of different errors. These logs are located in the */var/log/hbase* directory on each of the hosts. There are two ways to find them.

Using a file system

The server file system enables you to get direct access to logs using the following operations:

1. Connect to the host through SSH and run the following command:

```
$ ls -la /var/log/hbase
```

   The output looks similar to this:

```
total 10004
drwxr-xr-x.  2 hbase hbase     4096 Nov 29 06:17 .
drwxr-xr-x. 16 root  root      4096 Nov 29 05:35 ..
-rw-r--r--.  1 hbase hbase 10209158 Nov 29 11:34 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.log
-rw-r--r--.  1 hbase hbase      692 Nov 29 06:17 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out
-rw-r--r--.  1 hbase hbase      692 Nov 29 05:35 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.1
-rw-r--r--.  1 hbase hbase      692 Nov 26 07:43 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.2
-rw-r--r--.  1 hbase hbase      692 Nov 26 05:58 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.3
-rw-r--r--.  1 hbase hbase      692 Nov 26 05:26 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.4
-rw-r--r--.  1 hbase hbase      692 Nov 25 15:00 hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.5
```

NOTE

Logs are located in the local file system of the host, not in the HDFS.

2. View the content of the desired file:

```
$ tail /var/log/hbase/hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.log
```

   The output looks similar to this:

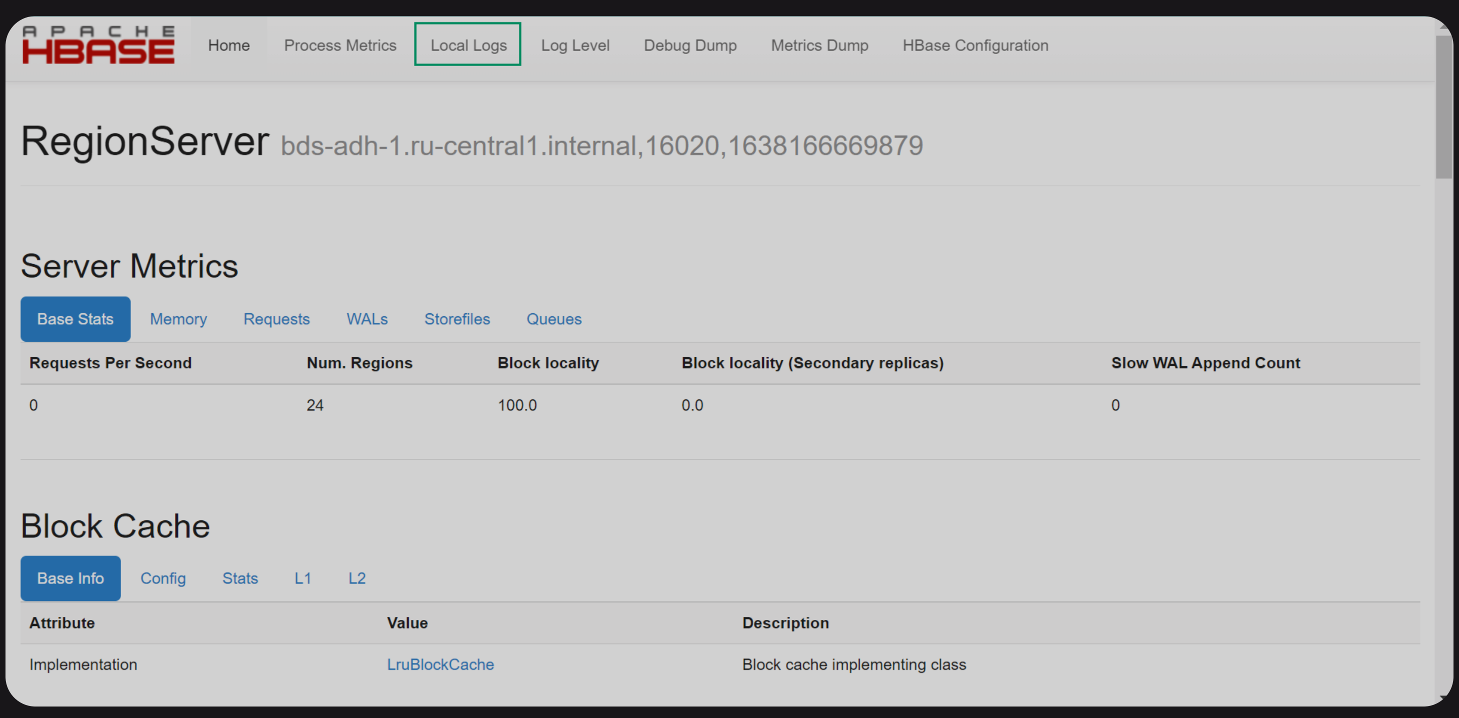
```
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-pig.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-thin-client.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
0    [RS-EventLoopGroup-1-9] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:48266, version=2.0.2,
sasl=false, ugi=hbase (auth:SIMPLE), service=ClientService
10629 [RS-EventLoopGroup-1-10] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:48328, version=2.0.
2, sasl=false, ugi=hbase (auth:SIMPLE), service=AdminService
10263041 [RS-EventLoopGroup-1-5] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:35086, version=2.
0.2, sasl=false, ugi=hbase (auth:SIMPLE), service=ClientService
10570940 [RS-EventLoopGroup-1-9] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:36368, version=2.
0.2, sasl=false, ugi=hbase (auth:SIMPLE), service=ClientService
18488201 [RS-EventLoopGroup-1-16] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:41578,
version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE), service=AdminService
18991216 [RS-EventLoopGroup-1-12] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:43712,
version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE), service=AdminService
```

Using Web UI

The following steps demonstrate how to view logs in HBase Web UI:

1. Open Web UI home page for the desired Master or Region Server.

2. Select the ***Local Logs*** menu item.



RegionServer bds-adh-1.ru-central1.internal,16020,1638166669879

Server Metrics

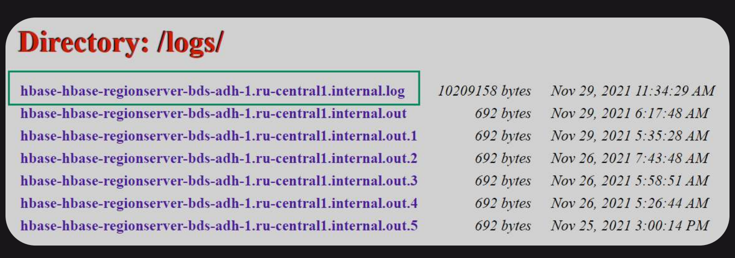
Requests Per Second	Num. Regions	Block locality	Block locality (Secondary replicas)	Slow WAL Append Count
0	24	100.0	0.0	0

Block Cache

Attribute	Value	Description
Implementation	LruBlockCache	Block cache implementing class

Switching to the list of logs

3. Click the desired file name in the opened list.

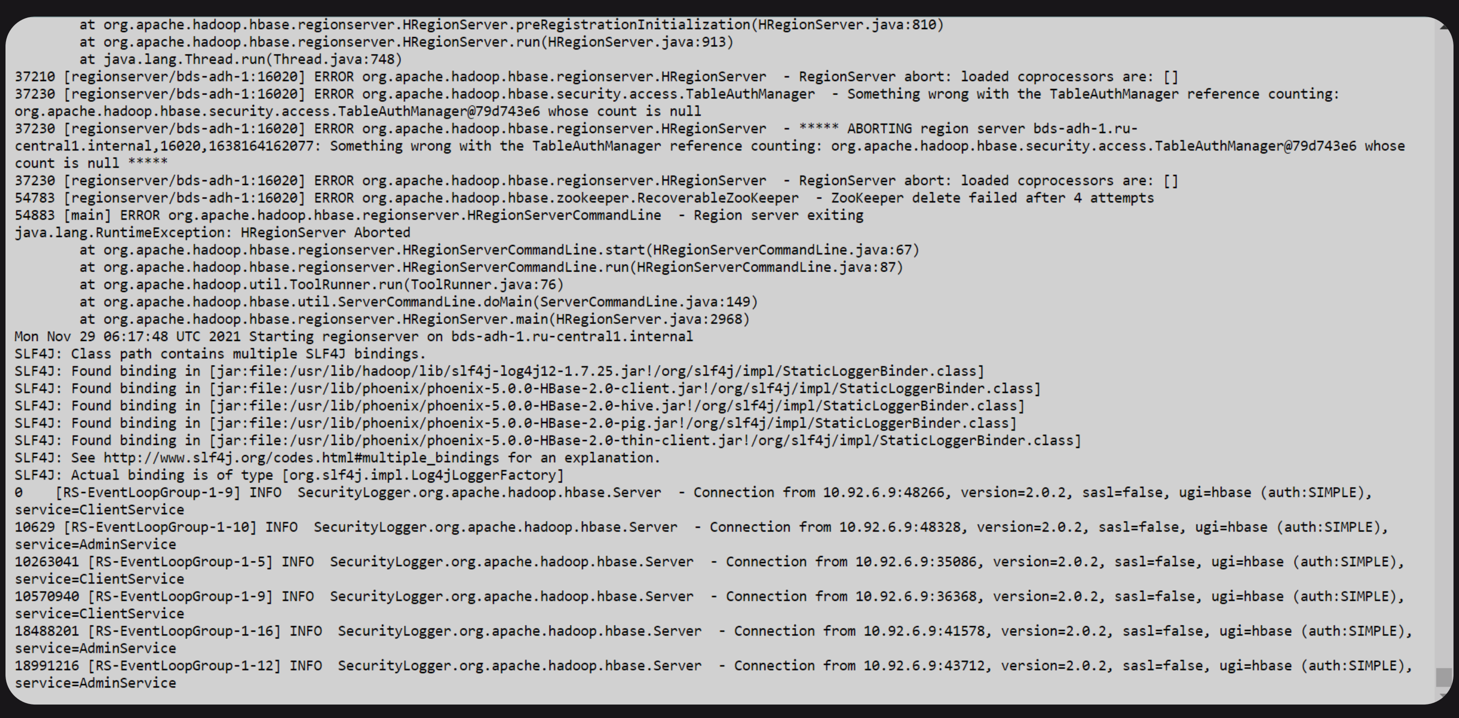


Directory: /logs/

hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.log	10209158 bytes	Nov 29, 2021 11:34:29 AM
hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out	692 bytes	Nov 29, 2021 6:17:48 AM
hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.1	692 bytes	Nov 29, 2021 5:35:28 AM
hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.2	692 bytes	Nov 26, 2021 7:43:48 AM
hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.3	692 bytes	Nov 26, 2021 5:58:51 AM
hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.4	692 bytes	Nov 26, 2021 5:26:44 AM
hbase-hbase-regionserver-bds-adh-1.ru-central1.internal.out.5	692 bytes	Nov 25, 2021 3:00:14 PM

Choosing the log file

4. View the opened file. The latest information is located in the end.



```
        at org.apache.hadoop.hbase.regionserver.HRegionServer.preRegistrationInitialization(HRegionServer.java:810)
        at org.apache.hadoop.hbase.regionserver.HRegionServer.run(HRegionServer.java:913)
        at java.lang.Thread.run(Thread.java:748)
37210 [regionserver/bds-adh-1:16020] ERROR org.apache.hadoop.hbase.regionserver.HRegionServer  - RegionServer abort: loaded coprocessors are: []
37230 [regionserver/bds-adh-1:16020] ERROR org.apache.hadoop.hbase.security.access.TableAuthManager  - Something wrong with the TableAuthManager reference counting:
org.apache.hadoop.hbase.security.access.TableAuthManager@79d743e6 whose count is null
37230 [regionserver/bds-adh-1:16020] ERROR org.apache.hadoop.hbase.regionserver.HRegionServer  - ***** ABORTING region server bds-adh-1.ru-
central1.internal,16020,1638164162077: Something wrong with the TableAuthManager reference counting: org.apache.hadoop.hbase.security.access.TableAuthManager@79d743e6 whose
count is null *****
37230 [regionserver/bds-adh-1:16020] ERROR org.apache.hadoop.hbase.regionserver.HRegionServer  - RegionServer abort: loaded coprocessors are: []
54783 [regionserver/bds-adh-1:16020] ERROR org.apache.hadoop.hbase.zookeeper.RecoverableZooKeeper  - ZooKeeper delete failed after 4 attempts
54883 [main] ERROR org.apache.hadoop.hbase.regionserver.HRegionServerCommandLine  - Region server exiting
java.lang.RuntimeException: HRegionServer Aborted
        at org.apache.hadoop.hbase.regionserver.HRegionServerCommandLine.start(HRegionServerCommandLine.java:67)
        at org.apache.hadoop.hbase.regionserver.HRegionServerCommandLine.run(HRegionServerCommandLine.java:87)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.hadoop.hbase.util.ServerCommandLine.doMain(ServerCommandLine.java:149)
        at org.apache.hadoop.hbase.regionserver.HRegionServer.main(HRegionServer.java:2968)
Mon Nov 29 06:17:48 UTC 2021 Starting regionserver on bds-adh-1.ru-central1.internal
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-hive.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-pig.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-thin-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
0    [RS-EventLoopGroup-1-9] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:48266, version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE),
service=ClientService
10629 [RS-EventLoopGroup-1-10] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:48328, version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE),
service=AdminService
10263041 [RS-EventLoopGroup-1-5] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:35086, version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE),
service=ClientService
10570940 [RS-EventLoopGroup-1-9] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:36368, version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE),
service=ClientService
18488201 [RS-EventLoopGroup-1-16] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:41578, version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE),
service=AdminService
18991216 [RS-EventLoopGroup-1-12] INFO  SecurityLogger.org.apache.hadoop.hbase.Server  - Connection from 10.92.6.9:43712, version=2.0.2, sasl=false, ugi=hbase (auth:SIMPLE),
service=AdminService
```

Viewing the log file

HBase service management via ADCM

Eugenia Kuzina

To Table of Contents

The ADCM UI provides actions to manage the HBase service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the HBase service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute HBase components. When running this action, you can specify the following parameters on the ***Configuration*** step: ▪ ***Remove RegionServer without region unloading***. When set to `true`, removes the RegionServer component without unloading data from it. ▪ ***Limit the number of threads used by the region mover***. Sets the maximum number of threads for the RegionMover process. ▪ ***Timeout for moving regions***. If regions cannot be moved within the specified timeout, the process exits with error.
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM. The service supports the ***Rolling restart*** option that allows you to restart its components one by one (or in batches) rather than all at once. This feature helps to minimize the service downtime during restarts. The ***Rolling restart*** option has the following parameters: ▪ ***Limit the number of threads used by the region mover*** — the maximum number of regions that can be moved in parallel. ▪ ***Timeout for moving regions*** — maximum time allowed to move regions before the operation fails. If regions cannot be moved within the allocated timeout, the restart process exits with an error. ▪ ***Graceful restart RegionServers*** — enables/disables a graceful restart of RegionServer components. This parameter can cause regions to be moved multiple times, which can degrade the HBase client performance. ▪ ***Roll masters*** — allows restarting HBase Master Server components one by one. If set to `false`, all HBase Master Server components are restarted in parallel. ▪ ***max_wait_time*** — the maximum timeout in seconds to wait for checking the results after the restart (Znode checks). ▪ ***pause_after_masters_restart*** — the idle timeout in seconds after restarting HBase Master Servers. May be required for initialization/stabilization needs.
Manage Ranger plugin	Enables or disables Ranger plugin for HBase. During this action RegionServers are restarted. When a RegionServer is shutdown, regions hosted by that RegionServer are moved to other RegionServers before stopping the current RegionServer. When you run this action, you have to specify the following parameters: `Limit the number of threads used by the region mover for graceful RegionServer restarting` and `Timeout for moving regions for graceful RegionServer restarting`. If regions are not moved by the timeout value, the action is interrupted, and an error occurs
Reconfig and graceful restart	Applies settings specified in ADCM UI and restarts the HBase service. During this action RegionServers are restarted. When a RegionServer is shutdown, regions hosted by that RegionServer are moved to other RegionServers before stopping the current RegionServer. When you run this action, you have to specify the following parameters: `Limit the number of threads used by the region mover for graceful RegionServer restarting` and `Timeout for moving regions for graceful RegionServer restarting`. If regions are not moved by the timeout value, the action is interrupted, and an error occurs

HBase components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Connect to HDFS

Elena Kostyuchenko

Contents

To Table of Contents

- Command line
- HTTP REST API
- C libhdfs library
  - Preparation steps
  - Write a connector
  - Compile and run
- Java API

This page describes how to connect to HDFS using CLI, WebHDFS API, and HDFS libraries. Additionally, you can interact with HDFS via web interface. For more information, see HDFS UI overview.

Command line

Using CLI, you can access HDFS from any host in a cluster. The following steps describe how to connect to HDFS using the command line. It's implied that access by SSH has already been configured for the cluster and it doesn't require a password.

1. Connect to an ADH cluster host via SSH:

```
$ ssh <USER>@<HOST>
```

   Where:

   - `<USER>` — name of a host user;
   - `<HOST>` — host's IP address.

2. When connection to the host is established, run a desired HDFS command. For example:

```
$ hdfs dfs -ls /
```

   Possible output of the `ls` command:

```
Found 4 items
drwxrwxrwt   - yarn hadoop          0 2023-07-24 13:47 /logs
drwxr-xr-x   - hdfs hadoop          0 2023-08-03 17:49 /system
drwxrwxrwx   - hdfs hadoop          0 2023-08-30 19:51 /tmp
drwxr-xr-x   - hdfs hadoop          0 2023-07-24 13:47 /user
```

For more information on the diferences between `hadoop fs`, `hadoop dfs`, and `hdfs dfs` commands, see the Hadoop command-line article.

Some commands require admin-level access (user `hdfs`). If you try to login as `hdfs` user, a password will be required. Since there is no default password for the `hdfs` user, log in as `root` on your local host. It will make it possible to authorize as the `hdfs` user without a password.

HTTP REST API

You can access HDFS using WebHDFS REST API.

To make a query using curl or a browser, structure the request as follows:

```
http://<HOST>:<HTTP_PORT>/webhdfs/v1/<PATH>?<AUTH>op=<COMMAND>
```

Where:

- `<HOST>` — host's IP address.
- `<HTTP_PORT>` — HTTP port of the active NameNode.
- `<PATH>` — query's target directory.
- `<AUTH>` — authentication request in the following format: `user.name=<USER>&`, where `<USER>` is the name of the host's user. This is an optional parameter. If not set, the request will be sent from the default user if it's configured. Otherwise, the server will return an error.
- `<COMMAND>` — a file system command.

If you have SSL enabled, replace `webhdfs` with `swebhdfs`.

HDFS REST API authentication

If security is on, authentication method is different for each security model.

When Kerberos SPNEGO is on, use the `--negotiate` and `-u` options:

```
$ curl -i --negotiate -u : "http://<HOST>:<HTTP_PORT>/webhdfs/v1/<PATH>?op=<COMMAND>"
```

When authentication with the Hadoop delegation token is on, pass the token in place of the username:

```
$ curl -i "http://<HOST>:<HTTP_PORT>/webhdfs/v1/<PATH>?delegation=<TOKEN>&op=..."
```

For more information about authentication methods, see Authentication for Hadoop HTTP web-consoles.

An example command for connecting to HDFS to run the `ls` command can look like this:

```
$ curl -i "http://127.0.0.1:14000/webhdfs/v1/?user.name=admin&op=LISTSTATUS"
```

And the possible output of the command:

```
HTTP/1.1 200 OK
Date: Mon, 04 Sep 2023 11:50:00 GMT
Cache-Control: no-cache
Expires: Mon, 04 Sep 2023 11:50:00 GMT
Date: Mon, 04 Sep 2023 11:50:00 GMT
Pragma: no-cache
Content-Type: application/json
Set-Cookie: hadoop.auth="u=admin&p=admin&t=simple-dt&e=6593757/Hvhgbi/PxjQ="; Path=/; HttpOnly
Transfer-Encoding: chunked

{"FileStatuses":{"FileStatus":
[{"pathSuffix":"logs","type":"DIRECTORY","length":0,"owner":"yarn","group":"hadoop","permission":"1777","accessTime":0,"modificat
ionTime":1690206465850,"blockSize":0,"replication":0},
{"pathSuffix":"system","type":"DIRECTORY","length":0,"owner":"hdfs","group":"hadoop","permission":"755","accessTime":0,"modificat
ionTime":1691084968551,"blockSize":0,"replication":0},
{"pathSuffix":"tmp","type":"DIRECTORY","length":0,"owner":"hdfs","group":"hadoop","permission":"777","accessTime":0,"modification
Time":1693425076362,"blockSize":0,"replication":0},
{"pathSuffix":"user","type":"DIRECTORY","length":0,"owner":"hdfs","group":"hadoop","permission":"755","accessTime":0,"modificatio
nTime":1690206432153,"blockSize":0,"replication":0}]}}
```

C libhdfs library

To connect to a remote HDFS host using the libhdfs library, you need to set up an environment and install the dependencies. In this guide, we'll create a program for connecting to a remote HDFS host. The developer tools used in the process are Vi text editor, virtual machine with CentOS 7, and GCC compiler. A different setup will require calibration.

Preparation steps

To build and run the following example, install the following:

- ADH or build one yourself (your local ADH version must match version of the remote one).
- Text editor of your choice.
- The C compiler of your choice.
- JDK.
- `libhdfs.so` with the header files or build it yourself from source.

Write a connector

The example code below creates a connection to a remote HDFS, outputs confirmation upon successful connection, and returns an error message if it was unable to connect. If you successfully connected to HDFS but no longer need it, terminate the connection.

```
#include <hdfs.h> ①
#include <stdio.h>

int main(int argc, char **argv) {
    struct hdfsBuilder *builder = hdfsNewBuilder(); ②
    hdfsBuilderSetNameNode(builder, "hdfs://127.0.0.1:8020"); ③
    hdfsFS fs = hdfsBuilderConnect(builder); ④
    if (fs != NULL) {
      printf("Successfully connected to HDFS ");
    } else {
      printf("Failed to connect to HDFS ");
    }
}
```

① Import the *libhdfs* library.

② Initialize the builder object for connecting to a remote HDFS using the `hdfsBuilder` method. It's preferable to use this method instead of the deprecated `hdfsConnect`.

③ Set parameters for connection in the URI format `hdfs://<NN-IP>:<PORT>`, where `<NN-IP>` is the IP address of the active NameNode, and `<PORT>` is the NameNode metadata service port.

④ Call `builder` to create a connection.

Compile and run

1. Create a compilation script for GCC and fill in the paths variables:

```
#!/bin/bash

export HADOOP_HOME=/usr/lib/hadoop ①
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.382.b05-1.el7_9.x86_64/jre ②
export CLASSPATH=`$HADOOP_HOME/bin/hadoop classpath --glob` ③

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JAVA_HOME/jre/lib/amd64/server

gcc -std=c99 -lpthread main.c -I$HADOOP_HOME/include -L$HADOOP_HOME/lib/native -lhdfs -I$JAVA_HOME/include -
I$JAVA_HOME/include/linux -L$JAVA_HOME/jre/lib/amd64/server -ljvm -o my_hdfs_program ④

./my_hdfs_program ⑤
```

   ① Set the Hadoop libraries' path.

   ② Set the JDK libraries' path.

   ③ Generate the correct `CLASSPATH` for your deployment using the classpath command.

   ④ Compile the program using the correct paths and additional options.

   ⑤ Run the program.

2. Run the compilation script:

```
$ ./<script-name>.sh
```

   Where `<script-name>` is the name of your compilation script.

Java API

To connect to a remote HDFS host using the Java API, you need to set up an environment and install the dependencies. The example code below creates a connection to a remote HDFS, outputs confirmation upon successful connection, and returns an error message if it was unable to connect.

The developer tools used in the process are Vi text editor, virtual machine with CentOS 7, and Maven compiler. A different setup will require calibration.

1. Install Maven.
2. Write a connector class:

```
package my.example.hdfs;

import org.apache.hadoop.conf.Configuration; ①
import org.apache.hadoop.fs.FileSystem;

import java.net.URI;

public class Main {
   public static void main(String[] args) throws Exception {
      String hdfs_uri = "hdfs://127.0.0.1:8020"; ②

      Configuration conf = new Configuration(); ③

      conf.set("fs.defaultFS", hdfs_uri);

      try {
          FileSystem fs = FileSystem.get(URI.create(hdfs_uri), conf); ④
          System.out.println("Successfully connected to HDFS ");

          fs.close(); ⑤
      } catch (Exception e) {
          System.out.print("Failed to connect to HDFS: ");
          System.out.println(e.getMessage());
      }
   }
}
```

   ① Import the necessary libraries.

   ② Set parameters for connection in the URI format `hdfs://<NN-IP>:<PORT>`, where `<NN-IP>` is the IP address of the active NameNode, and `<PORT>` is the NameNode metadata service port.

   ③ Initialize the filesystem object.

   ④ Call FileSystem.get to create a connection.

   ⑤ Terminate a connection if it's no longer needed.

3. Create a *pom.xml* file for building the project with Maven.

Sample pom.xml file

```
<project>
        <modelVersion>4.0.0</modelVersion>
        <groupId>my.example.hdfs</groupId>
        <artifactId>my-hdfs-program</artifactId>
        <version>0.1.0-SNAPSHOT</version>
        <packaging>jar</packaging>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>2.7.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-core</artifactId>
            <version>1.2.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>2.7.1</version>
        </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.1.0</version>
            <configuration>
              <archive>
                <manifest>
                  <addClasspath>true</addClasspath>
                  <mainClass>my.example.hdfs.Main</mainClass>
                </manifest>
              </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>assemble-all</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
      <plugin>
        <!-- Build an executable JAR -->
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.1.0</version>
        <configuration>
          <archive>
            <manifest>
              <addClasspath>true</addClasspath>
              <mainClass>my.example.hdfs.Main</mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

4. Build the project:

```
$ mvn package
```

5. Run the created Java class:

```
$ java -jar <java-class>.jar
```

   Where `<java-class>` is the name of your Java connector program.

HDFS UI overview

Elena Kostyuchenko

Contents

To Table of Contents

- NameNode
  - Overview
  - Datanodes
  - Datanode Volume Failures
  - Snapshot
  - Startup Progress
  - Utilities
- DataNode
  - Overview
  - Utilities
- HttpFS Server

HDFS web interfaces (UI) enable direct access to the file system and display diagnostic and status information. The UIs are available by the `http://<server-hostname>:<port>/` URL, where `<server-hostname>` is the IP address of a host or, if you have configured the redirect in the `hosts` file, FQDN of a host. And `<port>` is the port of the HDFS component's UI.

If you have an SSL certificate configured, use `https://` instead of `http://`.

HDFS provides several web interfaces:

- one interface for each NameNode;
- one interface for each DataNode;
- one HttpFS interface.

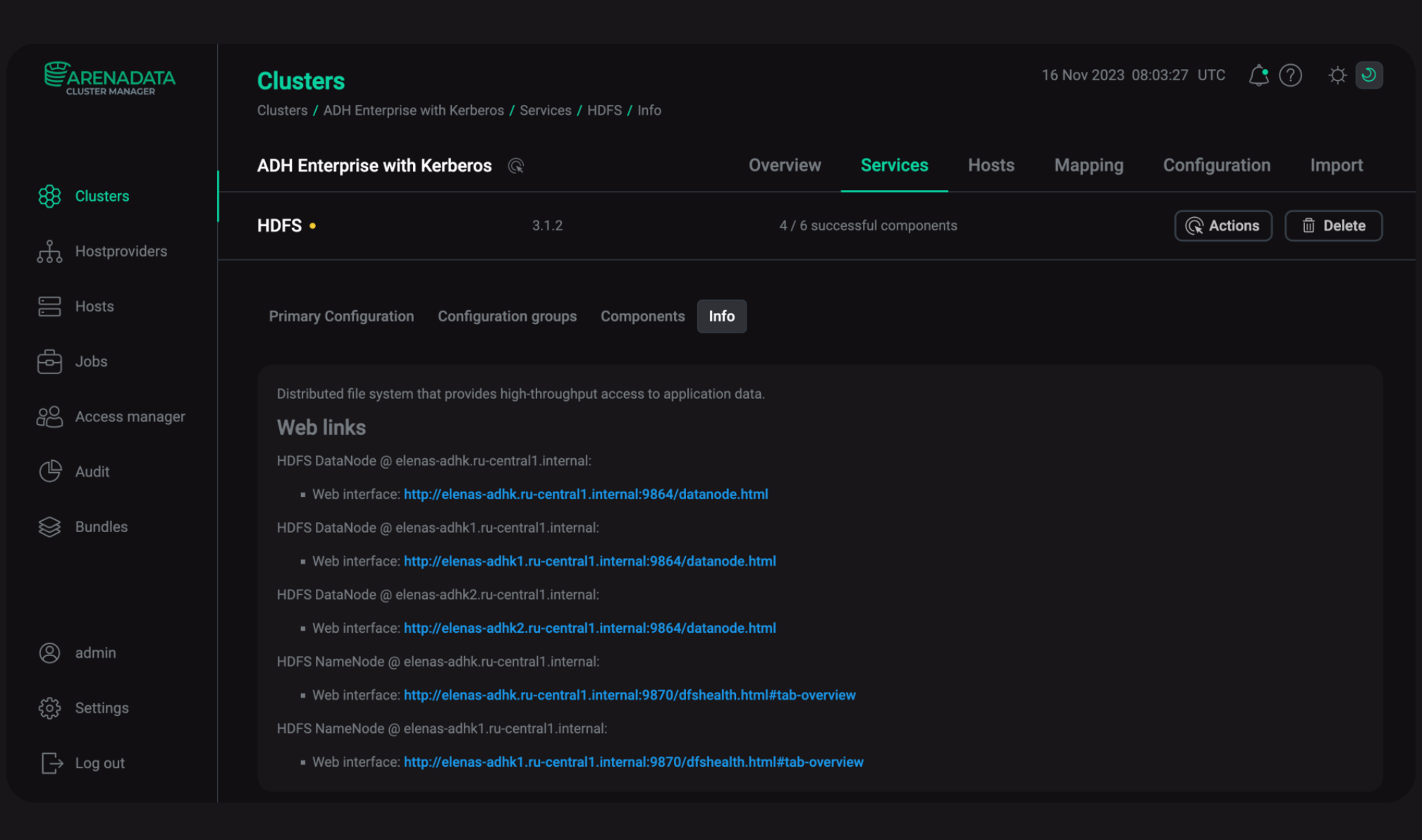
The default web server ports are listed in the table below.

Component	HTTP Port	HTTPs Port
NameNode	9870	9871
DataNode	9864	9865
HttpFS	14000	14000

To change the default web server port, edit the corresponfnding `http-address` parameter in the *hdfs-site.xml* configuration file.

You can find links to HDFS UIs in ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Select ***Info***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH Enterprise with Kerberos / Services / HDFS / Info
16 Nov 2023 08:03:27 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
HDFS
3.1.2
4 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info

Distributed file system that provides high-throughput access to application data.

Web links

HDFS DataNode @ elenas-adhk.ru-central1.internal:

- Web interface: http://elenas-adhk.ru-central1.internal:9864/datanode.html

HDFS DataNode @ elenas-adhk1.ru-central1.internal:

- Web interface: http://elenas-adhk1.ru-central1.internal:9864/datanode.html

HDFS DataNode @ elenas-adhk2.ru-central1.internal:

- Web interface: http://elenas-adhk2.ru-central1.internal:9864/datanode.html

HDFS NameNode @ elenas-adhk.ru-central1.internal:

- Web interface: http://elenas-adhk.ru-central1.internal:9870/dfshealth.html#tab-overview

HDFS NameNode @ elenas-adhk1.ru-central1.internal:

- Web interface: http://elenas-adhk1.ru-central1.internal:9870/dfshealth.html#tab-overview

HDFS service web interface links

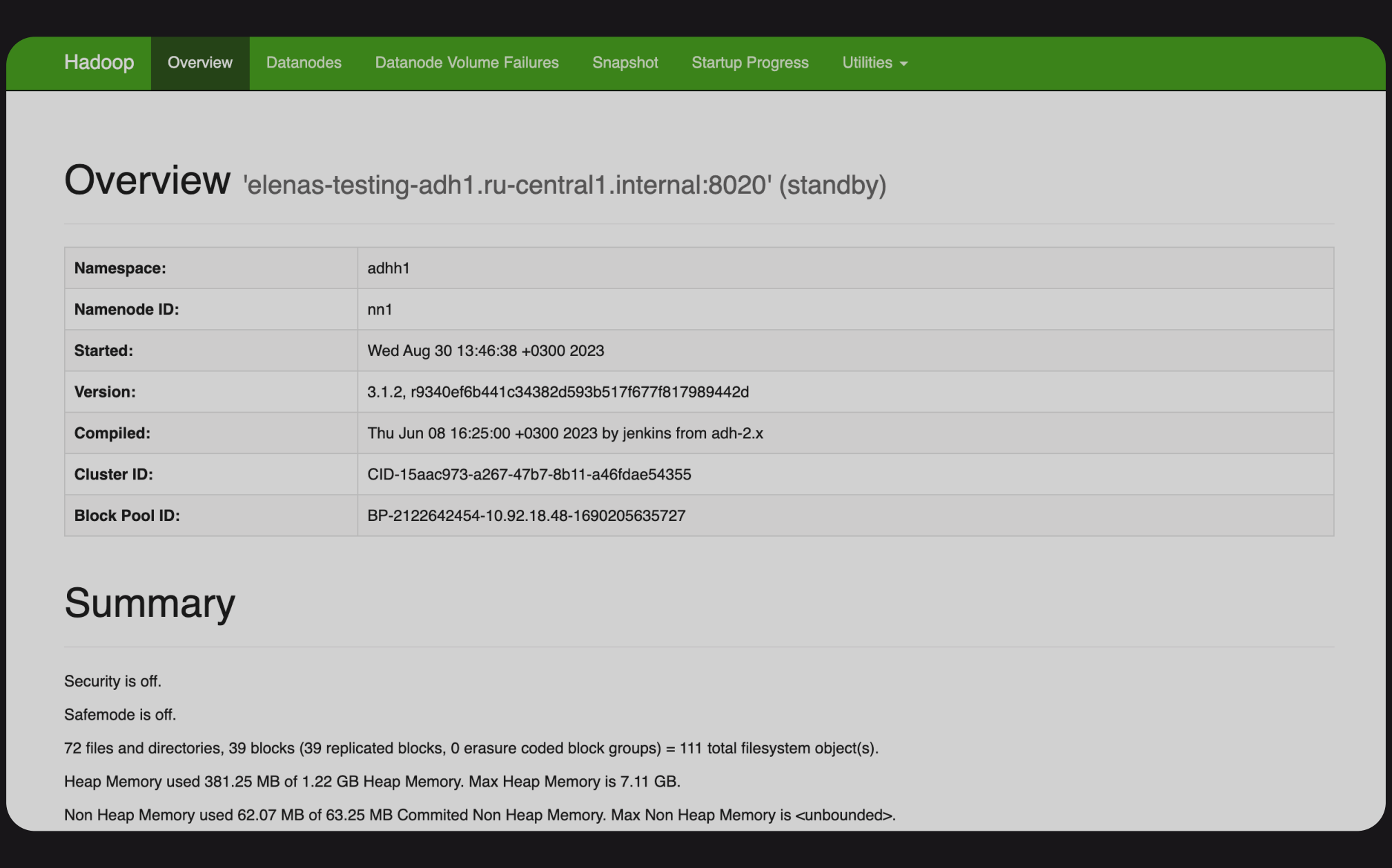
NameNode

The NameNode interface includes information about cluster configuration, DataNode status, file system, and logs.

Overview

The main page contains the following information:

- ***Overview*** — basic information about the NameNode: type (active or standby), hostname, ID, and other.
- ***Summary*** — status and configuration information.
- ***NameNode Journal Status*** — information about the current *edits* file.
- ***NameNode Storage*** — NameNode storage information: directory, type, and state.
- ***DFS Storage Types*** — storage types used and their capacity.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Overview 'elenas-testing-adh1.ru-central1.internal:8020' (standby)

Namespace:	adhh1
Namenode ID:	nn1
Started:	Wed Aug 30 13:46:38 +0300 2023
Version:	3.1.2, r9340ef6b441c34382d593b517f677f817989442d
Compiled:	Thu Jun 08 16:25:00 +0300 2023 by jenkins from adh-2.x
Cluster ID:	CID-15aac973-a267-47b7-8b11-a46fdae54355
Block Pool ID:	BP-2122642454-10.92.18.48-1690205635727

Summary

Security is off.

Safemode is off.

72 files and directories, 39 blocks (39 replicated blocks, 0 erasure coded block groups) = 111 total filesystem object(s).

Heap Memory used 381.25 MB of 1.22 GB Heap Memory. Max Heap Memory is 7.11 GB.

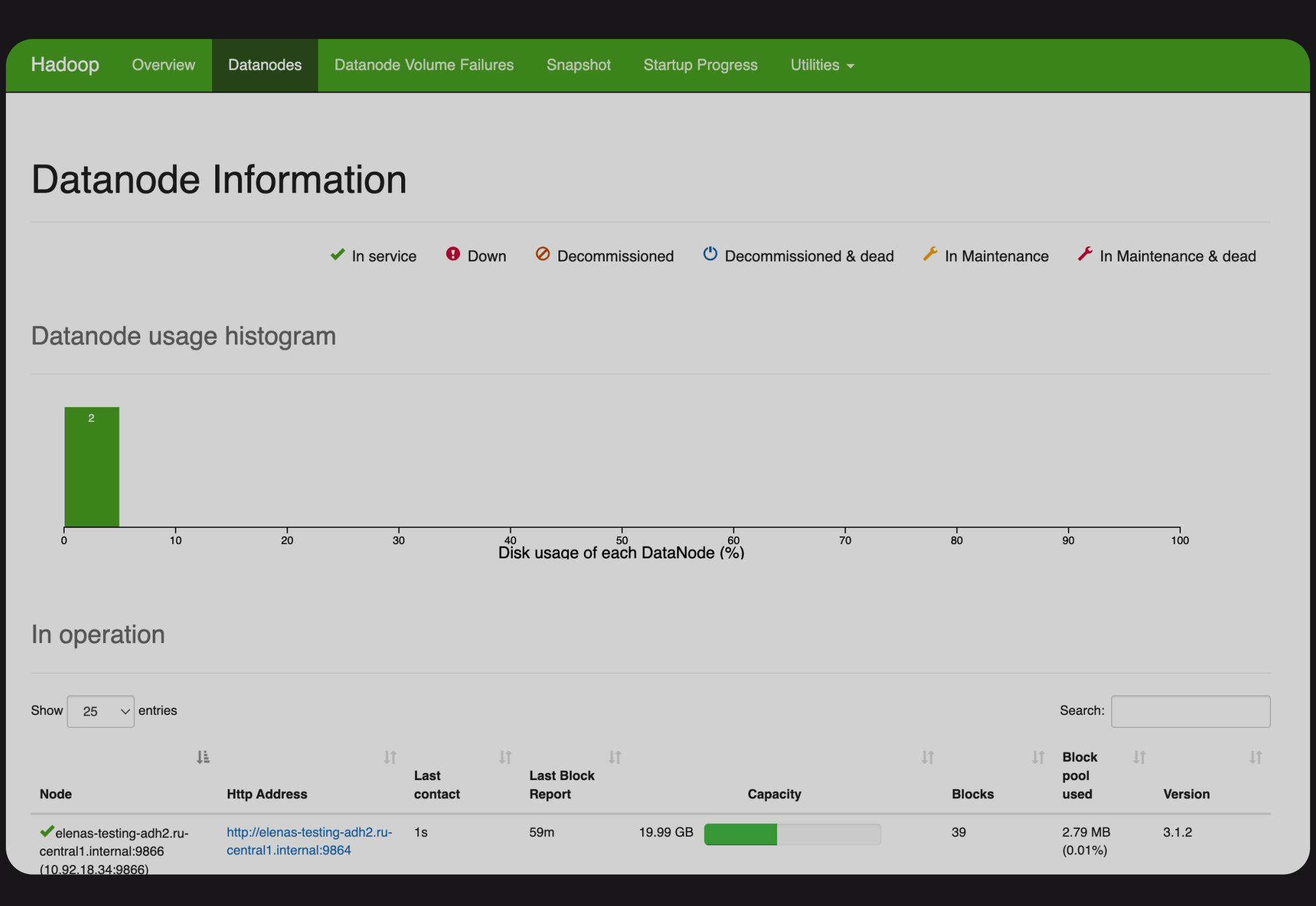
Non Heap Memory used 62.07 MB of 63.25 MB Commited Non Heap Memory. Max Non Heap Memory is <unbounded>.

NameNode UI: Overview page

Datanodes

Datanodes page displays a histogram of disk usage for each DataNode, status information for each DataNode, and links to DataNodes UIs.

For DataNodes preparing for maintenance or being decommissioned, the page shows hostnames and blocks replication statistics.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Datanode Information
✓ In service
Down
Decommissioned
Decommissioned & dead
In Maintenance
In Maintenance & dead
Datanode usage histogram
2
0
10
20
30
40
50
60
70
80
90
100
Disk usage of each DataNode (%)
In operation
Show 25 entries
Search:

Node	Http Address	Last contact	Last Block Report	Capacity		Blocks	Block pool used	Version
✓elenas-testing-adh2.ru-central1.internal:9866 (10.92.18.34:9866)	http://elenas-testing-adh2.ru-central1.internal:9864	1s	59m	19.99 GB		39	2.79 MB (0.01%)	3.1.2

NameNode UI: Datanodes page

DataNodes have one of the following states:

- ***In service*** — the DataNode regularly sends heartbeat to the NameNode.
- ***Down*** — the DataNode has not been sending heartbeat to the NameNode for more than 10 minutes.
- ***Decommissioned*** — the DataNode is in the `Decommissioned` state but can be reinstated.
- ***Decommissioned & dead*** — the DataNode has been `Decommissioned` and now is in the `Down` state.
- ***In Maintenance*** — the DataNode is in the maintenance state.
- ***In Maintenance & dead*** — the DataNode has been put under maintenance and now is in the `Down` state.

Datanode Volume Failures

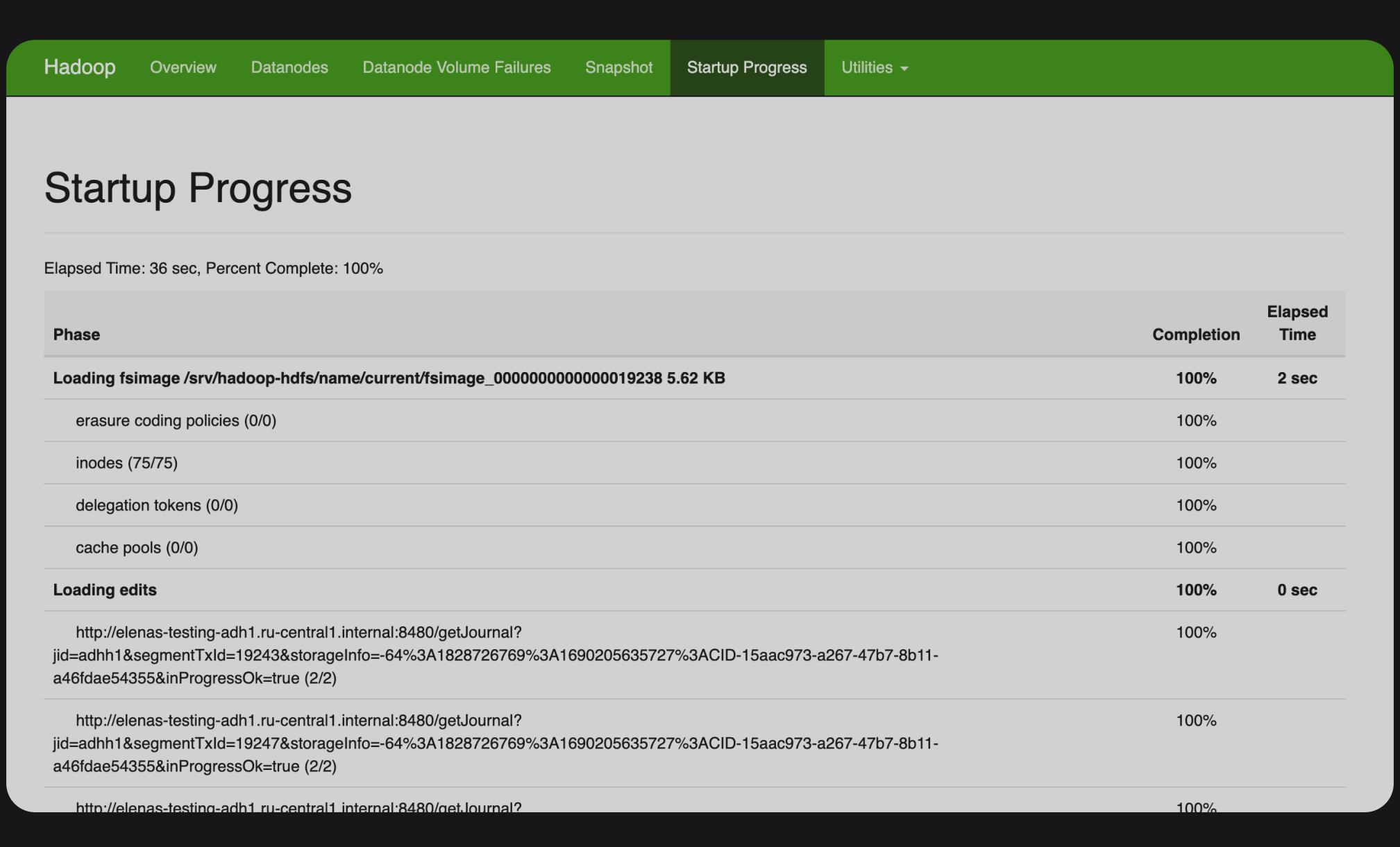
The page provides information about volume failures registered on the DataNodes.

Snapshot

The Snapshot page provides information on created snapshots and directories that can be snapshotted.

Startup Progress

The page provides information about the latest fsimage loaded and the applied *edits*.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities

Startup Progress

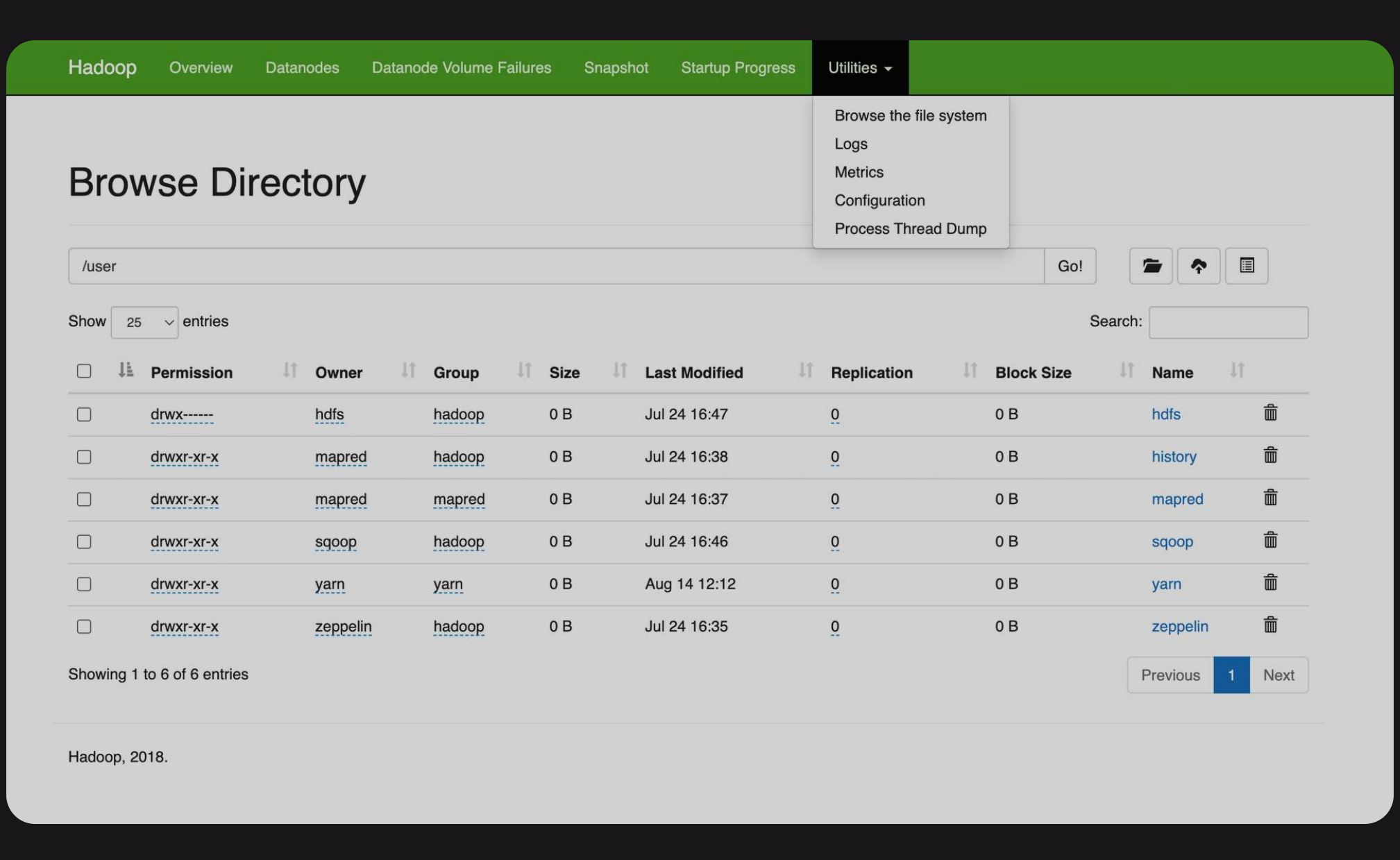
Elapsed Time: 36 sec, Percent Complete: 100%

Phase	Completion	Elapsed Time
Loading fsimage /srv/hadoop-hdfs/name/current/fsimage_0000000000000019238 5.62 KB	**100%**	**2 sec**
erasure coding policies (0/0)	100%	
inodes (75/75)	100%	
delegation tokens (0/0)	100%	
cache pools (0/0)	100%	
Loading edits	**100%**	**0 sec**
http://elenas-testing-adh1.ru-central1.internal:8480/getJournal?jid=adhh1&segmentTxId=19243&storageInfo=-64%3A1828726769%3A1690205635727%3ACID-15aac973-a267-47b7-8b11-a46fdae54355&inProgressOk=true (2/2)	100%	
http://elenas-testing-adh1.ru-central1.internal:8480/getJournal?jid=adhh1&segmentTxId=19247&storageInfo=-64%3A1828726769%3A1690205635727%3ACID-15aac973-a267-47b7-8b11-a46fdae54355&inProgressOk=true (2/2)	100%	

NameNode UI: Startup Progress

Utilities

Utilities provide access to the file system, configuration, show information about available metrics, logs, and active threads.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse the file system
Logs
Metrics
Configuration
Process Thread Dump
Go!
Previous
1
Next

Browse Directory

/user

Show 25 entries

☐	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwx------	hdfs	hadoop	0 B	Jul 24 16:47	0	0 B	hdfs
☐	drwxr-xr-x	mapred	hadoop	0 B	Jul 24 16:38	0	0 B	history
☐	drwxr-xr-x	mapred	mapred	0 B	Jul 24 16:37	0	0 B	mapred
☐	drwxr-xr-x	sqoop	hadoop	0 B	Jul 24 16:46	0	0 B	sqoop
☐	drwxr-xr-x	yarn	yarn	0 B	Aug 14 12:12	0	0 B	yarn
☐	drwxr-xr-x	zeppelin	hadoop	0 B	Jul 24 16:35	0	0 B	zeppelin

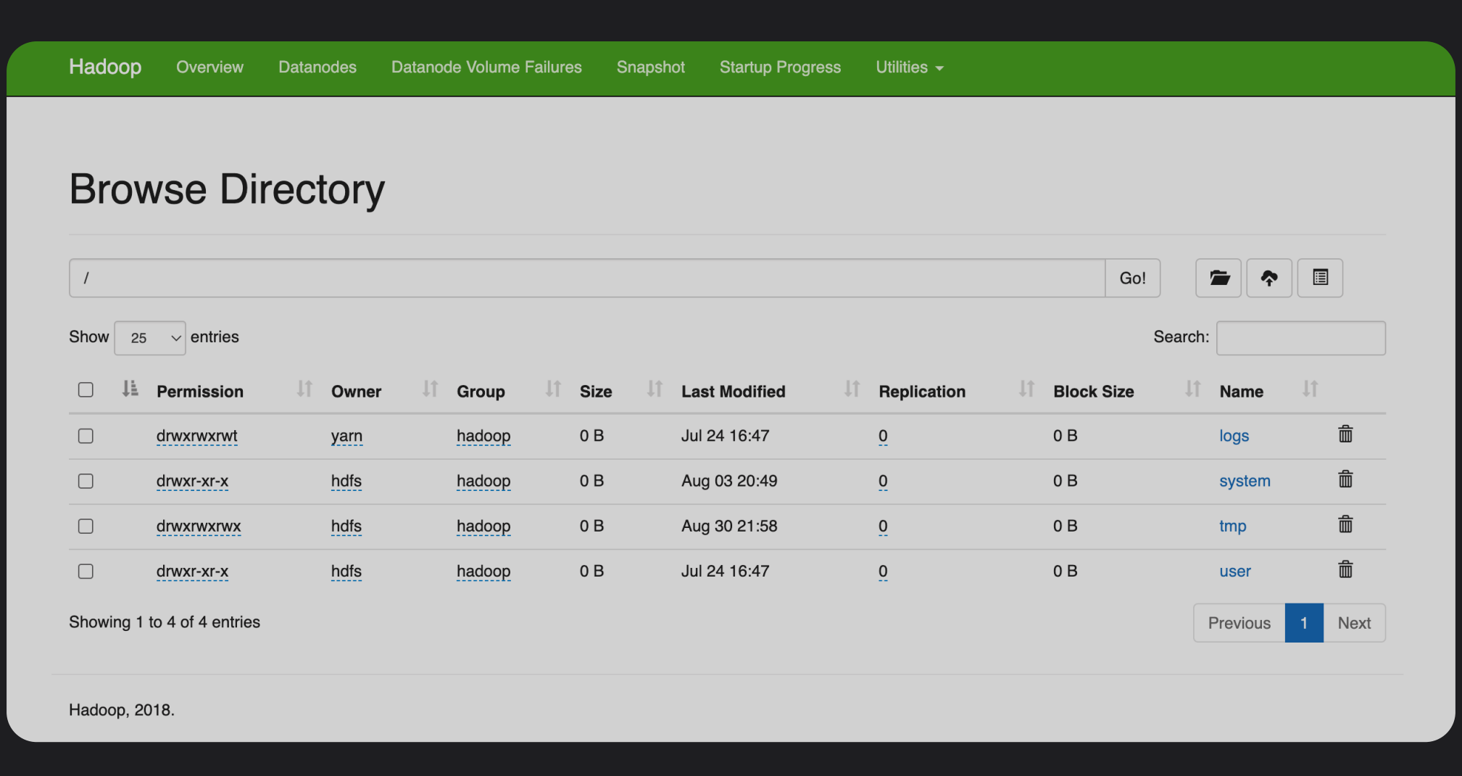
Showing 1 to 6 of 6 entries

Hadoop, 2018.

NameNode UI: Utilities menu

Browse the file system

This page provides access to the file system. It can be accessed only via the active NameNode UI.


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Go!
Previous
1
Next

Browse Directory

/

Show 25 entries

☐	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxrwxrwt	yarn	hadoop	0 B	Jul 24 16:47	0	0 B	logs
☐	drwxr-xr-x	hdfs	hadoop	0 B	Aug 03 20:49	0	0 B	system
☐	drwxrwxrwx	hdfs	hadoop	0 B	Aug 30 21:58	0	0 B	tmp
☐	drwxr-xr-x	hdfs	hadoop	0 B	Jul 24 16:47	0	0 B	user

Showing 1 to 4 of 4 entries

Hadoop, 2018.

NameNode UI: Browse Directory

For each file or directory the following information is available:

- ***Permission*** — permissions set for a file or directory. Click to change the permissions.
- ***Owner*** — the owner of a file or directory.
- ***Group*** — the security group for a file or directory.
- ***Size*** — the size of a file or directory.
- ***Last Modified*** — the date of the last changes made to a file or directory.
- ***Replication*** — the amount of replicas for each block. You can click on this parameter to set a different replication factor for a file. The default replication factor is 3.
- ***Block Size*** — the size of a single block in a file. The default block size is 128 MB.
- ***Name*** — file or directory name. By clicking on the directory name, you open a child directory; clicking on the file name opens the ***File information window***, where you can download the file and see information about its blocks.

In the same page, there are following available actions:

- ***Create directory*** — creates a new directory.
- ***Upload files*** — enables file uploading directly from UI.
- ***Cut & Paste*** — provides cut and paste actions.
- ***Delete*** — deletes a file or directory.

Some of the actions might require additional access rights for the Web UI application user, `dr.who`.

Logs

The ***Logs*** page provides the list of logs for DataNodes, NameNodes, JournalNodes, and the HDFS ZKFC service.

Metrics

The ***Metrics*** page provides the list of metrics available in HDFS.

Configuration

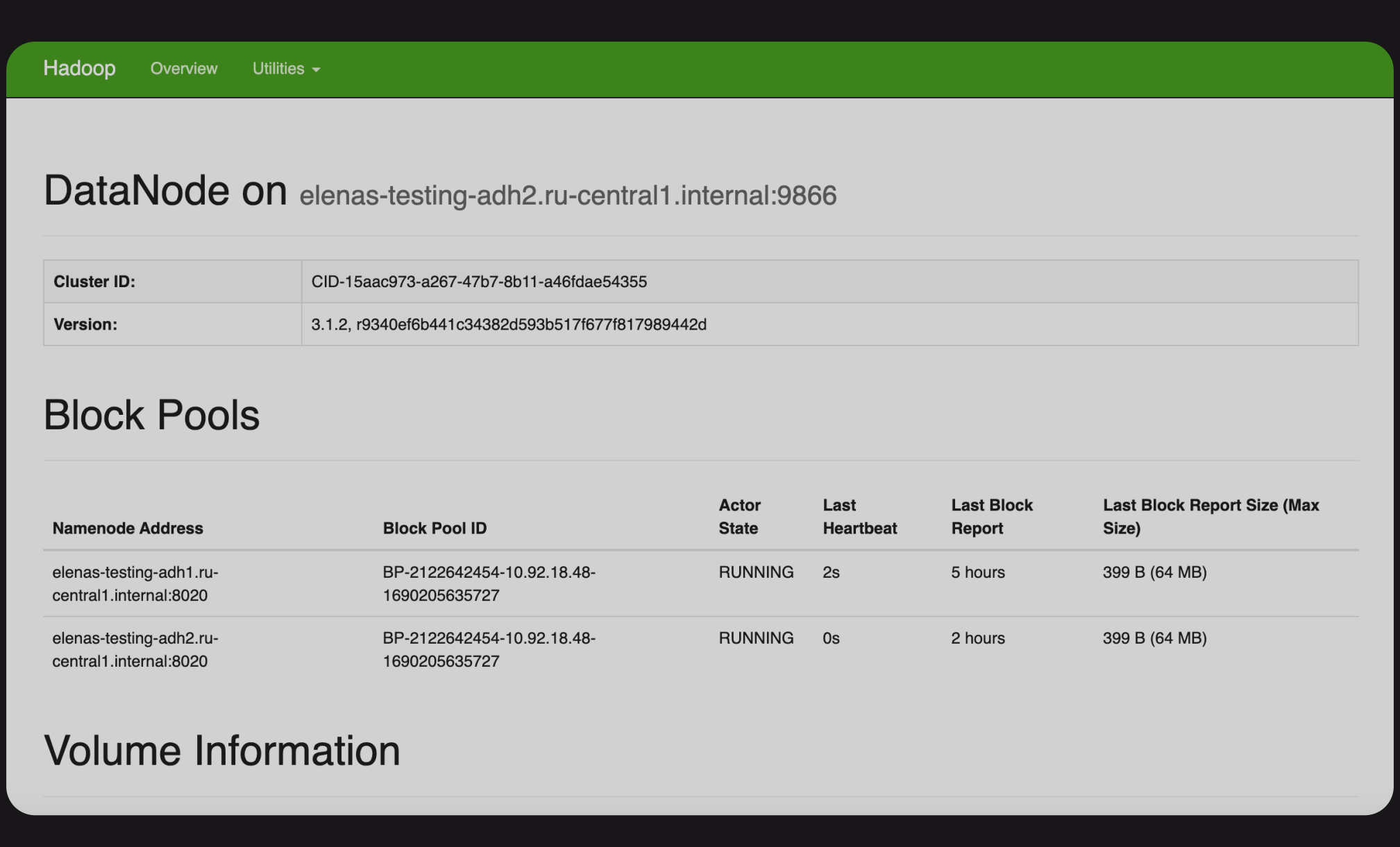
The ***Configuration*** page provides the list of all HDFS configuration parameters and their values in a single XML file.

Process Thread Dump

The ***Process Thread Dump*** page provides the list of active threads and their status.

DataNode

The DataNode interface includes basic information about the DataNode, it's status, configuration, and logs.


Hadoop
Overview
Utilities

DataNode on elenas-testing-adh2.ru-central1.internal:9866

Cluster ID:	CID-15aac973-a267-47b7-8b11-a46fdae54355
Version:	3.1.2, r9340ef6b441c34382d593b517f677f817989442d

Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
elenas-testing-adh1.ru-central1.internal:8020	BP-2122642454-10.92.18.48-1690205635727	RUNNING	2s	5 hours	399 B (64 MB)
elenas-testing-adh2.ru-central1.internal:8020	BP-2122642454-10.92.18.48-1690205635727	RUNNING	0s	2 hours	399 B (64 MB)

Volume Information

DataNode UI: Overview page

Overview

The main page contains the following information:

- ***DataNode on*** — DataNode hostname, port, cluster ID, and HDFS version.
- ***Block Pools*** — information about Block Pools stored in the DataNode.
- ***Volume Information*** — information about the DataNode directories and disk space usage.

Utilities

DataNode utilities provide the same information as the NameNode utilities: HDFS configuration, available metrics, logs, and active threads.

Logs

The ***Logs*** page provides the list of logs for DataNodes, NameNodes, JournalNodes, and the HDFS ZKFC service.

Metrics

The ***Metrics*** page provides the list of metrics available in HDFS.

Configuration

The ***Configuration*** page provides the list of all HDFS configuration parameters and their values in a single XML file.

Process Thread Dump

The ***Process Thread Dump*** page provides the list of active threads and their status.

HttpFS Server

The HttpFS Server UI is a web application that provides access to the file system via webhdfs REST HTTP API.

This UI contains the list of links:

- ***/webhdfs/v1/?op=LISTSTATUS*** — root directory list.
- ***HttpFS configuration properties*** — the list of all HDFS configuration parameters and their values in a single XML file.
- ***HttpFS JMX*** — the list of metrics available in HDFS.
- ***HttpFS log level*** — `GET` or `SET` log level.
- ***HttpFS log files*** — the list of logs for HDFS HttpFS Server.
- ***HttpFS stacks*** — the list of active threads and their status.

Protect files in HDFS

Albert Bagdasaryan

Contents

To Table of Contents

File and directory permissions
Access control list

You can use two methods for file protection in HDFS:

- Setting file permissions.
- Creating access control lists (ACL).

File and directory permissions

In HDFS, you can restrict access to files or directories using a standard model based on POSIX with modifications. In HDFS, you can grant permissions to a file for its owner, a specified user group, and other users.

For HDFS files, you can use the following flags:

- `r` — for reading.
- `w` — for writing.

Unlike other file systems, HDFS stores only data files, so there is no way to start execution of a particular file. Therefore, HDFS files lack such an access flag as `x`, which would mean the execution privilege. For the same reason, HDFS does not have such access flags as `setUID` and `setGID`.

The following flags grant different access types to a directory in HDFS:

- `r` — for listing contents in the directory.
- `w` — for creating, renaming, and deleting files and directories in it (assuming that also the permission `x` is granted).
- `x` — for entering the directory and getting access to all children in it (makes the directory a working directory for the affected users).

As with files, we don't have flags similar to `setUID` and `setGID` flags for directories either.

Before managing permissions, you should get an access to the NameNode CLI (command-line interface). After that, follow these steps using CLI:

1. Read the root directory contents:

```
$ hdfs dfs -ls /
```

The output looks similar to this:

```
-rw-r--r--   3 hdfs hadoop         20 2021-10-07 13:34 /hadoop
drwxrwxrwt   - yarn hadoop          0 2021-09-15 16:58 /logs
drwxr-xr-x   - hdfs hadoop          0 2021-10-12 13:17 /staging
drwxr-xr-x   - hdfs hadoop          0 2021-09-15 16:57 /system
drwxrwxrwx   - hdfs hadoop          0 2021-10-20 09:00 /tmp
drwxr-xr-x   - hdfs hadoop          0 2021-09-27 12:24 /user
```

Take a look at the permissions to the */logs* directory, where we have the flags `drwxrwxrwt`:

- `d` specifies a directory.
- The first three `rwx` flags indicate the file or the directory owner's permissions.
- The second `rwx` flags indicate permissions for the specified user group.
- The third `rwx` flags indicate permissions for other users.

2. List file permissions in the */tmp* directory as in the following example:

```
$ hdfs dfs -ls /tmp
```

The output looks similar to this:

```
drwxr-xr-x   - hdfs hadoop          0 2021-10-20 09:03 /tmp/hadoop01
-rw-r--r--   3 hdfs hadoop         20 2021-10-07 15:36 /tmp/test01
-rw-r--r--   3 hdfs hadoop         21 2021-10-08 08:40 /tmp/test02
-rw-r--r--   3 hdfs hadoop         19 2021-10-08 08:27 /tmp/test03
-rw-r--r--   3 hdfs hadoop          0 2021-10-20 09:00 /tmp/test04
```

In this example, the files are assigned the `read` and `write` privileges for the owner (`hdfs` user), `read` for the `hadoop` group, and `read` for other users.

3. Before changing file permissions, use the `sudo` and `su` commands (switch user) to switch to the file owner whose name is `hdfs`.

```
$ sudo -s
$ su - hdfs
```

You used the above actions to grant yourself the privileges to access files and directories as their owner. Now we execute all the following commands using the `hdfs` user privileges.

4. Change the permissions for the group and other users by adding the `w` flag for them:

```
$ hdfs dfs -chmod  go+w /tmp/test01
```

Check the permissions to the */tmp/test01* file:

```
$ hdfs dfs -ls /tmp
```

Ensure that the access permissions for the `hadoop` group and other users have changed:

```
drwxr-xr-x   - hdfs hadoop          0 2021-10-20 09:03 /tmp/hadoop01
-rw-rw-rw-   3 hdfs hadoop         20 2021-10-07 15:36 /tmp/test01
-rw-r--r--   3 hdfs hadoop         21 2021-10-08 08:40 /tmp/test02
-rw-r--r--   3 hdfs hadoop         19 2021-10-08 08:27 /tmp/test03
-rw-r--r--   3 hdfs hadoop          0 2021-10-20 09:00 /tmp/test04
```

The required permissions are granted. In this way you can change permissions for any file or directory available in your HDFS.

You can also change `group` or `owner` for files and directories using the `chgrp` and `chown` commands.

Access control list

Access control list contains access permissions for users and groups to files and directories. ACL provides you with a more flexible and visual representation of this process. For example, you can use ACL if you need to set permissions for different users and groups to the same file or directory. To enable ACL in HDFS, activate the appropriate option on your Hyperwave cluster.

The following steps will guide you through the ACL setup process:

1. Enable ACL by adding the following property in the *hdfs-site.xml* file:

```
dfs.namenode.acls.enabled : true
```

NOTE

To configure ADH, we recommend you to use Arenadata Cluster Manager (ADCM).

2. Reset your HDFS service to apply the changed configuration. Using ADCM, run a task that reconfigures and restarts the cluster.

IMPORTANT

Check whether all your services have successful status (highlighted in green).

3. Run the terminal on the selected NameNode and execute the following command to read the contents of your test Hyperwave directory, for example, */tmp*:

```
$ hdfs dfs -ls /tmp
```

This command displays all files and directories in the */tmp* directory.

4. Choose a file, for example, *test01*, and execute the following command to get its ACL:

```
$ hdfs dfs -getfacl /tmp/test01
```

The output looks similar to the following:

```
# file: /tmp/test01
# owner: hdfs
# group: hadoop
user::rw-
group::rw-
other::rw-
```

5. Add a user, for example, `yarn`, with all privileges to this file by using the following command:

```
$ hdfs dfs -setfacl -m user:yarn:rwx /tmp/test01
```

6. Repeat the command `hdfs dfs -getfacl /tmp/test01` to check the updated access list similar to this:

```
# file: /tmp/test01
# owner: hdfs
# group: hadoop
user::rw-
user:yarn:rwx
group::rw-
mask::rwx
other::rw-
```

The new user `yarn` can read and write to the test file. This example illustrates that ACLs provide more advanced features for managing access to HDFS files and directories.

Copy data from the local file system to HDFS

Albert Bagdasaryan

To Table of Contents

This page describes how to copy any data from the local file system to the Hadoop file system. For this case, there is a special command `hdfs dfs -put` with the following syntax:

```
$ hdfs dfs -put <source_path> <destination_path>
```

NOTE

The destination path should be a directory.

For more information, type `hdfs dfs -put` in the terminal to list all the arguments for this command.

Follow these steps to copy data files from the local file system to HDFS.

1. Make sure the data file, for example, *file1.txt*, is stored in the local file system, for example, in your home directory:

```
$ ls -a ~
```

   The output looks similar to this:

```
.  ..  .ansible  .bash_history  .bash_logout  .bash_profile  .bashrc  file1.txt  .ssh  test
```

2. Check the HDFS root directory contents:

```
$ hdfs dfs -ls /
```

   The output looks similar to this:

```
Found 6 items
-rw-r--r--   3 hdfs hadoop         20 2021-10-07 13:34 /hadoop
drwxrwxrwt   - yarn hadoop          0 2021-09-15 16:58 /logs
drwxr-xr-x   - hdfs hadoop          0 2021-10-12 13:17 /staging
drwxr-xr-x   - hdfs hadoop          0 2021-09-15 16:57 /system
drwxrwxrwx   - hdfs hadoop          0 2021-10-20 09:00 /tmp
drwxr-xr-x   - hdfs hadoop          0 2021-09-27 12:24 /user
```

3. Create a separate HDFS directory to copy files to, for example:

```
$ hdfs dfs -mkdir /how_to_example
```

   Usually, when using a regular user profile, HDFS returns an access error message similar to this:

   `mkdir: Permission denied: user=user_name, access=WRITE, inode="/user user_name":centos:centos:drwxr-xr-x`

   A regular user does not have privileges to write to the HDFS root directory.

4. To get around this limitation, carefully execute the required commands using the `root` and `hdfs` privileges. To do this, first obtain these privileges:

```
$ sudo -s
$ su - hdfs
```

CAUTION

It is not safe to use the `root` privileges constantly. Use them only if necessary.

5. Using the `hdfs` user privileges, repeat the creation of a new directory called *hdfsDirectory* and ensure this operation is completed successfully:

```
$ hdfs dfs -mkdir /hdfsDirectory
$ hdfs dfs -ls /
```

   The output looks similar to the following:

```
Found 7 items
-rw-r--r--   3 hdfs hadoop         20 2021-10-07 13:34 /hadoop
drwxr-xr-x   - hdfs hadoop          0 2021-11-18 13:35 /hdfsDirectory
drwxrwxrwt   - yarn hadoop          0 2021-09-15 16:58 /logs
drwxr-xr-x   - hdfs hadoop          0 2021-10-12 13:17 /staging
drwxr-xr-x   - hdfs hadoop          0 2021-09-15 16:57 /system
drwxrwxrwx   - hdfs hadoop          0 2021-10-20 09:00 /tmp
drwxr-xr-x   - hdfs hadoop          0 2021-09-27 12:24 /user
```

6. Make your regular user, for example, `admin`, the owner of the new directory:

```
$ hdfs dfs -chown admin /hdfsDirectory
$ hdfs dfs -ls /
```

   The output looks similar to this:

```
Found 7 items
-rw-r--r--   3 hdfs hadoop         20 2021-10-07 13:34 /hadoop
drwxr-xr-x   - admin hadoop          0 2021-11-18 13:35 /hdfsDirectory
drwxrwxrwt   - yarn hadoop          0 2021-09-15 16:58 /logs
drwxr-xr-x   - hdfs hadoop          0 2021-10-12 13:17 /staging
drwxr-xr-x   - hdfs hadoop          0 2021-09-15 16:57 /system
drwxrwxrwx   - hdfs hadoop          0 2021-10-20 09:00 /tmp
drwxr-xr-x   - hdfs hadoop          0 2021-09-27 12:24 /user
```

   Now your regular user is the owner of the destination directory. Exit from the `hdfs` and superuser context:

```
$ exit
$ exit
```

7. Copy *file1.txt* from the local file system to the HDFS directory and verify that the file appears in the destination HDFS directory. You can use either `put` or `copyFromLocal` command:

```
$ hdfs dfs -copyFromLocal ~/file1.txt /hdfsDirectory
$ hdfs dfs -ls /hdfsDirectory/file1.txt
```

   The output looks similar to this:

```
-rw-r--r--   3 admin hadoop         29 2021-11-18 14:01 /hdfsDirectory/file1.txt
```

8. Grant the write privilege to */hdfsDirectory* for all users if you want to allow all of them to copy to that directory:

```
$ hdfs dfs -chmod 777 /hdfsDirectory
```

   Any user that has access to HDFS can now copy their files to this directory.

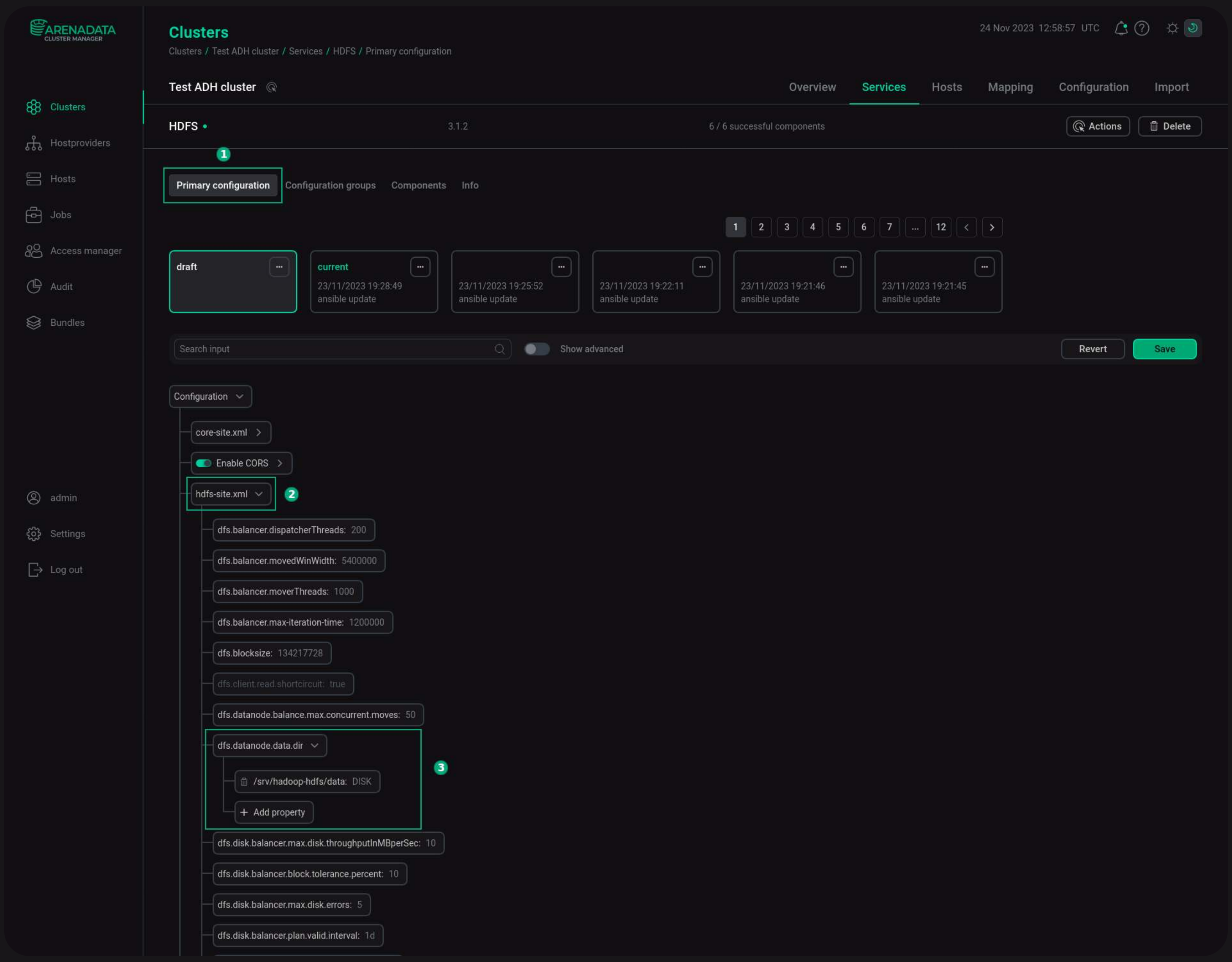
Using the previous steps, you have successfully copied the *file1.txt* file to HDFS and granted regular users the privilege to copy files to a particular HDFS directory.

Add HDFS data directories

Eugenia Kuzina

To Table of Contents

HDFS data directories are directories where DataNodes store their data blocks. These directories can be located on different devices, for example, on mounted physical disks. You can add local data directories in ADCM UI. Open the HDFS service of the ADH cluster. On the ***Primary configuration*** tab, in the configuration parameters tree, find the ***hdfs-site.xml*** node, expand it, and find the ***dfs.datanode.data.dir*** parameter among the nested elements.

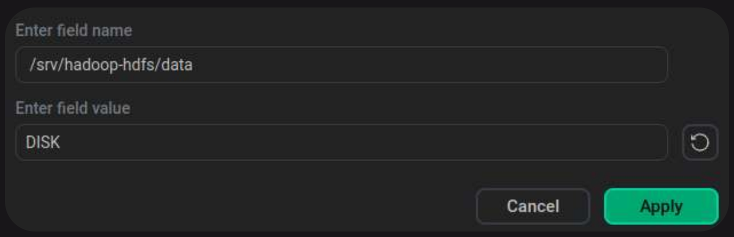

ARENADATA
CLUSTER MANAGER
Clusters
Clusters / Test ADH cluster / Services / HDFS / Primary configuration
24 Nov 2023 12:58:57 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
HDFS
3.1.2
6 / 6 successful components
Actions
Delete
1
Primary configuration
Configuration groups
Components
Info
1 2 3 4 5 6 7 ... 12
draft
current
Search input
Show advanced
Revert
Save
Configuration
core-site.xml
Enable CORS
hdfs-site.xml
2
3
Add property

current: 23/11/2023 19:28:49 ansible update
23/11/2023 19:25:52 ansible update
23/11/2023 19:22:11 ansible update
23/11/2023 19:21:46 ansible update
23/11/2023 19:21:45 ansible update

dfs.balancer.dispatcherThreads: 200
dfs.balancer.movedWinWidth: 5400000
dfs.balancer.moverThreads: 1000
dfs.balancer.max-iteration-time: 1200000
dfs.blocksize: 134217728
dfs.client.read.shortcircuit: true
dfs.datanode.balance.max.concurrent.moves: 50
dfs.datanode.data.dir
/srv/hadoop-hdfs/data: DISK
dfs.disk.balancer.max.disk.throughputInMBperSec: 10
dfs.disk.balancer.block.tolerance.percent: 10
dfs.disk.balancer.max.disk.errors: 5
dfs.disk.balancer.plan.valid.interval: 1d

Add a data directory

Click the ***Add property*** nested element to add a new data directory. In the dialog window, specify the path, for example: `/srv/hadoop-hdfs/data`, and one of the following storage types: `SSD`, `DISK`, `ARCHIVE`, or `RAM_DISK`.


Cancel
Apply

Enter field name: /srv/hadoop-hdfs/data
Enter field value: DISK

Specify HDFS data directory parameters

Click ***Apply*** to apply the settings. The item with the new data directory will be added to the configuration tree.

You also need to perform the steps below:

1. Before you add a new value to the `dfs.datanode.data.dir` parameter, you must set the owner and permissions on all DataNodes as follows:

```
$ chown -R hdfs:hadoop /path/to/data/dir
$ chmod -R 755 /path/to/data/dir
```

   Where `/path/to/data/dir` is the path to the data directory or the path in the local file system where this directory should be created.

2. Data directories must have the same path and content on all DataNodes. Different settings are allowed only for different configuration groups.

3. It is recommended to mount external data directories in the */etc/fstab* file with the `noatime` option for better performance, for example:

```
/path/to/data/dir          ext3    defaults,noatime 1 1
```

If you do not set the permissions, the HDFS fails on the next restart with NameNode error, for example: `the reported blocks 0 needs additional 478 blocks to reach the threshold 0.9990 of total blocks 479`. To fix this error, set the correct permissions for added data directories and restart HDFS. If it does not help, check that all added directories contain consistent data on different DataNodes. They should have the same set of files and directories.

DataNode hot swapping

Elena Kostyuchenko

To Table of Contents

DataNode hot swapping in HDFS is a process of changing a disk on a DataNode without shutting it down.

The following instructions describe how to configure a new disk and add it to the DataNode using CLI and ADCM.

To add a new disk to the DataNode:

1. Connect a disk to the desired host. You can check if it's visible to the system by running the `lsblk` command on the host. Possible output:

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda    253:0    0  100G  0 disk
├─vda1 253:1    0    1M  0 part
└─vda2 253:2    0  100G  0 part /
vdb    253:16   0   20G  0 disk
```

2. Create a directory for HDFS:

```
$ mkdir -p /srv/hadoop-hdfs/data1
```

3. Create a file system on the disk:

```
$ mkfs.xfs /dev/vdb
```

4. Mount the disk:

```
$ mount /dev/vdb /srv/hadoop-hdfs/data1
```

5. Add the new filesystem to the *fstab*:

```
$ echo "/dev/vdb /srv/hadoop-hdfs/data1 xfs defaults,noatime 0 0" | sudo tee --append /etc/fstab
```

6. Mount the file system:

```
$ mount -a
```

   You can check if the system has been mounted successfully using the `lsblk` command. Possible output:

```
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
vda    253:0    0  100G  0 disk
├─vda1 253:1    0    1M  0 part
└─vda2 253:2    0  100G  0 part /
vdb    253:16   0   20G  0 disk /srv/hadoop-hdfs/data1
```

7. Make `hdfs` to be the owner of the new directory and grant it permissions as follows:

```
$ chown -R hdfs:hadoop /srv/hadoop-hdfs/data1
$ chmod -R 755 /srv/hadoop-hdfs/data1
```

8. Specify the created directory in the `dfs.datanode.data.dir` parameter for the selected DataNode. You can do this manually, by editing the *hfds-site.xml* file on the DataNode host, or by creating a config group in ADCM. For more information on how to change the `dfs.datanode.data.dir` parameter value, see the Add HDFS data directories article. To learn how to create a config group, refer to the Set up configuration groups article.

> **CAUTION**
>
> Change the `dfs.datanode.data.dir` property only for the DataNode whose host has the required directory. Changing the parameter for the whole system may result in an error.

9. Start the DataNode reconfiguration by running the command:

```
$ hdfs dfsadmin -reconfig datanode <HOST>:9867 start
```

   Where `<HOST>` is the FQDN of the DataNode host. To check the status of the reconfiguration task, run:

```
$ hdfs dfsadmin -reconfig datanode <HOST>:9867 status
```

To delete a disk from the DataNode, edit the `dfs.datanode.data.dir` property on the DataNode host and run the reconfiguration command.

Remove files & directories

Elena Kostyuchenko

Contents

To Table of Contents

Trash bin

Delete files with CLI

In HDFS, you can remove files and directories using CLI, HDFS UI, or APIs. For more information on these utilities, see the Connect to HDFS article.

When deleting files in HDFS, the metadata of those files and their replicas is deleted first. If there is no metadata for file blocks on the NameNode, they are eventually deleted.

To make any changes to directories in HDFS, you must have access for writing (`w`). Otherwise, the system might return an error when trying to delete a file or create a trash directory. See more details on how to get the necessary access rights or change permissions for a directory in the Protect files in HDFS article.

Trash bin

Trash bin is a setting for deleting the data safely. If the trash option is enabled in the *core-site.xml* configuration file, then when you delete a file, the system creates a *.Trash* directory and moves the file there. In this case, the path to the file blocks is preserved in the metadata, so the file will not be deleted.

In ADH, the trash bin is enabled by default. The following parameters are responsible for managing the trash files:

- `fs.trash.interval` — time in minutes after which the files will be deleted forever. To disable the trash bin, set this parameter to `0`. The default value is `1440`.
- `fs.trash.checkpoint.interval` — the frequency of emptying the trash in minutes. The default value is `60`.

This means that every 60 minutes (the value of `fs.trash.checkpoint.interval`) the system checks how long the files have been in the trash. If any of the files are in the trash longer than 1440 minutes (the value of `fs.trash.interval`), they are deleted permanently.

Delete files with CLI

To delete a file or a directory, run one of the following commands:

- rm — removes a file or a directory.
- rmdir — removes a directory.
- rmr — removes a file or a directory recursively. This command is deprecated. It's recommended to use the `rm` command with the `R` option instead.

An example command for deleting a directory and skipping the trash bin:

```
$ hadoop fs -rm -R -skipTrash <URI>
```

Where `<URI>` is the path to a file or directory.

Enable WebHDFS

Elena Kostyuchenko

To Table of Contents

WebHDFS is a service that provides HTTP REST API access to HDFS.

In ADH, WebHDFS is enabled by default. For more information on how to connect to HDFS using HTTP REST API, see the Connect to HDFS article.

For secure clusters, configure additional parameters in hdfs-site.xml:

1. Create a new principal for the HTTP service:

```
$ addprinc -randkey HTTP/$<HOST>@$<REALM>.COM
```

   Where `<HOST>` is the FDQN or IP address of the NameNode's host, and `<REALM>` is the name of the Kerberos realm.

2. Generate the keytab for the HTTP service:

```
$ xst -norandkey -k /etc/security/spnego.service.keytab HTTP/$<HOST>
```

   Where `<HOST>` is the FDQN or IP address of the NameNode's host.

3. Make sure that the keytab is connected to the service:

```
$ klist -k -t /etc/security/spnego.service.keytab
```

4. Set the principal created earlier as the `dfs.web.authentication.kerberos.principal` parameter's value and the keytab as the `dfs.web.authentication.kerberos.keytab`. The result should look like this:

```
<property>
        <name>dfs.web.authentication.kerberos.principal</name>
        <value>HTTP/$<HOST>@$<REALM>.COM</value>
</property>
<property>
        <name>dfs.web.authentication.kerberos.keytab</name>
        <value>/etc/security/spnego.service.keytab</value>
</property>
```

5. Reboot the NameNode and DataNode services.

Check files via fsck

Elena Kostyuchenko

To Table of Contents

In HDFS, the fsck command prints out the detailed file system information, its health status, and provides several actions for damaged files.

By the means of `fsck` command, you can:

- Print out replication information for files: under replicated blocks, blocks by racks, block size, etc.
- Print out system information: health; network topology; number of DataNodes, racks, and directories, etc.
- Print out erasure coding information.
- Manage corrupted files: delete them or move to the */lost+found* directory.

The example `fsck` command that performs a check on the */tmp/* directory and additionally prints out the storage policy compliance:

```
$ hdfs fsck /tmp/ -storagepolicies
```

The output should look similar to this:

```
Connecting to namenode via http://127.0.0.1:9870/fsck?ugi=admin&storagepolicies=1&path=%2Ftmp
FSCK started by admin (auth:SIMPLE) from /127.0.0.1 for path /tmp at Wed Sep 13 12:56:43 UTC 2023

.
/tmp/test/test_file.tgz:  Under replicated BP-2122642454-127.0.0.1-1690205635727:blk_1073741974_1150. Target Replicas is 3 but found 2 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).
.
/tmp/test/test_file.zip:  Under replicated BP-2122642454-127.0.0.1-1690205635727:blk_1073741965_1141. Target Replicas is 3 but found 2 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).

/tmp/test/test_file.zip:  Under replicated BP-2122642454-127.0.0.1-1690205635727:blk_1073741966_1142. Target Replicas is 3 but found 2 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).

/tmp/test/test_file.zip:  Under replicated BP-2122642454-127.0.0.1-1690205635727:blk_1073741967_1143. Target Replicas is 3 but found 2 live replica(s), 0 decommissioned replica(s), 0 decommissioning replica(s).

Status: HEALTHY
 Number of data-nodes:  2
 Number of racks:               1
 Total dirs:                    2
 Total symlinks:                0

Replicated Blocks:
 Total size:    303063571 B
 Total files:   2
 Total blocks (validated):      4 (avg. block size 75765892 B)
 Minimally replicated blocks:   4 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       4 (100.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
 Average block replication:     2.0
 Missing blocks:                0
 Corrupt blocks:                0
 Missing replicas:              4 (33.333332 %)

Erasure Coded Block Groups:
 Total size:    0 B
 Total files:   0
 Total block groups (validated):        0
 Minimally erasure-coded block groups:  0
 Over-erasure-coded block groups:       0
 Under-erasure-coded block groups:      0
 Unsatisfactory placement block groups: 0
 Average block group size:      0.0
 Missing block groups:          0
 Corrupt block groups:          0
 Missing internal blocks:       0

Blocks satisfying the specified storage policy:
Storage Policy                  # of blocks       % of blocks
DISK:2(HOT)                        4             100.0000%

All blocks satisfy specified storage policy.
FSCK ended at Wed Sep 13 12:56:43 UTC 2023 in 7 milliseconds


The filesystem under path '/tmp' is HEALTHY
```

Local read

Elena Kostyuchenko

To Table of Contents

Local read, or the "short-circuit" read, is an HDFS setting that enables reading files directly from the source, bypassing the DataNode. This approach speeds up data reading, which can result in a substantial performance boost for your applications.

In ADH, local read is turned on by default. To be able to use it, the client must be in the same location as the data.

In hdfs-site.xml, the default local reads parameters are:

- `dfs.client.read.shortcircuit` — `True`,
- `dfs.domain.socket.path` — `/var/lib/hadoop-hdfs/dn_socket`.

If necessary, you can change these values via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Find a parameter, change its value, and click ***Apply***.
4. Confirm changes to HDFS configuration by clicking ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.

To view local reads statistics, open the ***Metrics*** page of the DataNode UI and find the ***ReadsFromLocalClient*** parameter. It indicates the number of local reads.

Use distcp

Elena Kostyuchenko

Contents

To Table of Contents

- Command overview
- Copy data between two kerberized clusters
  - Preparation steps
  - Set up trust
  - Use distcp

Command overview

The distcp command (Distributed Copy) is used to copy data. Its main advantage is that it uses MapReduce to distribute and parallel the data copying, which finds its best application when handling large amounts of data. For more information, see the MapReduce Tutorial.

Additionally, you can use `distcp` to upload, download, and update the data on object storages such as S3.

To use `distcp`, several conditions must be met:

- There must be no path collisions at the source data. Otherwise, the service will return an error. Proceed with caution when using multiple sources.
- Both the source and destination must run the same version of ADH. To copy data between clusters with different versions, use the WebHDFS protocol.
- Any operations with the files being copied should be suspended. If a client is writing to a source file or attempting to overwrite a destination file while `distcp` is running, the copy will likely fail.

The example command for copying data between clusters using the `-update` option:

```
$ hadoop distcp -update hdfs://<HOST1>:8020/tmp hdfs://<HOST2>:8020/tmp
```

Output example

```
2023-09-15 07:43:46,260 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=true,
deleteMissing=false, ignoreFailures=false, overwrite=false, append=false, useDiff=false, useRdiff=false, fromSnapshot=null,
toSnapshot=null, skipCRC=false, blocking=true, numListstatusThreads=0, maxMaps=20, mapBandwidth=0.0,
copyStrategy='uniformsize', preserveStatus=[BLOCKSIZE], atomicWorkPath=null, logPath=null, sourceFileListing=null,
sourcePaths=[hdfs://localhost:8020/tmp], targetPath=hdfs://localhost:8020/tmp, filtersFile='null', blocksPerChunk=0,
copyBufferSize=8192, verboseLog=false}, sourcePaths=[hdfs://localhost:8020/tmp], targetPathExists=true, preserveRawXattrsfalse
2023-09-15 07:43:46,429 WARN impl.MetricsConfig: Cannot locate configuration: tried hadoop-metrics2-jobtracker.
properties,hadoop-metrics2.properties
2023-09-15 07:43:46,520 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2023-09-15 07:43:46,520 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2023-09-15 07:43:46,765 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 6; dirCnt = 2
2023-09-15 07:43:46,765 INFO tools.SimpleCopyListing: Build file listing completed.
2023-09-15 07:43:46,766 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
2023-09-15 07:43:46,766 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
2023-09-15 07:43:46,793 INFO tools.DistCp: Number of paths in the copy list: 6
2023-09-15 07:43:46,804 INFO tools.DistCp: Number of paths in the copy list: 6
2023-09-15 07:43:46,812 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2023-09-15 07:43:46,884 INFO mapreduce.JobSubmitter: number of splits:1
2023-09-15 07:43:47,324 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1206860640_0001
2023-09-15 07:43:47,326 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-09-15 07:43:47,459 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2023-09-15 07:43:47,459 INFO tools.DistCp: DistCp job-id: job_local1206860640_0001
2023-09-15 07:43:47,460 INFO mapreduce.Job: Running job: job_local1206860640_0001
2023-09-15 07:43:47,462 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2023-09-15 07:43:47,468 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-09-15 07:43:47,468 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2023-09-15 07:43:47,469 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.tools.mapred.CopyCommitter
2023-09-15 07:43:47,501 INFO mapred.LocalJobRunner: Waiting for map tasks
2023-09-15 07:43:47,502 INFO mapred.LocalJobRunner: Starting task: attempt_local1206860640_0001_m_000000_0
2023-09-15 07:43:47,525 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-09-15 07:43:47,525 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2023-09-15 07:43:47,588 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2023-09-15 07:43:47,591 INFO mapred.MapTask: Processing split: file:/tmp/hadoop/mapred/staging/admin1599424582/.
staging/_distcp15296078/fileList.seq:0+1192
2023-09-15 07:43:47,597 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2023-09-15 07:43:47,597 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output
directory:false, ignore cleanup failures: false
2023-09-15 07:43:47,605 INFO mapred.CopyMapper: Copying hdfs://localhost:8020/tmp to hdfs://localhost:8020/tmp/tmp
2023-09-15 07:43:47,640 INFO mapred.CopyMapper: Copying hdfs://localhost:8020/tmp/test/hadoop-jvm.tgz to
hdfs://localhost:8020/tmp/tmp/test/hadoop-jvm.tgz
2023-09-15 07:43:47,652 INFO mapred.RetriableFileCopyCommand: Creating temp file: hdfs://localhost:8020/tmp/.distcp.tmp.
attempt_local1206860640_0001_m_000000_0
2023-09-15 07:43:48,462 INFO mapreduce.Job: Job job_local1206860640_0001 running in uber mode : false
2023-09-15 07:43:48,463 INFO mapreduce.Job:  map 0% reduce 0%
2023-09-15 07:43:48,698 INFO mapred.CopyMapper: Copying hdfs://localhost:8020/tmp/test to hdfs://localhost:8020/tmp/tmp/test
2023-09-15 07:43:48,708 INFO mapred.CopyMapper: Copying hdfs://localhost:8020/tmp/test/pom.xml to
hdfs://localhost:8020/tmp/tmp/test/pom.xml
2023-09-15 07:43:48,717 INFO mapred.RetriableFileCopyCommand: Creating temp file: hdfs://localhost:8020/tmp/.distcp.tmp.
attempt_local1206860640_0001_m_000000_0
2023-09-15 07:43:48,900 INFO mapred.CopyMapper: Copying hdfs://localhost:8020/tmp/test/03_02.mov to
hdfs://localhost:8020/tmp/tmp/test/03_02.mov
2023-09-15 07:43:48,909 INFO mapred.RetriableFileCopyCommand: Creating temp file: hdfs://localhost:8020/tmp/.distcp.tmp.
attempt_local1206860640_0001_m_000000_0
2023-09-15 07:43:59,535 INFO mapred.LocalJobRunner: 66.2% Copying hdfs://localhost:8020/tmp/test/03_02.mov to
hdfs://localhost:8020/tmp/tmp/test/03_02.mov [674.9M/1018.9M] > map
2023-09-15 07:44:00,470 INFO mapreduce.Job:  map 82% reduce 0%
2023-09-15 07:44:05,536 INFO mapred.LocalJobRunner: 100.0% Copying hdfs://localhost:8020/tmp/test/03_02.mov to
hdfs://localhost:8020/tmp/tmp/test/03_02.mov [1018.9M/1018.9M] > map
2023-09-15 07:44:07,014 INFO mapred.CopyMapper: Copying hdfs://localhost:8020/tmp/test/test_file.zip to
hdfs://localhost:8020/tmp/tmp/test/test_file.zip
2023-09-15 07:44:07,023 INFO mapred.RetriableFileCopyCommand: Creating temp file: hdfs://localhost:8020/tmp/.distcp.tmp.
attempt_local1206860640_0001_m_000000_0
2023-09-15 07:44:11,537 INFO mapred.LocalJobRunner: 88.6% Copying hdfs://localhost:8020/tmp/test/test_file.zip to
hdfs://localhost:8020/tmp/tmp/test/test_file.zip [256.1M/289.0M] > map
2023-09-15 07:44:12,412 INFO mapred.LocalJobRunner: 88.6% Copying hdfs://localhost:8020/tmp/test/test_file.zip to
hdfs://localhost:8020/tmp/tmp/test/test_file.zip [256.1M/289.0M] > map
2023-09-15 07:44:12,412 INFO mapred.Task: Task:attempt_local1206860640_0001_m_000000_0 is done. And is in the process of
committing
2023-09-15 07:44:12,413 INFO mapred.LocalJobRunner: 88.6% Copying hdfs://localhost:8020/tmp/test/test_file.zip to
hdfs://localhost:8020/tmp/tmp/test/test_file.zip [256.1M/289.0M] > map
2023-09-15 07:44:12,414 INFO mapred.Task: Task attempt_local1206860640_0001_m_000000_0 is allowed to commit now
2023-09-15 07:44:12,415 INFO output.FileOutputCommitter: Saved output of task 'attempt_local1206860640_0001_m_000000_0' to
file:/tmp/hadoop/mapred/staging/admin1599424582/.staging/_distcp15296078/_logs
2023-09-15 07:44:12,416 INFO mapred.LocalJobRunner: 100.0% Copying hdfs://localhost:8020/tmp/test/test_file.zip to
hdfs://localhost:8020/tmp/tmp/test/test_file.zip [289.0M/289.0M]
2023-09-15 07:44:12,417 INFO mapred.Task: Task 'attempt_local1206860640_0001_m_000000_0' done.
2023-09-15 07:44:12,424 INFO mapred.Task: Final Counters for attempt_local1206860640_0001_m_000000_0: Counters: 25
        File System Counters
                FILE: Number of bytes read=191251
                FILE: Number of bytes written=697086
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1371500500
                HDFS: Number of bytes written=1371500500
                HDFS: Number of read operations=48
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=15
        Map-Reduce Framework
                Map input records=6
                Map output records=0
                Input split bytes=150
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=41
                Total committed heap usage (bytes)=326631424
        File Input Format Counters
                Bytes Read=1232
        File Output Format Counters
                Bytes Written=8
        DistCp Counters
                Bandwidth in Btyes=57145854
                Bytes Copied=1371500500
                Bytes Expected=1371500500
                Files Copied=4
                DIR_COPY=2
2023-09-15 07:44:12,424 INFO mapred.LocalJobRunner: Finishing task: attempt_local1206860640_0001_m_000000_0
2023-09-15 07:44:12,425 INFO mapred.LocalJobRunner: map task executor complete.
2023-09-15 07:44:12,436 INFO mapred.CopyCommitter: About to preserve attributes: B
2023-09-15 07:44:12,441 INFO mapred.CopyCommitter: Preserved status on 0 dir entries on target
2023-09-15 07:44:12,441 INFO mapred.CopyCommitter: Cleaning up temporary work folder:
file:/tmp/hadoop/mapred/staging/admin1599424582/.staging/_distcp15296078
2023-09-15 07:44:12,475 INFO mapreduce.Job:  map 100% reduce 0%
2023-09-15 07:44:12,476 INFO mapreduce.Job: Job job_local1206860640_0001 completed successfully
2023-09-15 07:44:12,482 INFO mapreduce.Job: Counters: 25
        File System Counters
                FILE: Number of bytes read=191251
                FILE: Number of bytes written=697086
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1371500500
                HDFS: Number of bytes written=1371500500
                HDFS: Number of read operations=48
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=15
        Map-Reduce Framework
                Map input records=6
                Map output records=0
                Input split bytes=150
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=41
                Total committed heap usage (bytes)=326631424
        File Input Format Counters
                Bytes Read=1232
        File Output Format Counters
                Bytes Written=8
        DistCp Counters
                Bandwidth in Btyes=57145854
                Bytes Copied=1371500500
                Bytes Expected=1371500500
                Files Copied=4
                DIR_COPY=2
```

Copy data between two kerberized clusters

To run `distcp` in a cluster from a different Kerberos realm, you need to establish trust between the clusters. Usually, it is done manually by editing the Kerberos configuration files. For more information on how to set up a trust between clusters manually, see Setting up a Realm Trust.

In this example, the trust is configured via ADCM, and the `distcp` is called using the ***Additional nameservices*** parameter. See more details about the Internal nameservice concept in the HDFS service management via ADCM article.

Preparation steps

It is assumed that Kerberos is already installed and configured. To learn more about how to install Kerberos, see Kerberos server settings.

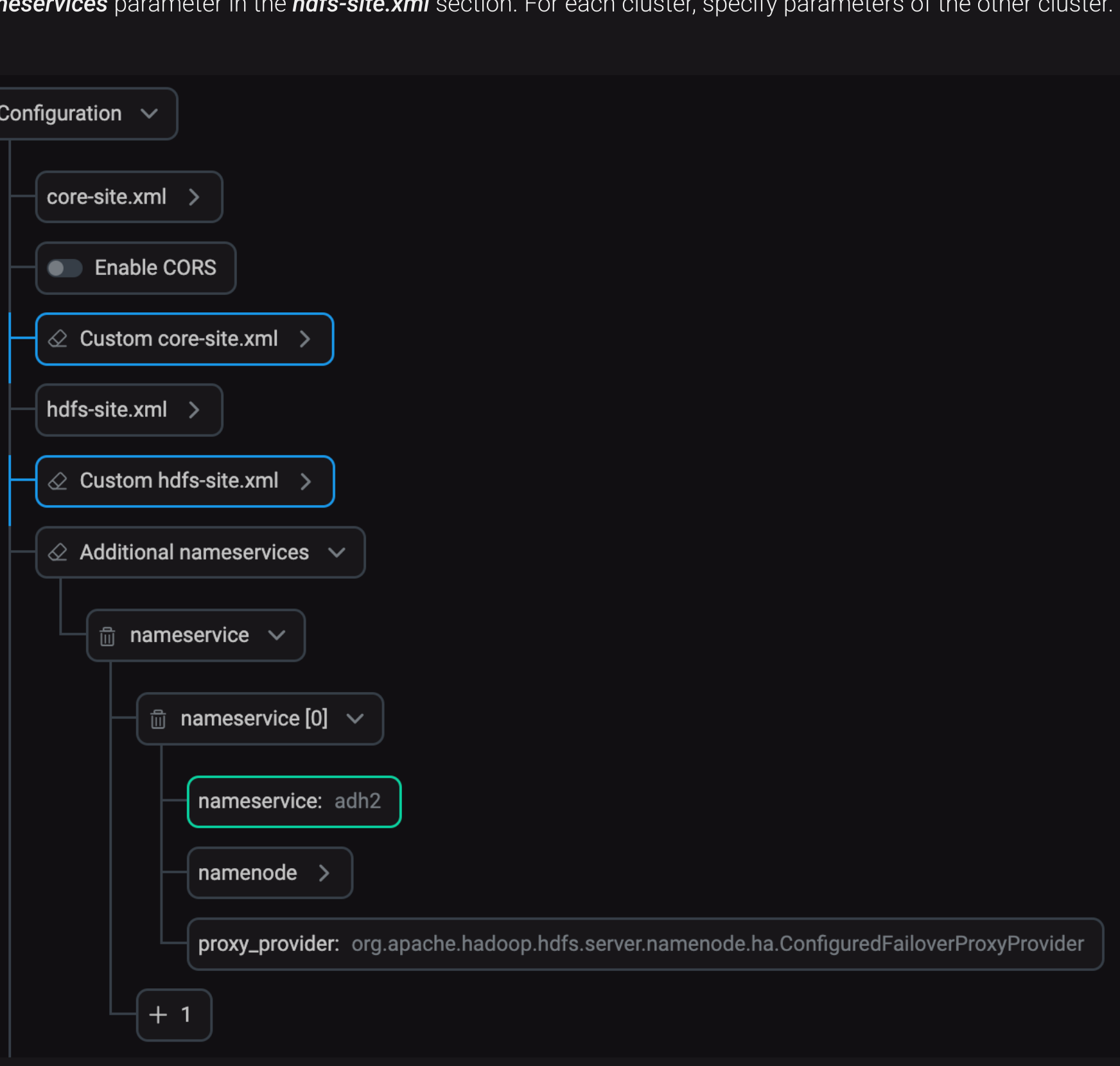
The following information is required to perform the next steps:

- cluster realms;
- cluster nameservices (you can find the cluster nameservice in the `dfs.internal.nameservices` parameter on the HDFS configuration page);
- Kerberos server FQDN or IP;
- NameNodes server FQDN and IP;
- NameNodes ID (you can find the ID in the NameNode UI: `http://<NameNode IP or FQDN>:9870/`);
- Kerberos admin account credentials.

Set up trust

To set up trust between two Kerberos servers via ADCM, perform these actions for both clusters:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at HDFS.
3. Find and fill in the ***Additional nameservices*** parameter in the ***hdfs-site.xml*** section. For each cluster, specify parameters of the other cluster.


Configuration
core-site.xml
Enable CORS
Custom core-site.xml
hdfs-site.xml
Custom hdfs-site.xml
Additional nameservices
nameservice
nameservice [0]
namenode
+ 1

nameservice: adh2

proxy_provider: org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider

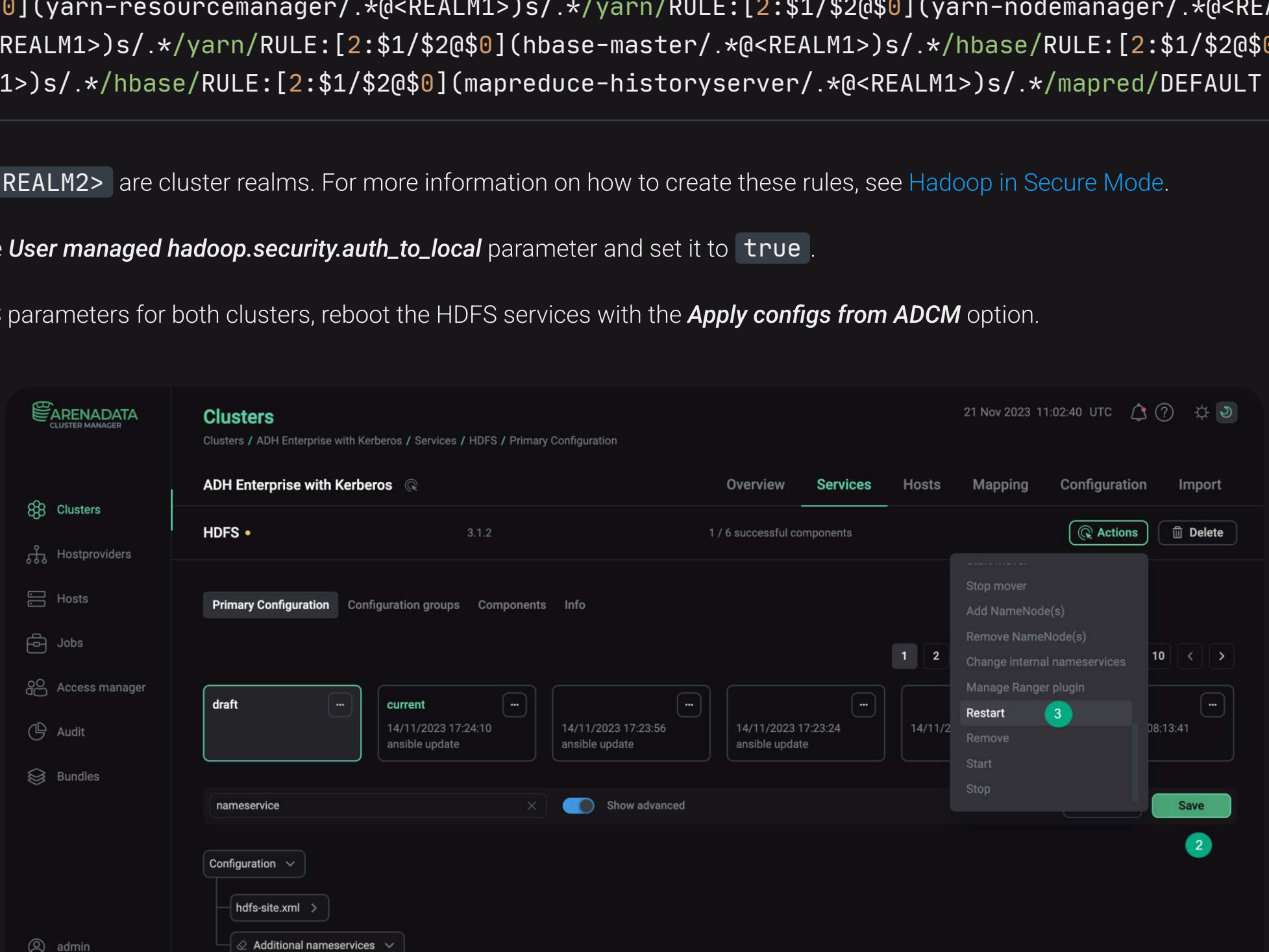
Additional nameservices parameter in ADCM

4. In the ***hadoop.security.auth_to_local*** HDFS parameter, paste the following rules that map users of the first cluster to users of the second one:

```
RULE:[2:$1/$2@$0](hdfs-namenode/.*@<REALM2>)s/.*/hdfs/RULE:[2:$1/$2@$0](hdfs-datanode/.*@<REALM2>)s/.*/hdfs/RULE:[2:$1/$2@$0](hdfs-journalnode/.*@<REALM2>)s/.*/hdfs/RULE:[2:$1/$2@$0](hdfs/.*@<REALM2>)s/.*/hdfs/RULE:[2:$1/$2@$0](yarn-resourcemanager/.*@<REALM2>)s/.*/yarn/RULE:[2:$1/$2@$0](yarn-nodemanager/.*@<REALM2>)s/.*/yarn/RULE:[2:$1/$2@$0](yarn/.*@<REALM2>)s/.*/yarn/RULE:[2:$1/$2@$0](hbase-master/.*@<REALM2>)s/.*/hbase/RULE:[2:$1/$2@$0](hbase-regionserver/.*@<REALM2>)s/.*/hbase/RULE:[2:$1/$2@$0](mapreduce-historyserver/.*@<REALM2>)s/.*/mapred/RULE:[2:$1/$2@$0](hdfs-namenode/.*@<REALM1>)s/.*/hdfs/RULE:[2:$1/$2@$0](hdfs-datanode/.*@<REALM1>)s/.*/hdfs/RULE:[2:$1/$2@$0](hdfs-journalnode/.*@<REALM1>)s/.*/hdfs/RULE:[2:$1/$2@$0](hdfs/.*@<REALM1>)s/.*/hdfs/RULE:[2:$1/$2@$0](yarn-resourcemanager/.*@<REALM1>)s/.*/yarn/RULE:[2:$1/$2@$0](yarn-nodemanager/.*@<REALM1>)s/.*/yarn/RULE:[2:$1/$2@$0](yarn/.*@<REALM1>)s/.*/yarn/RULE:[2:$1/$2@$0](hbase-master/.*@<REALM1>)s/.*/hbase/RULE:[2:$1/$2@$0](hbase-regionserver/.*@<REALM1>)s/.*/hbase/RULE:[2:$1/$2@$0](mapreduce-historyserver/.*@<REALM1>)s/.*/mapred/DEFAULT
```

   Where `<REALM1>` and `<REALM2>` are cluster realms. For more information on how to create these rules, see Hadoop in Secure Mode.

5. On the same page, find the ***User managed hadoop.security.auth_to_local*** parameter and set it to `true`.
6. After you configured HDFS parameters for both clusters, reboot the HDFS services with the ***Apply configs from ADCM*** option.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / ADH Enterprise with Kerberos / Services / HDFS / Primary Configuration
21 Nov 2023 11:02:40 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
HDFS
3.1.2
1 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
Stop mover
Add NameNode(s)
Remove NameNode(s)
Change internal nameservices
Manage Ranger plugin
Restart
Remove
Start
Stop
draft
current
14/11/2023 17:24:10
ansible update
14/11/2023 17:23:56
ansible update
14/11/2023 17:23:24
ansible update
nameservice
Show advanced
Save
Configuration
hdfs-site.xml
Additional nameservices
nameservice
1
2
3

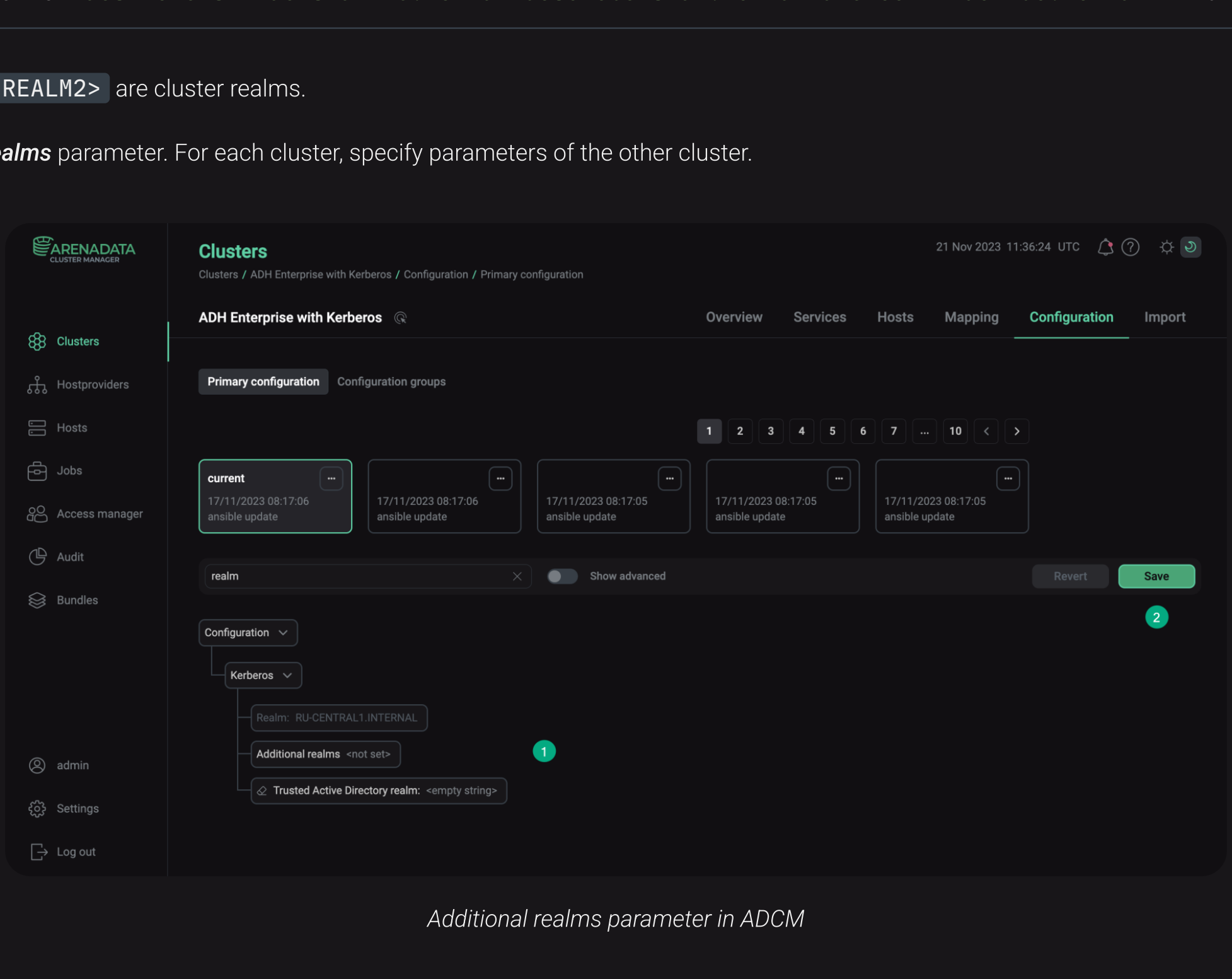
Configure HDFS for Kerberos trust

7. On both Kerberos servers, create a common principal by running the following commands:

```
$ kadmin.local addprinc -e "aes128-cts-hmac-sha1-96:normal des3-cbc-sha1:normal arcfour-hmac-md5:normal" krbtgt/<REALM1>@<REALM2>
$ kadmin.local addprinc -e "aes128-cts-hmac-sha1-96:normal des3-cbc-sha1:normal arcfour-hmac-md5:normal" krbtgt/<REALM2>@<REALM1>
```

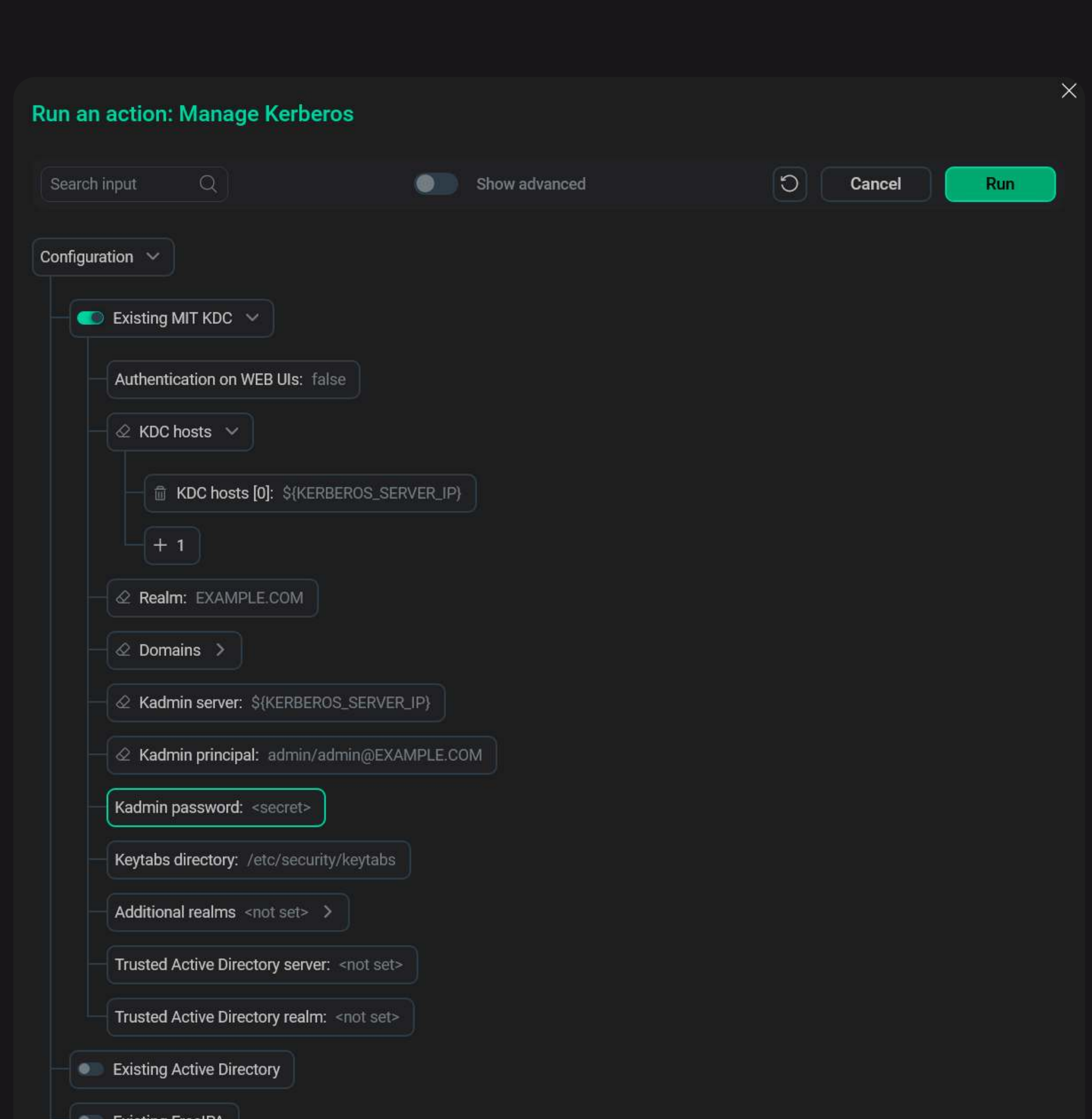
   Where `<REALM1>` and `<REALM2>` are cluster realms.

8. Configure the ***Additional realms*** parameter. For each cluster, specify parameters of the other cluster.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / ADH Enterprise with Kerberos / Configuration / Primary configuration
21 Nov 2023 11:36:24 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
Primary configuration
Configuration groups
current
17/11/2023 08:17:06
ansible update
17/11/2023 08:17:05
ansible update
realm
Show advanced
Revert
Save
Configuration
Kerberos
Realm: RU-CENTRAL1.INTERNAL
Additional realms <not set>
Trusted Active Directory realm: <empty string>
1
2

Additional realms parameter in ADCM

9. Enable Kerberos on both clusters. Check the ***Advanced*** flag, then click ***Existing MIT KDC***, and fill in the parameters:
   - KDC hosts;
   - Realm;
   - Kadmin server;
   - Kadmin principal;
   - Kadmin password.


Run an action: Manage Kerberos
Search input
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
KDC hosts
+ 1
Domains
Existing Active Directory
Existing FreeIPA
Custom kerberization settings

Authentication on WEB UIs: false

KDC hosts [0]: ${KERBEROS_SERVER_IP}

Realm: EXAMPLE.COM

Kadmin server: ${KERBEROS_SERVER_IP}

Kadmin principal: admin/admin@EXAMPLE.COM

Kadmin password: <secret>

Keytabs directory: /etc/security/keytabs

Additional realms <not set>

Trusted Active Directory server: <not set>

Trusted Active Directory realm: <not set>

Manage Kerberos action window

Use distcp

1. Connect to a cluster's host via SSH and log in as the HDFS principal:

```
$ kinit hdfs
```

2. Run the `distcp` command:

```
$ hadoop distcp hdfs://<nameservice1>:8020/<source directory> hdfs://<nameservice2>:8020/<target directory>
```

   Where:

   - `<nameservice1>` — the cluster nameservice from which you want to copy the data;
   - `<nameservice2>` — the cluster nameservice to which the data must be copied;
   - `<source directory>` — the directory of the data you want to copy;
   - `<target directory>` — the directory in which to store the data.

Use HttpFS

Elena Kostyuchenko

To Table of Contents

HDFS provides a built-in HttpFS server, which can be used to transfer data between clusters running different Hadoop versions, access data through firewall, use HTTP utilities and more.

In Arenadata Hyperwave, HttpFS server comes preinstalled and available as a HDFS UI. For more information, see HDFS UI overview.

You can look up the default HttpFS server settings in the following sections on the HDFS configuration page: ***httpfs-site.xml***, ***httpfs-env.sh***, and ***HDFS Heap Memory settings***.

To find these parameters in ADCM, go to the ***Clusters*** page, select the desired cluster, navigate to the ***Services*** tab, and click at HDFS.

An example command that calls the `ls` function for the *user/tmp* directory as the `admin` user:

```
$ curl 'http://<httpfs-host>:14000/webhdfs/v1/user/tmp?op=LISTSTATUS&user.name=admin'
```

Where `<httpfs-host>` is the IP address of the host with installed HttpFS server.

Storage policies

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Storage types

Storage policies

Commands

Overview

Storage policies are a set of rules that help improve performance of applications by optimizing the data distribution based on demand for computing power and data accessibility. For example, if some data is used more often than the other, you can move it to an SSD disk to improve performance, and move the less popular data to the archive to save the resources.

If a storage policy is specified for a directory, HDFS will store at least one replica of each file in that directory in the designated storage type. If there's not enough space, HDFS will store the new data in the **Fallback storages for creation** and replicas will be stored in the **Fallback storages for replication**.

Storage types

Depending on the selected policy, different storage types are available:

- **ARCHIVE** — a storage with low computational resources but high capacity;
- **DISK** — the default storage type with average resource for compute and capacity;
- **SSD** — a storage hosted on SSD drives;
- **RAM_DISK** — DataNode memory;
- **PROVIDED** — a storage outside of the HDFS system. For more information on Providede storage, see the HDFS Provided Storage article.

Before setting a storage policy, make sure that the required type of storage exists. See more details on how to create a specific type of storage in the Add HDFS data directories article.

Storage policies

In HDFS, the following policies exist:

- **Hot**. This policy is fitting for data regularly used for processing. All data blocks are stored in DISK when there's enough space, and, if DISK space is limited, blocks will be stored in ARCHIVE.
- **Cold**. The Cold storage policy is applied to rarely used data. All blocks are stored in ARCHIVE.
- **Warm**. When the data is used only occasionally, you can set the Warm storage policy. In this case, only one replica is stored on DISK and the rest replicas are stored in ARCHIVE.
- **All_SSD**. This storage policy is applied for storing all data on SSD. If there's no space on SSD, the data is stored on DISK.
- **One_SSD**. According to this storage policy, only one of the replicas is stored on the SSD. The remaining replicas are stored in DISK.
- **Lazy_Persist**. When the Lazy_Persist storage policy is used, one replica is always stored in memory. This policy is useful only for single replica blocks. For blocks with more than one replica, all the replicas will be written to DISK.
- **Provided**. This storage policy is applied for storing one replica in the PROVIDED type of storage and the rest replicas are stored in DISK.

Policy name	ID	Block placement	Fallback storages for creation	Fallback storages for replication
Hot	7	All replicas are stored in DISK	—	ARCHIVE
Cold	2	All replicas are stored in ARCHIVE	—	—
Warm	5	One replica is stored in DISK and the rest in ARCHIVE	ARCHIVE, DISK	ARCHIVE, DISK
All_SSD	12	All replicas are stored in SSD	DISK	DISK
One_SSD	10	One replica is stored in SSD and the rest in DISK	SSD, DISK	SSD, DISK
Lazy_Persist	15	One replica is stored in RAM_DISK. If there's more than one replica, all of the replicas are stored in DISK	DISK	DISK
Provided	1	One replica is stored in PROVIDED and the rest in DISK	PROVIDED, DISK	PROVIDED, DISK

IMPORTANT

For the striped erasure coded files, the only suitable storage policies are All_SSD, Hot, or Cold. If you set a different policy, it will not take effect.

Commands

For managing storage policies, use the storagepolicies command.

The example of using the command to check the assigned storage policy:

```
$ hdfs storagepolicies -getStoragePolicy -path /tmp
```

When you set a new policy, it's not applied automatically. To make sure that block placement complies with the storage policies, use the Mover action.

The output:

```
The storage policy of /tmp:
BlockStoragePolicy{HOT:7, storageTypes=[DISK], creationFallbacks=[], replicationFallbacks=[ARCHIVE]}
```

Use quotas

Elena Kostyuchenko

Contents

To Table of Contents

Name quotas

Space quotas

Storage type quotas

In HDFS, the administrator can limit the size and population of directories by using quotas. There are three types of quotas that can be set:

- Name quotas — limit the amount of names in the directory.
- Space quotas — limit the disk space used by the directory.
- Storage type quotas — limit the disk space of a specific type of storage.

All types of quotas work independently, which means you can configure all three types of quotas at the same time.

The configured quotas are written into the *fsimage*. When a NameNode starts, it prints out any existing quota violations. Setting or removing a quota creates a journal entry in the EditLogs.

Name quotas

The name quota is a limitation on the number of names in a directory, including the directory itself. If a new file or directory is created or renamed inside the quoted catalog, it will count to the quota.

When the quota has been filled, any file creation or rename operations will fail.

To set a name quota for a directory, use the command:

```
$ hdfs dfsadmin -setQuota <N> <directory>...<directory>
```

Where:

- `<N>` — a positive integer representing the number of names allowed in the directory;
- `<directory>` — a path to the directory for which to set the name quota.

To remove the name quota for a directory, use the command:

```
$ hdfs dfsadmin -clrQuota <directory>...<directory>
```

Where `<directory>` is the path to the directory for which to remove the quotas.

Space quotas

The space quota is a limitation on the directory size. The quota is measured in blocks, including their replicas. This means that changing the replication rate will affect the quota. For example, 1 GB of data with replication rate of 3 consumes 3 GB of quota.

Since there's no block creation, the following does not count against the quota:

- creating a directory;
- creating a file (if the quota is full, the file will be created but it's going to be empty);
- hosting the metadata.

To set the space quota for a directory, use the command:

```
$ hdfs dfsadmin -setSpaceQuota <N> <directory>...<directory>
```

Where:

- `<N>` — a positive integer representing the size limit for the directory in bytes. You can specify size with a binary prefix for convenience, for example: 50g for 50 GB and 2t for 2 TB etc.
- `<directory>` — the path to the directory for which to set the space quota.

To remove the space quota for a directory, use the command:

```
$ hdfs dfsadmin -clrSpaceQuota <directory>...<directory>
```

Where `<directory>` is the path to the directory for which to remove the space quota.

Storage type quotas

The storage type quotas limit how much of a specific disk space can be used by a directory. When counting the occupied space, this type of quotas work similarly to the space quotas.

Before setting a storage type quota, make sure you have configured the storage policy for the directory. For information on storage policies, refer to Storage policies.

To set the storage type quota for a directory, use the command:

```
$ hdfs dfsadmin -setSpaceQuota <N> -storageType <storagetype> <directory>...<directory>
```

Where:

- `<N>` — a positive integer representing the space limit of the specified storage type for the directory in bytes. This limit applies to all the files under the directory tree. You can specify size with a binary prefix for convenience, for example: 50g for 50 GB and 2t for 2 TB etc.
- `<storagetype>` — the type of storage to put the limit on. Available storage type values are `RAM_DISK`, `DISK`, `SSD`, `ARCHIVE`.
- `<directory>` — the path to the directory for which to set the storage type quota.

To remove the storage type quota for a directory, use the command:

```
$ hdfs dfsadmin -clrSpaceQuota -storageType <storagetype> <directory>...<directory>
```

- `<storagetype>` — the type of storage to take the limit off. Available storage type values are `RAM_DISK`, `DISK`, `SSD`, `ARCHIVE`.
- `<directory>` — the path to the directory for which to remove the storage type quota.

Cache management

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Cache directives and cache pools

Configuration

Data flow

Commands

Overview

In HDFS, you can configure the **centralized cache management** setting, which enables deliberate storage of a specific directory in cache memory. This feature ensures fast access to data and speeds up tasks related to the cached data.

Without this feature, there would be no control over what data is cached. Low-priority workloads could wipe out cached high-priority jobs; and applications that make task placement decisions (such as MapReduce or Impala) would not be able to take advantage of the cached data.

Centralized cache exposes cluster's cache state to task schedulers and makes it possible for clients to pin their data in cluster memory. Additionally, applications could potentially do efficient zero-copy reads of cached data, and could skip verifying checksums of already verified cached data.

Having the centrilized cache configured, benifits the performance of several Hyperwave services, including HBase, Impala, and Hive.

Cache directives and cache pools

There are two main concepts used in cache management: **cache directives** and **cache pools**.

A **cache directive** is a path that a user or an application requests to be stored in cache. This can be either a file or a directory.

Caching is non-recursive. Which means that if a file is specified, all of it's blocks will be stored in cache. But if a directory is specified, only the files at the first level of that directory will be cached.

If a directive contains a symlink, the symlink target will not be cached.

When creating a cache directive, you will need to specify two parameters:

- **Cache replication factor** — the number of block replicas to cache. If multiple cache directives refer to the same file, the maximum cache replication factor is applied.
- **Expiration time (time-to-live, TTL)** — a relative expiration time in the future. After a cache directive expires, it is no longer considered by the NameNode when making caching decisions.

After receiving a directive, the NameNode will be checking periodically if files in the directive directory have been changed. If so, the NameNode will instruct the DataNodes to cache the new blocks.

Cache pool is an administrative and resource management tool. When a cache directive is created, it must belong to a specific cache pool.

For each cache pool you can configure the following:

- UNIX-like permissions — a code that indicates which users and groups have access to the pool. Write permissions allow users to add and remove cache directives, read permissions allow to see the list of directives in a pool and additional metadata.
- Maximum size — a size limit in bytes for all cache directives in a pool. Typically, the sum of all pool limits will approximately equal the amount of aggregate memory reserved for HDFS caching on the cluster.
- Maximum TTL — the maximum time-to-live for cache directives in a pool.

Configuration

To enable cache management, you need to set a value other than zero for the `dfs.datanode.max.locked.memory` parameter in *hdfs-site.xml*. This parameter describes how much of the DataNode's memory can be reserved for caching.

When setting this parameter, take into the account:

- In Unix-like systems, the locked-in-memory size ulimit (`ulimit -l`) of the DataNode user needs to be increased to match this parameter (see possible errors section below).
- The DataNode, application JVM heaps, and the operating system page cache also require space.
- The `dfs.datanode.max.locked.memory` setting is shared with the Lazy Persist Writes feature.

Possible errors: Linux ulimit configuration

One of the possible errors when configuring the centralized cache is:

```
Cannot start datanode because the configured max locked memory size… is more than the datanode's available RLIMIT_MEMLOCK ulimit.
```

It means that the memory limit configured on the OS level is lower than the limit you have set for the DataNode. To fix this, adjust the `ulimit -l` value that the DataNode runs with.

To make sure you have the right configuration, run:

```
$ ulimit -l
```

If the output value is bigger than the `dfs.datanode.max.locked.memory` value or is `unlimited`, the configuration is correct.

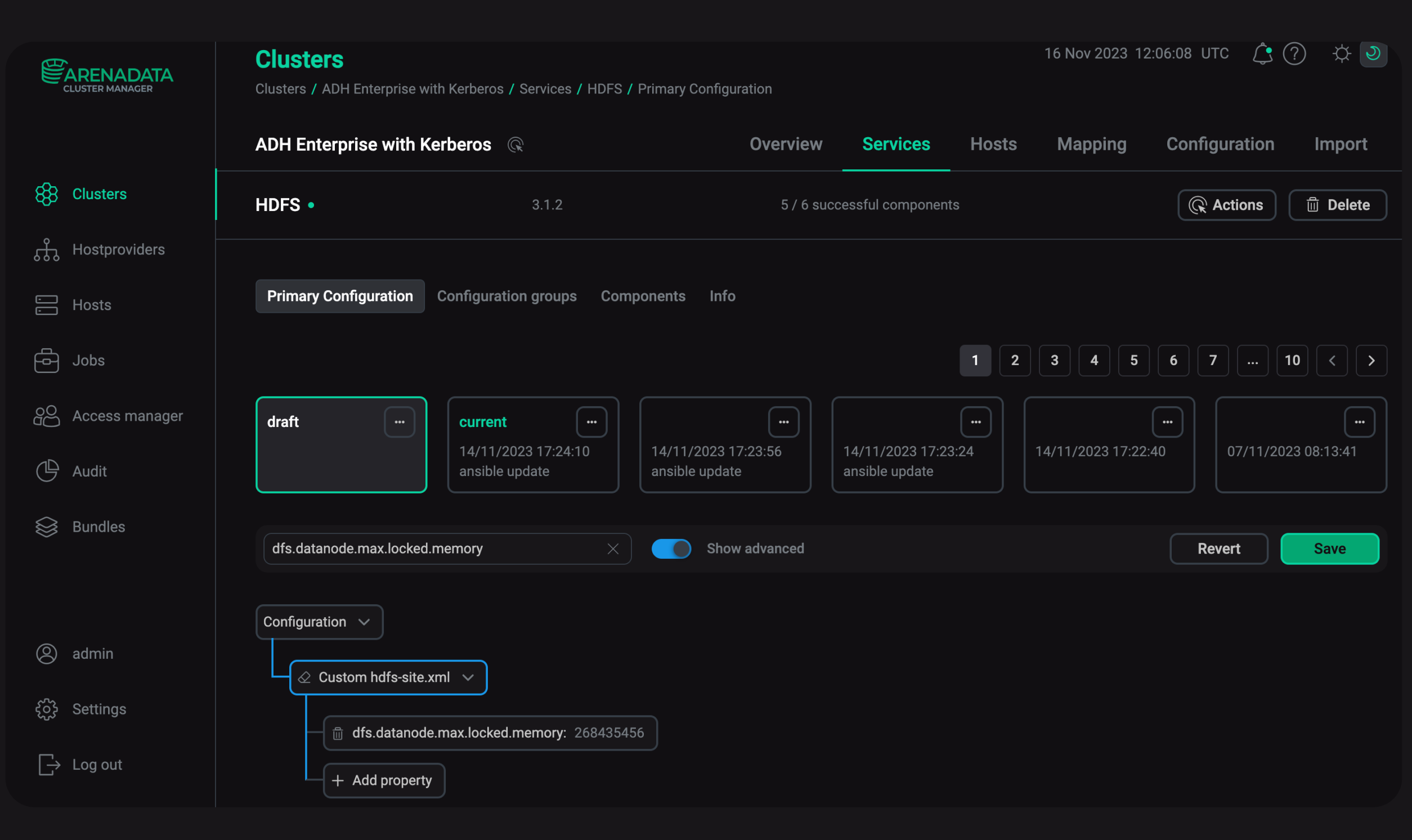
Note that this command outputs the memory lock limit in KB, but the value of `dfs.datanode.max.locked.memory` is in bytes.

By default, centralized cache is not available in ADH, but you can configure it using the custom configuration parameter in ADCM.

Before enabling the cache management, make sure you have the *libhadoop.so* native library available for the DataNodes. For more information on how to install the *libhadoop.so* library, see the Native Libraries Guide.

To turn on the centralized cache via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at HDFS.
3. Toggle the ***Show advanced*** option and find ***Custom hdfs-site.xml***.
4. Open the parameter drop-down list and select ***Add property***.
5. Enter the parameter name `dfs.datanode.max.locked.memory` and the desired value in bytes.
6. Click ***Apply*** and confirm changes to HDFS configuration by clicking ***Save***.
7. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.



dfs.datanode.max.locked.memory

dfs.datanode.max.locked.memory: 268435456

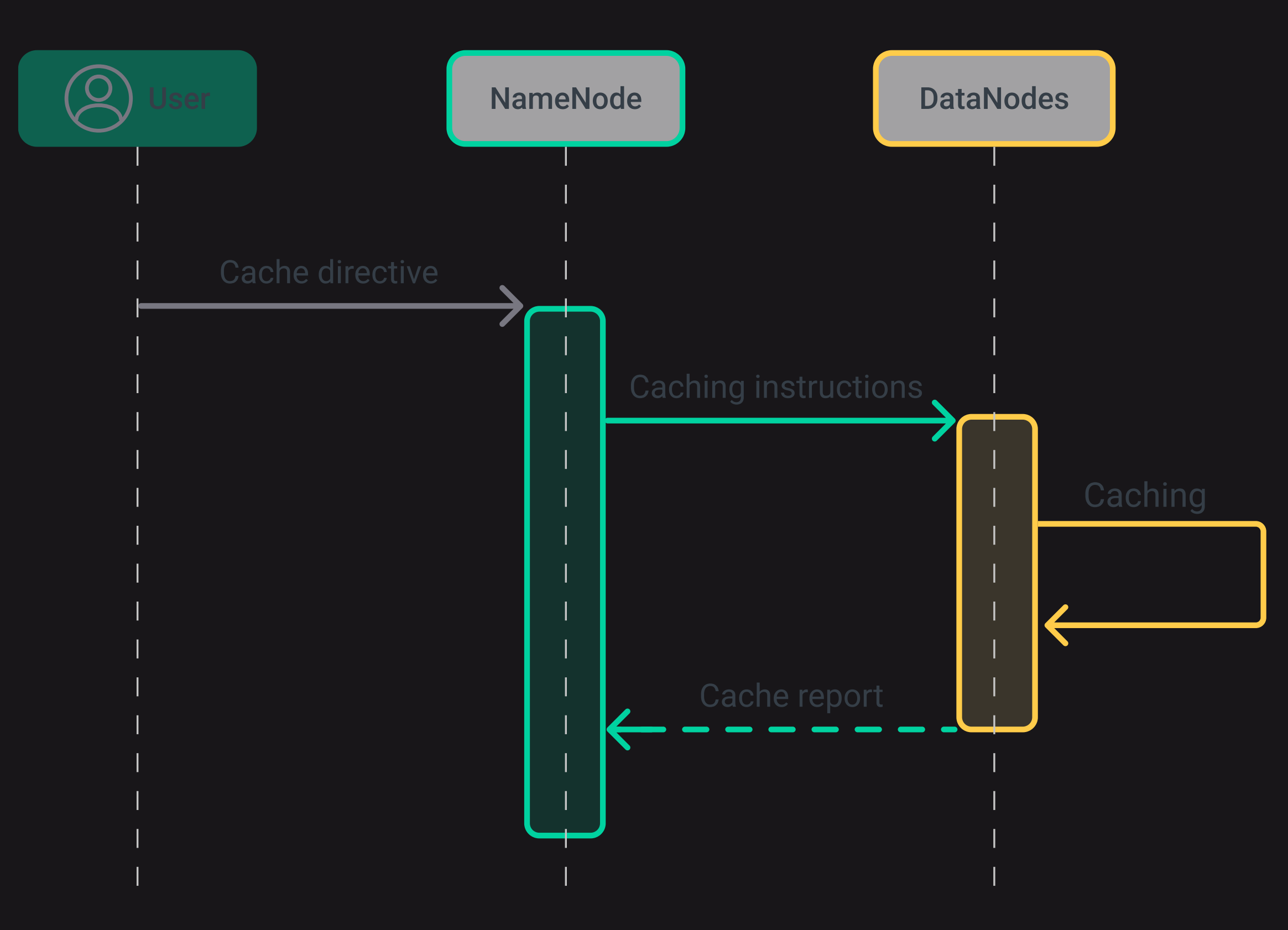
Custom hdfs-site.xml parameter in ADCM

You can configure cache settings by adding other parameters the same way. See optional cache parameters in the table below.

Optional cache parameters

Parameter	Description	Default value
dfs.namenode.path.based.cache.refresh.interval.ms	The parameter describes how often the NameNode checks if there are any blocks that should be cached, and on what DataNodes. Specified milliseconds	30000
dfs.datanode.fsdatasetcache.max.threads.per.volume	The maximum number of threads to use for caching new data on the DataNode. These threads consume computational power, which can affect normal DataNode operations	4
dfs.cachereport.intervalMsec	The parameter describes how often a DataNode sends cache reports to the NameNode. Specified milliseconds	10000
dfs.namenode.path.based.cache.block.map.allocation.percent	The percentage of the Java heap which will be reserved for the cached blocks map. Smaller maps may be accessed more slowly if the number of cached blocks is large; larger maps will consume more memory. Specified as a decimal	0.25

Data flow



Sequence diagram of a caching request

In HDFS, centralized cashe works similarly to storing regular data blocks: cache is managed by the NameNode and stored on DataNodes.

When a client creates a cache directive, the active NameNode checks with the DataNodes that have the desired blocks, and instructs them to cache these blocks.

At the same time, the NameNode makes an entry about the new directive in the edit log. This allows the NameNode to re-cache data on restart.

DataNodes pin the requested blocks into memory and send cache reports to all existing NameNodes. The NameNode updates its map of cached blocks and sends instructions to DataNodes on which replicas to cache and uncache.

Commands

An administrator can manage cache directives and pools using the cacheadmin command.

To create a new directive in a new cache pool:

1. Add a new cache pool:

```
$ hdfs cacheadmin -addPool <name> -owner <owner> -group <group> -mode <mode> -limit <limit> -maxTtl <maxTtl>
```

   Where:

   - `<name>` — the name of the new pool.
   - `<owner>` — the username of the new pool's owner. The default owner is the current user.
   - `<group>` — the pool's user group. The default group is the primary group of the current user.
   - `<mode>` — UNIX-style permissions for the pool. Permissions are specified in octal. By default, this is set to 0755.
   - `<limit>` — the maximum pool size in bytes. By default, the pool has no limit.
   - `<maxTtl>` — the maximum allowed time-to-live for directives in the pool. This can be specified in seconds, minutes, hours, and days, e.g. 120s, 30m, 4h, 2d. If the value is `never`, there is no limit. By default, no maximum is set.

2. Add a new cache directive:

```
$ hdfs cacheadmin -addDirective -path <path> -pool <pool-name> -force -replication <replication> -ttl <time-to-live>
```

   Where:

   - `<path>` — a path to a directory or a file that must be cached.
   - `<pool-name>` — the pool in which to add the directive. You must have write permission on the cache pool in order to add new directives.
   - `-force` — if this option specified, the cache pool resource limit check will be skipped.
   - `<replication>` — the cache replication factor to use. The default value is 1.
   - `<time-to-live>` — the amount of time the directive is valid. This can be specified in seconds, minutes, hours, and days, e.g. 120s, 30m, 4h, 2d. If the value is `never`, the directive never expires. The default value is `never`.

Replication factor

Elena Kostyuchenko

Contents

To Table of Contents

Change the replication factor
  - Increased replication
  - Decreased replication

Set the block size

The HDFS **replication factor** is a parameter that describes how many copies of a file to store in HDFS. When a file is created, it's divided into blocks, fixed-sized pieces of data comprising the file. Each block of that file is then copied as many times as the assigned replication factor. The default HDFS replication factor is 3.

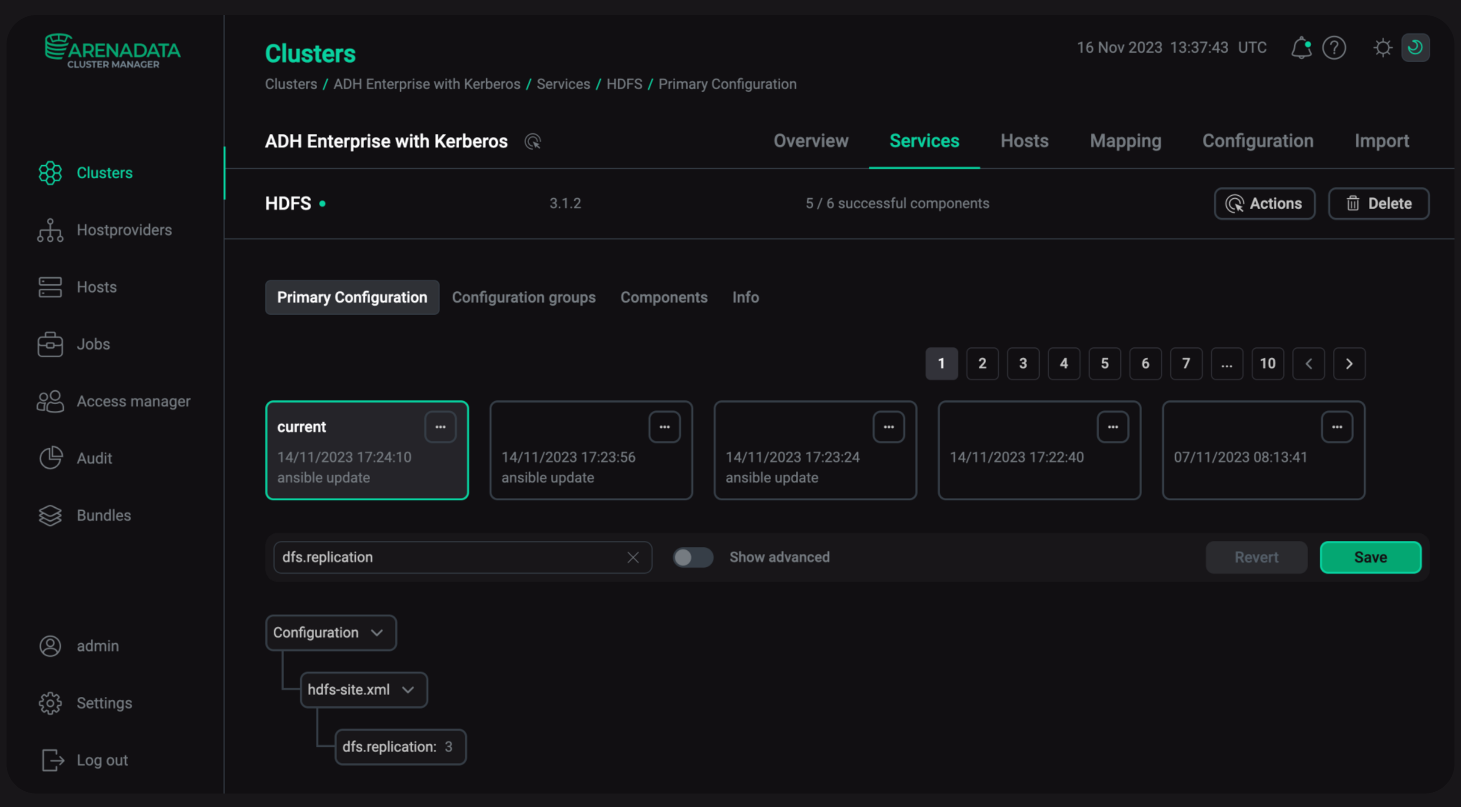
For more information about the replication, see the HDFS architecture article.

Change the replication factor

You can change the replication factor in HDFS either for the whole system, for a single file, or for a directory.

To change the default replication factor, update the value of the `dfs.replication` parameter in the *hdfs-site.xml* configuration file. To do this using ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Find the ***dfs.replication*** parameter field, enter the new default replication rate value and click ***Apply***. Confirm changes to HDFS configuration by clicking ***Save***.
4. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.



ARENADATA CLUSTER MANAGER
Clusters
16 Nov 2023 13:37:43 UTC
Clusters / ADH Enterprise with Kerberos / Services / HDFS / Primary Configuration
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
HDFS
3.1.2
5 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1 2 3 4 5 6 7 ... 10
current
14/11/2023 17:24:10 ansible update
14/11/2023 17:23:56 ansible update
14/11/2023 17:23:24 ansible update
14/11/2023 17:22:40
07/11/2023 08:13:41
Show advanced
Revert
Save
Configuration
hdfs-site.xml

dfs.replication

dfs.replication: 3

Parameter dfs.replication in ADCM

To set the replication factor for a file or a directory, use the setrep command:

```
$ hadoop fs -setrep <numReplicas> <path>
```

Where:

- `<numReplicas>` — the number indicating the replication rate for a file or a directory.
- `<path>` — a path to a file or a directory. If a path to a directory is provided, the command changes the replication factor of all files under the directory, child directories included.

> NOTE
>
> When the replication factor for a directory has been changed, it will apply only to the new files added to the directory.

Increased replication

When the default replication factor has increased, the NameNode will mark all the blocks as underreplicated and initiate replication. Make sure that the replication factor is not higher than the number of DataNodes in the cluster.

Decreased replication

When the replication factor is reduced, the NameNode sends instructions to DataNodes on which blocks to delete. It might take some time for the DataNodes to delete the blocks and report on the new available disk space.

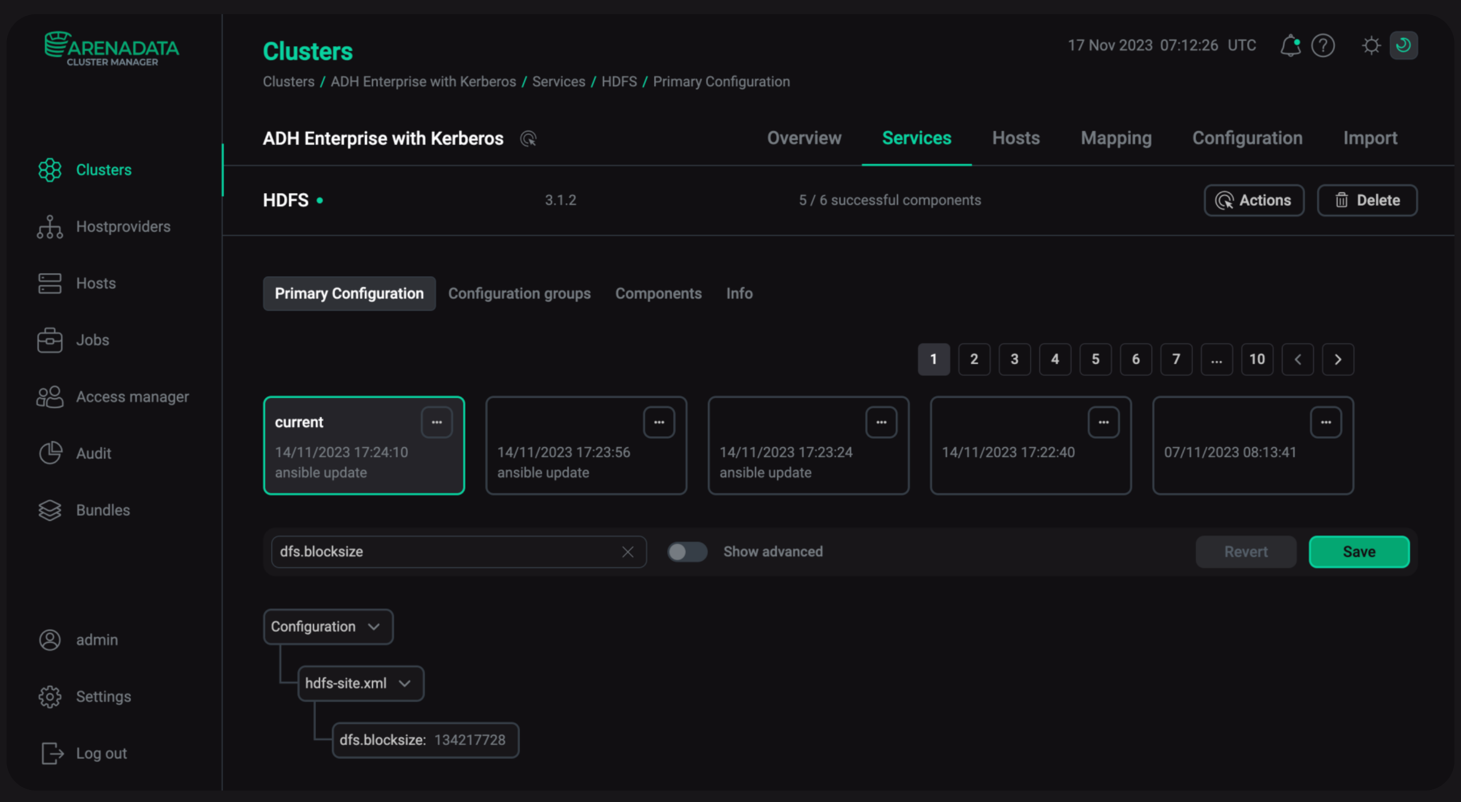
You can set the replication factor to be 1. This will decrease HDFS fault tolerance, but it can be necessary, for example, for single node clusters. If you want to set the replication factor 1 to save the resources, consider implementing storage policies or use erasure coding instead.

Set the block size

The default block size is 128 MB. You can change the block size for HDFS in the `dfs.blocksize` parameter in the *hdfs-site.xml* configuration file. To do this using ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Find the ***dfs.blocksize*** parameter field, enter the new block size in bytes and click ***Apply***. You can also use contractions, such as `k` for kilobytes, `m` for megabytes, and `g` for gigabytes.
4. Confirm changes to HDFS configuration by clicking ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.

The new block size will be applicable for all the new files added to HDFS.



ARENADATA CLUSTER MANAGER
Clusters
17 Nov 2023 07:12:26 UTC
Clusters / ADH Enterprise with Kerberos / Services / HDFS / Primary Configuration
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
HDFS
3.1.2
5 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1 2 3 4 5 6 7 ... 10
current
14/11/2023 17:24:10 ansible update
14/11/2023 17:23:56 ansible update
14/11/2023 17:23:24 ansible update
14/11/2023 17:22:40
07/11/2023 08:13:41
Show advanced
Revert
Save
Configuration
hdfs-site.xml

dfs.blocksize

dfs.blocksize: 134217728

Parameter dfs.blocksize in ADCM

Rack awareness

Elena Kostyuchenko

To Table of Contents

Rack awareness in Hadoop is an HDFS feature that provides additional fault tolerance by taking network topology into account when placing data blocks. If turned on, HDFS will place one block replica on a different rack. This insures availability of data in case of a network switch failure or other issues.

To configure rack awareness via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Toggle the ***Show advanced*** option and find the ***Topology script*** and ***Topology data*** fields.
4. In the ***Topology script***, add a script describing your network topology.

Example topology script

```
#!/bin/bash

# Adjust/Add the property "net.topology.script.file.name"
# to core-site.xml with the "absolute" path the this
# file.  ENSURE the file is "executable".

# Supply appropriate rack prefix
RACK_PREFIX=default

# To test, supply a hostname as script input:
if [ $# -gt 0 ]; then

CTL_FILE=${CTL_FILE:-"topology.data"}

HADOOP_CONF=${HADOOP_CONF:-"/etc/hadoop/conf"}

if [ ! -f ${HADOOP_CONF}/${CTL_FILE} ]; then
  echo -n "/$RACK_PREFIX/rack "
  exit 0
fi

while [ $# -gt 0 ] ; do
  nodeArg=$1
  exec< ${HADOOP_CONF}/${CTL_FILE}
  result=""
  while read line ; do
    ar=( $line )
    if [ "${ar[0]}" = "$nodeArg" ] ; then
      result="${ar[1]}"
    fi
  done
  shift
  if [ -z "$result" ] ; then
    echo -n "/$RACK_PREFIX/rack "
  else
    echo -n "/$RACK_PREFIX/rack_$result "
  fi
done

else
  echo -n "/$RACK_PREFIX/rack "
fi
```

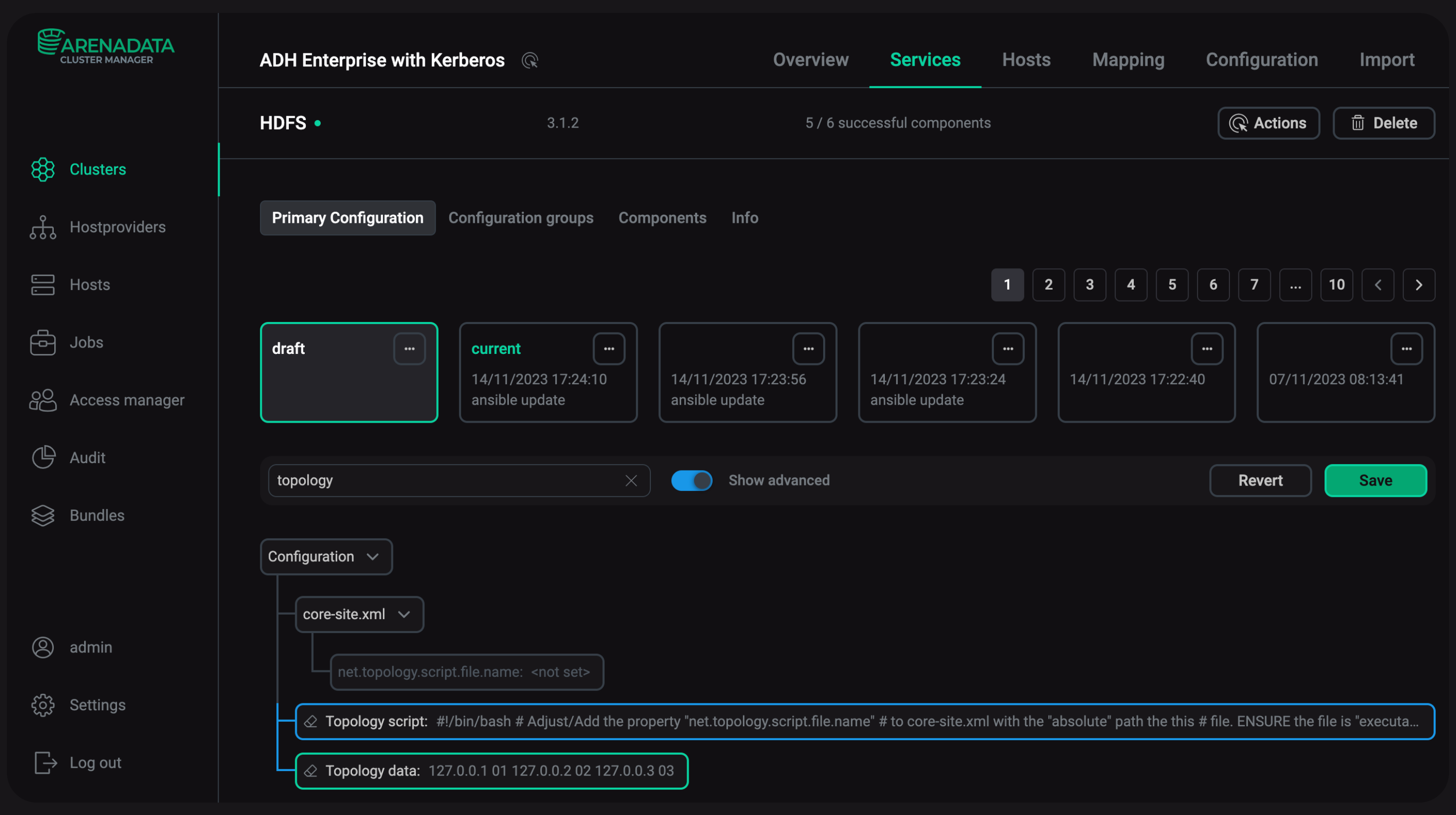
   You can find additional script examples in the Rack Awareness article.

5. In the ***Topology data*** field, list racks IPs and their corresponding IDs as follows:

Example topology data

```
# This file should be:
#  - Placed in the /etc/hadoop/conf directory
#    - On the Namenode (and backups IE: HA, Failover, etc)
#    - On the Job Tracker OR Resource Manager (and any Failover JT's/RM's)
# This file should be placed in the /etc/hadoop/conf directory.

# Add Hostnames to this file. Format <host ip> <rack_location>
127.0.0.1 01
127.0.0.2 02
127.0.0.3 03
```

6. Confirm changes to HDFS configuration by clicking ***Save***.
7. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.
8. Restart the YARN service the same way.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
HDFS
3.1.2
5 / 6 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
1 2 3 4 5 6 7 ... 10
draft
current
14/11/2023 17:24:10
ansible update
14/11/2023 17:23:56
ansible update
14/11/2023 17:23:24
ansible update
14/11/2023 17:22:40
07/11/2023 08:13:41
topology
Show advanced
Revert
Save
Configuration
core-site.xml

net.topology.script.file.name: <not set>

Topology script: #!/bin/bash # Adjust/Add the property "net.topology.script.file.name" # to core-site.xml with the "absolute" path the this # file. ENSURE the file is "executa...

Topology data: 127.0.0.1 01 127.0.0.2 02 127.0.0.3 03

Topology script and data fields in ADCM

To check if rack awareness has been successfully configured, you can grep the NameNode logs or use the fsck command.

The NameNode log will contain messages similar to these:

```
2023-10-23 13:09:52,474 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default/rack_03/127.0.0.1:1019
2023-10-23 13:09:52,569 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default/rack_01/127.0.0.2:1019
2023-10-23 13:09:52,571 INFO org.apache.hadoop.net.NetworkTopology: Adding a new node: /default/rack_02/127.0.0.3:1019
```

The `fsck` command output will display the number of racks:

```
Status: HEALTHY
 Number of data-nodes:  3
 Number of racks:               3
```

Erasure coding

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
  - Policies
- Requirements
- Manage EC

Overview

Erasure coding (EC) is a data protection mechanism for rarely used data, which uses encoding instead of replication for ensuring data resiliency. It allows decreasing resource consumption for warm and cold data while maintaining fault-tolerance.

The replication factor of three enables HDFS to sustain two data failures, but it comes at a price of storage space, costing as much as 200% overhead. Erasure coding, typically, allows the same amount of failures while costing significantly less: for example, a file with the RS-3-2-1024k EC policy would sustain the same two data failures, but incurs only a 50% overhead.

When an erasure coding policy is applied to a file, it gets striped into small blocks called **striping cells**. Then, HDFS uses an encoding algorithm on a group of cells to create a **parity file**. If a striping cell has been corrupted or lost, the parity file can be used to restore the missing cell by reconstructing it entirely using the parity file and the remaining cell. This process is called decoding.

Erasure coding is compute intensive and network intensive: data needs to be encoded during write operations and decoded for read operations if data integrity is lost.

> NOTE
>
> Since the erasure coding is an alternative to replication, any operations related to replication rate (for example, the `-setrep` command) won't have any effect on erasure coded files.

For more information about the replication, see the Replication factor article.

Policies

The erasure coding policy determines how data is encoded and how much data resiliency it provides. The name of the policy looks like `RS-10-4-1024k` and comprises the following information:

- The type of codec: either Reed-Solomon (RS) or XOR.
- The number of data blocks.
- The number of parity blocks.
- The size of a stiping cell.

Currently, five built-in policies are supported: RS-3-2-1024k, RS-6-3-1024k, RS-10-4-1024k, RS-LEGACY-6-3-1024k, XOR-2-1-1024k.

You can create a custom policy by creating an XML file *user_ec_policies.xml* describing the policy settings and using the `hdfs ec -addPolicies` command. For more information on how to create this file, see the user_ec_policies.xml.template example.

Requirements

Erasure coding places additional hardware requirements for clusters:

- **CPU**. Encoding and decoding work consumes additional CPU on both HDFS clients and DataNodes.
- **DataNodes**. Erasure coding requires a minimum of as many DataNodes as the sum of data and parity blocks. For example, for the RS-3-2-1024k policy the minimum number of DataNodes would be 5 (3 data blocks + 2 parity blocks). This number of DataNodes is required for the EC policy to work correctly, but if your cluster doesn't satisfy the requirement, HDFS will still implement the policy.
- **Racks**. Erasure coding supports rack awareness. It is beneficial to have at least as many racks as the number of parity blocks. For clusters with fewer racks, HDFS cannot maintain rack fault-tolerance, but will still attempt to spread a striped file across multiple DataNodes. For this reason, it is recommended to setup racks with similar number of DataNodes. For information about rack awareness, refer to Rack awareness.

Manage EC

Erasure coding can only be applied to a directory. Once applied, it affects only the new files added to the directory. This makes it possible for files with different EC policies and replicated files to be in the same directory.

Setting or changing a policy for a file requires rewriting its data. If you want to apply EC to existing files, you can create a new directory, set an erasure coding policy for it, and copy the files there using the distcp command.

You can manage erasure coding using the hdfs ec command.

For example, to set the XOR-2-1-1024k policy for a directory:

1. Make sure it's enabled by running the command:

```
$ hdfs ec -listPolicies
```

Example output:

```
ErasureCodingPolicy=[Name=RS-10-4-1024k, Schema=[ECSchema=[Codec=rs, numDataUnits=10, numParityUnits=4]], CellSize=1048576, Id=5], State=DISABLED
ErasureCodingPolicy=[Name=RS-3-2-1024k, Schema=[ECSchema=[Codec=rs, numDataUnits=3, numParityUnits=2]], CellSize=1048576, Id=2], State=DISABLED
ErasureCodingPolicy=[Name=RS-6-3-1024k, Schema=[ECSchema=[Codec=rs, numDataUnits=6, numParityUnits=3]], CellSize=1048576, Id=1], State=ENABLED
ErasureCodingPolicy=[Name=RS-LEGACY-6-3-1024k, Schema=[ECSchema=[Codec=rs-legacy, numDataUnits=6, numParityUnits=3]], CellSize=1048576, Id=3], State=DISABLED
ErasureCodingPolicy=[Name=XOR-2-1-1024k, Schema=[ECSchema=[Codec=xor, numDataUnits=2, numParityUnits=1]], CellSize=1048576, Id=4], State=DISABLED
```

Before applying a policy, it must be enabled first. The RS-6-3-1024k policy is enabled by default.

2. To enable the XOR-2-1-1024k policy, use the command:

```
$ hdfs ec -enablePolicy -policy XOR-2-1-1024k
```

3. To set the enabled policy for the *tmp* directory, use the command:

```
$ hdfs ec -setPolicy -path /tmp -policy XOR-2-1-1024k
```

4. Check the directory policy:

```
$ hdfs ec -getPolicy -path /tmp
```

The output will look like this:

```
XOR-2-1-1024k
```

NameNode safemode

Elena Kostyuchenko

To Table of Contents

Safemode is a NameNode state, in which it doesn't replicate blocks and doesn't allow any modifications to the file system. The operations that don't imply any changes to data blocks or the system, like read operations, are still available.

The NameNode enters this state automatically during start up to make time for synchronizing with the DataNodes. After it loads the file system state from the *fsimage* and the *edits log* file, it waits for DataNodes block reports. If the number of safely replicated blocks is at the acceptable threshold, the NameNode exits the safemode state.

The NameNode front page shows whether safemode is on or off. For more information on the NameNode UI, see HDFS UI overview.

You can place the NameNode in the safemode state using the dfs admin -safemode command.

To turn the safemode on, run:

```
$ hdfs dfsadmin -safemode enter
```

The ouput will look like this:

```
Safe mode is ON in elenas-adh.ru-central1.internal/127.0.0.1:8020
Safe mode is ON in elenas-adh1.ru-central1.internal/127.0.0.2:8020
```

If you turned on the NameNode safemode state manually, it has to be manually turned off.

To turn the safemode off, run:

```
$ hdfs dfsadmin -safemode leave
```

The ouput will look like this:

```
Safe mode is OFF in elenas-adh.ru-central1.internal/127.0.0.1:8020
Safe mode is OFF in elenas-adh1.ru-central1.internal/127.0.0.2:8020
```

In some cases, the NameNode will remain in safemode. You can forcefully turn it off, but it might mean losing some metadata.

To forcefully turn off the safemode, run:

```
$ hdfs dfsadmin -safemode forceExit
```

Snapshots in HDFS

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Snapshottable directories
- Snapshot paths
- Commands
  - Allow snapshots
  - Disable snapshots
  - List snapshottable directories
  - List snapshots
  - List snapshotted files
  - Restore from a snapshot
  - Create snapshots
  - Delete snapshots
  - Rename snapshots
  - Get snapshots difference report

Overview

The **snapshot** is a read-only image of the filesystem metadata state at the specific time. When a snapshot is made, only metadata is copied: the list of blocks for each file and its size. Even though the data itself is not saved, snapshots can still be used as a backup option and for emergency data recovery.

You can create a snapshot of a specific directory or the whole system. It requires an insignificant amount of resources from the system and can be done immediately.

Snapshottable directories

Before making a snapshot, the administrator must allow creating snapshots for the directory by flagging it as **snapshottable**. This can be done by using the allowSnapshot command.

Only one directory in a path can be marked as snapshottable; nested snapshottable directories are not allowed. This means that a directory cannot be set to snapshottable if one of its ancestors/descendants is a snapshottable directory.

Once a snapshot is made, the system creates the *.snapshot* directory that will contain snapshots. A snapshottable directory can contain up to 65536 snapshots simultaneously. The directory that contains snapshots can't be deleted or renamed before all its snapshots are deleted.

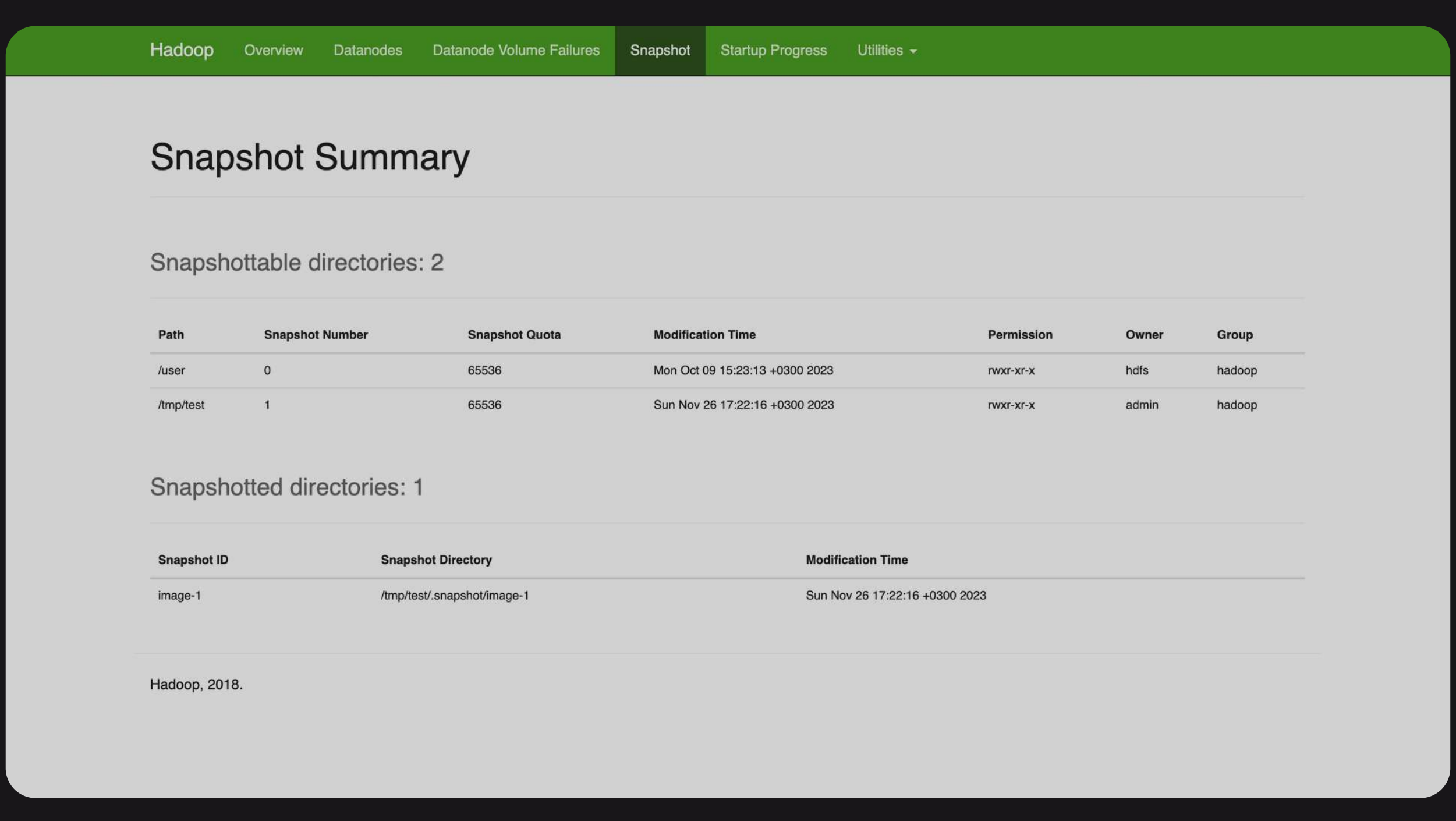
> **NOTE**
>
> Snapshots are not supported in versions of Hadoop prior to version 3.1.2. When upgrading from an older version, make sure there are no directories with the name *.snapshot*. This will help to avoid conflicts with the reserved path.

The list of snapshottable directories and available snapshots can be found in the NameNode UI, on the ***Snapshots*** page.

The page is available at:

`http://<HOST>:9870/dfshealth.html#tab-snapshot/`

Where `<HOST>` is the IP address or FQDN of the NameNode host.



Snapshot Summary

Snapshottable directories: 2

Path	Snapshot Number	Snapshot Quota	Modification Time	Permission	Owner	Group
/user	0	65536	Mon Oct 09 15:23:13 +0300 2023	rwxr-xr-x	hdfs	hadoop
/tmp/test	1	65536	Sun Nov 26 17:22:16 +0300 2023	rwxr-xr-x	admin	hadoop

Snapshotted directories: 1

Snapshot ID	Snapshot Directory	Modification Time
image-1	/tmp/test/.snapshot/image-1	Sun Nov 26 17:22:16 +0300 2023

Hadoop, 2018.

NameNode UI: Snapshots page

Snapshot paths

To access files in a snapshot using CLI and API calls, specify the path as follows:

`/<PATH>/.snapshot/<NAME>/<FILE>`

Where:

- `<PATH>` — the path to the snapshotable directory;
- `<NAME>` — the name of the snapshot;
- `<FILE>` — the path to the file in the snapshot.

Commands

Allow snapshots

Allows snapshots for a directory. The operation requires HDFS superuser privilege.

```
$ hdfs dfsadmin -allowSnapshot <path>
```

Where `<path>` is the path to the directory for which to allow snapshots.

Disable snapshots

Disables snapshots for a directory. All snapshots of the directory must be deleted before disabling snapshots. The operation requires HDFS superuser privilege.

```
$ hdfs dfsadmin -disallowSnapshot <path>
```

Where `<path>` is the path to the directory for which to disable snapshots.

List snapshottable directories

Returns the list of snapshottable directories owned by the current user. When run as a superuser, it returns all snapshottable directories.

```
$ hdfs lsSnapshottableDir
```

List snapshots

Lists all snapshots in a directory.

```
$ hdfs dfs -ls /<DIR>/.snapshot
```

Where `<DIR>` is the name of the snapshotable directory.

List snapshotted files

Lists all files and directories in a snapshot.

```
$ hdfs dfs -ls /<DIR>/.snapshot/<NAME>/
```

Where:

- `<DIR>` — the name of the snapshotable directory;
- `<NAME>` — the name of the snapshot.

Restore from a snapshot

To restore a file or a directory from a snapshot, use a regular copy command like cp or distcp with the snapshot path as source path.

The example command for restoring the deleted *text.txt* file by copying it from the *image-1* snapshot:

```
$ hdfs dfs -cp -ptopax hdfs://127.0.0.1:8020/tmp/test/.snapshot/image-1/text.txt hdfs://127.0.0.1:8020/tmp/test/text.txt
```

The `-ptopax` option is used to preserve timestamps, ownership, permission, ACLs, and XAttrs.

Create snapshots

Makes a snapshot of the specified directory. The directory must be snapshotable.

```
$ hdfs dfs -createSnapshot <path> <snapshotName>
```

Where:

- `<path>` — the path to the snapshottable directory.
- `<snapshotName>` — the name of the snapshot. This is an optional argument. If not provided, the system will generate the name using a timestamp.

Delete snapshots

Deletes the specified snapshot.

```
$ hdfs dfs -deleteSnapshot <path> <snapshotName>
```

Where:

- `<path>` — the path to the snapshottable directory;
- `<snapshotName>` — the name of the snapshot.

Rename snapshots

Renames the specified snapshot.

```
$ hdfs dfs -renameSnapshot <path> <oldName> <newName>
```

Where:

- `<path>` — the path to the snapshottable directory;
- `<oldName>` — the snapshot`s current name;
- `<newName>` — the snapshot`s new name.

Get snapshots difference report

Outputs a report on difference between two snapshots or between a snapshot and the directory's current state.

```
$ hdfs snapshotDiff <path> <start> <end>
```

Where:

- `<path>` — the path to the snapshottable directory;
- `<start>` — the name of the starting snapshot or `.` for the current state;
- `<end>` — the name of the ending snapshot or `.` for the current state.

The results interpretation:

- `+` — the file or a directory has been created;
- `-` — the file or a directory has been deleted;
- `M` — the file or a directory has been modified;
- `R` — the file or a directory has been renamed.

Example output:

```
Difference between snapshot image-1 and current directory under directory /tmp/test:
M       .
+       ./text-2.txt
+       ./text.txt
-       ./text.txt
```

NameNode recovery process in HDFS

Elena Kostyuchenko

Contents

To Table of Contents

- Before starting recovery
- Recovery process
- Recovery actions

Before starting recovery

> **CAUTION**
>
> Manual NameNode recovery can cause data loss. Do not attempt it if there's another valid copy of the edit log and *fsimage*. For example, on the other NameNode. It is preferable to use that copy rather than trying to recover the corrupted copy. If you decide to recover NameNode manually, save your metadata before trying.

In Hadoop, NameNode recovery is a process of attempting to recover lost or corrupted metadata when there is no other way to restore it. For example, in clusters with no StandBy NameNode and Checkpointing Nodes.

For High Availability clusters, manual NameNode recovery is almost never necessary. The Journal Nodes and the StandBy NameNode are usually enough to insure the metadata integrity.

To avoid metadata corruption, you can:

- Keep multiple copies of NameNode metadata on different disks.
- Configure HA mode or use checkpointing.

Recovery process

To manually recover after a NameNode failure:

1. Connect to the NameNode host via SSH.
2. As `root`, create the log directory for running the recovery command and give the `hdfs` user necessary permissions:

```
mkdir /usr/lib/hadoop/logs
chown -R hdfs:hadoop /usr/lib/hadoop/logs
chmod -R 755 /usr/lib/hadoop/logs
```

3. Login as the `hdfs` user and turn on the NameNode safemode:

```
su hdfs
$ hdfs dfsadmin -safemode enter
```

4. As `root`, turn off the NameNode:

```
systemctl stop hadoop-hdfs-namenode
```

5. As the `hdfs` user, run the recovery command (add the `-force` option to skip choice):

```
$ hdfs namenode -recover
```

   The following message appears:

```
You have selected Metadata Recovery mode.  This mode is intended to recover lost metadata on a corrupt filesystem.  Metadata recovery mode often permanently deletes data from your HDFS filesystem.  Please back up your edit log and fsimage before trying this!

Are you ready to proceed? (Y/N)
 (Y or N)
```

6. Press `Y` to start the recovery process. The system prints the status of the recovery process in the console and prompts for action if necessary. Once the recovery is complete, the system will print following:

```
2023-12-06 11:27:00,283 INFO namenode.MetaRecoveryContext: RECOVERY COMPLETE
2023-12-06 11:27:00,318 INFO namenode.FSNamesystem: Stopping services started for active state
2023-12-06 11:27:00,421 INFO namenode.FSNamesystem: Stopping services started for standby state
2023-12-06 11:27:00,423 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at elenas-adh2.ru-central1.internal/127.0.0.1
```

7. As `root`, start the NameNode:

```
systemctl start hadoop-hdfs-namenode
```

8. As the `hdfs` user, turn off the safemode:

```
$ hdfs dfsadmin -safemode leave
```

Recovery actions

During the recovery, the system might require user intervention to choose between several possible actions. If an error is encountered, the system describes it and suggests to choose one of four actions:

- `c` (Continue) — ignore the error and attempt to save the rest of the data.
- `s` (Stop) — stop reading the edit log and forfeit attempting to save the rest of the data. In this case, the edits that have not been read will be permanently lost.
- `q` (Quit) — quit recovery without saving.
- `a` (Always) — always select option `c`. The system will automatically choose to continue without prompting a user for action.

When the `-force` option is used, the system always chooses the first action, which is `c`.

Configure HDFS

Albert Bagdasaryan

Contents

To Table of Contents

- core-site.xml
- hdfs-site.xml
- httpfs-site.xml
- hadoop-env.sh
- ranger-hdfs-audit.xml
- ranger-hdfs-security.xml

You can configure HDFS in two different ways:

- Using the Arenadata Cluster Manager.
- Editing configuration files using any text editor.

There are several main configuration files:

- *core-site.xml*
- *hdfs-site.xml*
- *httpfs-site.xml*
- *ranger-hdfs-audit.xml*
- *ranger-hdfs-security.xml*
- *hadoop-env.sh*

All those configuration files are stored in the *<hadoop_home>/etc/hadoop/* directory. You can edit those files using a text editor.

NOTE

To configure ADH, we recommend that you use Arenadata Cluster Manager (ADCM).

core-site.xml

This is the most important configuration file in the HDFS. It is used to configure runtime environment settings for Hadoop, such as where the NameNode runs or which ports are used.

For more information about these parameters, see core-site.xml parameters.

hdfs-site.xml

This file contains the configuration settings for NameNodes and DataNodes. Also, this file defines the default block size for the replication process.

For more information about these parameters, see hdfs-site.xml parameters.

httpfs-site.xml

You can use the HttpFS service to interact with HDFS. For this case, you need to configure the *httpfs-site.xml* file properly. This file is used for the following purposes:

- **High availability**. WebHDFS does not support High Availability failover. The best way is to use HttpFS instead.
- **Secure impersonation**. The HDFS user is not available for secure impersonation. If you have enabled secure impersonation in an environment where the HDFS superuser is restricted from use, you can enable HttpFS and use the HttpFS superuser for secure impersonation.

TIP

If you are enabling HttpFS for use with High Availability, you should avoid enabling the HttpFS service on the primary NameNode of the ADH cluster.

For more information about these parameters, see httpfs-site.xml parameters.

hadoop-env.sh

This file is used to define configuration parameters related to a Hadoop operating environment, such as `Java_HOME`. Hadoop uses JRE, and one of the environment variables in Hadoop daemons is `Java_Home` in the *Hadoop-env.sh* file.

ranger-hdfs-audit.xml

For more profound control of the environment, this file contains configuration parameters for audit tracking and policy analytics.

For more information about these parameters, see ranger-hdfs-audit.xml parameters.

ranger-hdfs-security.xml

You can enable the Apache Ranger plugin for the HDFS service using the Ranger configuration. A security administrator has an option to configure layers of authorization controls to check access to HDFS by settings parameters in the *ranger-hdfs-security.xml* file.

When you set all the parameters, the authorization engine will check HDFS ACLs (Access Control Lists). If ACL is disabled, decisions will be based only on Ranger policies without checking HDFS ACLs. When a user attempts to access data through a data service such as Hive, access policies for both the data service and HDFS are checked.

For more information about these parameters, see ranger-hdfs-security.xml parameters.

Checkpointing in HDFS

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Configuration

Commands

Import checkpoint

Overview

Checkpointing in HDFS is a feature implemented for safer and cost-effective handling of metadata. Normally, the NameNode loads *fsimage* and applies edits during start up. If HDFS accumulates too many edits, it takes longer for the NameNode to restart.

The designated checkpointing service periodically updates the *fsimage* with the information from the edit logs. This helps preventing the edit logs from occupying too much space and enables a faster restart of the NameNode.

Creating a new *fsimage* is a computation-intensive task. To maintain system availability during this operation, the task is delegated to either Secondary NameNode or Standby NameNode.

In a regular Hyperwave cluster, the Secondary NameNode is responsible for processing edit logs and updating the *fsimage*. For safety reasons, you can keep a local up-to-date copy of the filesystem by enabling either Checkpoint node or Backup node.

> IMPORTANT
>
> In Hight availability (HA) Hyperwave clusters, merging of the edit logs into a new *fsimage* is done by the Standby NameNode; and the edit logs are managed by the Quorum Journal Manager. For this reason, using Checkpoint nodes and Backup nodes is unnecessary in HA clusters.

For more information about HDFS components, refer to the HDFS architecture article.

Configuration

Checkpointing can be carried out by one of two services:

- **Checkpoint node** — periodically updates *fsimage* on the active NameNode with the information from the edit logs.
- **Backup node** — has the same functions as the checkpoint node and, additionally, keeps in memory an up-to-date copy of the file system namespace that is synchronized with the active NameNode state.

You can configure either one but not both.

Checkpoint nodes and the backup node have the same memory requirements as a regular NameNode. An attempt to run them on the same host as the NameNode will likely result in an error.

The parameters that trigger the checkpoint creation:

- `dfs.namenode.checkpoint.period` — a time period between checkpoints. You can use a suffix to specify the time: ms (millis), s (sec), m (min), h (hour), d (day). The default is 3600s.
- `dfs.namenode.checkpoint.txns` — a maximum number of edits per checkpoint. The default is 1000000.

In the `dfs.namenode.checkpoint.check.period` parameter, you can specify how often should a checkpoint node or a backup node check if any of these conditions are met.

To configure checkpointing using ADCM, perform these steps:

Checkpoint node

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Toggle the ***Show advanced*** option and find the ***Custom hdfs-site.xml*** field.
4. Add the `dfs.namenode.backup.address` parameter and it's value `<host>:50100`, where `<host>` is the FQDN address of the checkpoint node's host. You can specify several checkpoint nodes if necessary.
5. Optionally, add the `dfs.namenode.backup.http-address` parameter and its value `<host>:50105`, where `<host>` is the FQDN address of the checkpoint node's host.
6. Confirm changes to HDFS configuration by clicking ***Save***.
7. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.
8. Connect to a cluster host via SSH and run the command as the HDFS superuser:

```
$ hdfs namenode -checkpoint
```

Backup node

<<<<<<< HEAD . On the ***CLUSTERS*** page, select the desired cluster. . Go to the ***Services*** page and click at HDFS. . Navigate to the ***Configuration*** page, turn on the ***Advanced*** flag, and find the ***Custom hdfs-site.xml*** field. . Add the `dfs.namenode.backup.address` parameter and its value `<host>:50100`, where `<host>` is the FQDN address of the backup node's host. HDFS supports only one backup node in a cluster.

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Toggle the ***Show advanced*** option and find the ***Custom hdfs-site.xml*** field.
4. Add the `dfs.namenode.backup.address` parameter and it's value `<host>:50100`, where `<host>` is the FQDN address of the backup node's host. HDFS supports only one backup node in a cluster. >>>>>>> 677937f9 (Updated screenshots and instructions)
5. Optionally, add the `dfs.namenode.backup.http-address` parameter and its value `<host>:50105`, where `<host>` is the FQDN address of the backup node's host.
6. Confirm changes to HDFS configuration by clicking ***Save***.
7. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.
8. Connect to a cluster host via SSH and run the command as the HDFS superuser:

```
$ hdfs namenode -backup
```

> TIP
>
> One of the possible errors is insufficient permissions or missing directories. Modifying permissions for some directories might be required.

Commands

To create a new checkpoint manually, run:

```
$ hdfs dfsadmin -saveNamespace
```

To clear the redundant checkpoints, use the expunge command.

For more information on checkpointing options, see the description of the following commands:

- secondarynamenode
- namenode

Import checkpoint

If the *fsimage* file has been lost or you want to create a new NameNode with the existing checkpoint, you can import a checkpoint manually.

To import a checkpoint to the NameNode:

1. Make sure that the directory specified in the `dfs.namenode.name.dir` parameter exists, and it's empty.
2. Make sure that the `dfs.namenode.checkpoint.dir` parameter contains the path to the checkpoint directory.
3. Start the NameNode using the `-importCheckpoint` option:

```
$ hdfs namenode -importCheckpoint
```

The NameNode will upload the checkpoint from the directory specified in `dfs.namenode.checkpoint.dir` and save it in the NameNode directory, set in `dfs.namenode.name.dir`.

HDFS integration with S3

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Configure S3 integration
  - Preparation steps
  - Dependencies
  - Authentication
  - Anonymous connection
- Connect to the storage
- Copy data

Overview

Simple Storage Service (S3) is an object storage that can be accessed via the S3-type protocol. A single storage container within S3 object storage is called a **bucket**. Buckets can be public, which means that anyone can access the storage, or private, which require authentication.

Hadoop supports S3A protocol for connecting to S3 storage. This enables using HDFS command line tools for interacting with remote object storages.

This article provides an example of connecting to an external S3 storage to copy data to local HDFS using the `distcp` command.

For more information on HDFS configuration for working with an S3 storage, see the Hadoop-AWS module: Integration with Amazon Web Services article. S3 storage providers may support different instruments for connecting to the storage. Consult with your S3 provider for information on other ways of integrating with the storage.

Configure S3 integration

Preparation steps

On the storage side, make sure that the user for HDFS is created and has the right type of access. Passing authentication requires credentials: the access key ID and secret access key.

If the bucket is public, you can access it anonymously.

Dependencies

ADH has a built-in object storage connector — S3A. It requires two additional libraries in the classpath:

- hadoop-aws-3.1.2.jar
- aws-java-sdk-bundle-1.12.191.jar

Add the libraries to the *usr/lib/hadoop/lib/* directory for each host.

Authentication

To connect to a private S3 bucket, a user must provide the correct credentials. When connecting to a storage, HDFS tries the following types of authentication in this order:

- URI arguments (triggers a warning);
- HDFS configuration;
- AWS Environment Variables;
- Amazon EC2 Instance Metadata Service.

For more information on how to configure each type of authentication, see the Hadoop-AWS module: Integration with Amazon Web Services article.

CAUTION

Loss of credentials can compromise the account. Protect your credentials from leaking into logs and bug reports. For this reason, avoid embedding credentials in S3A URLs.

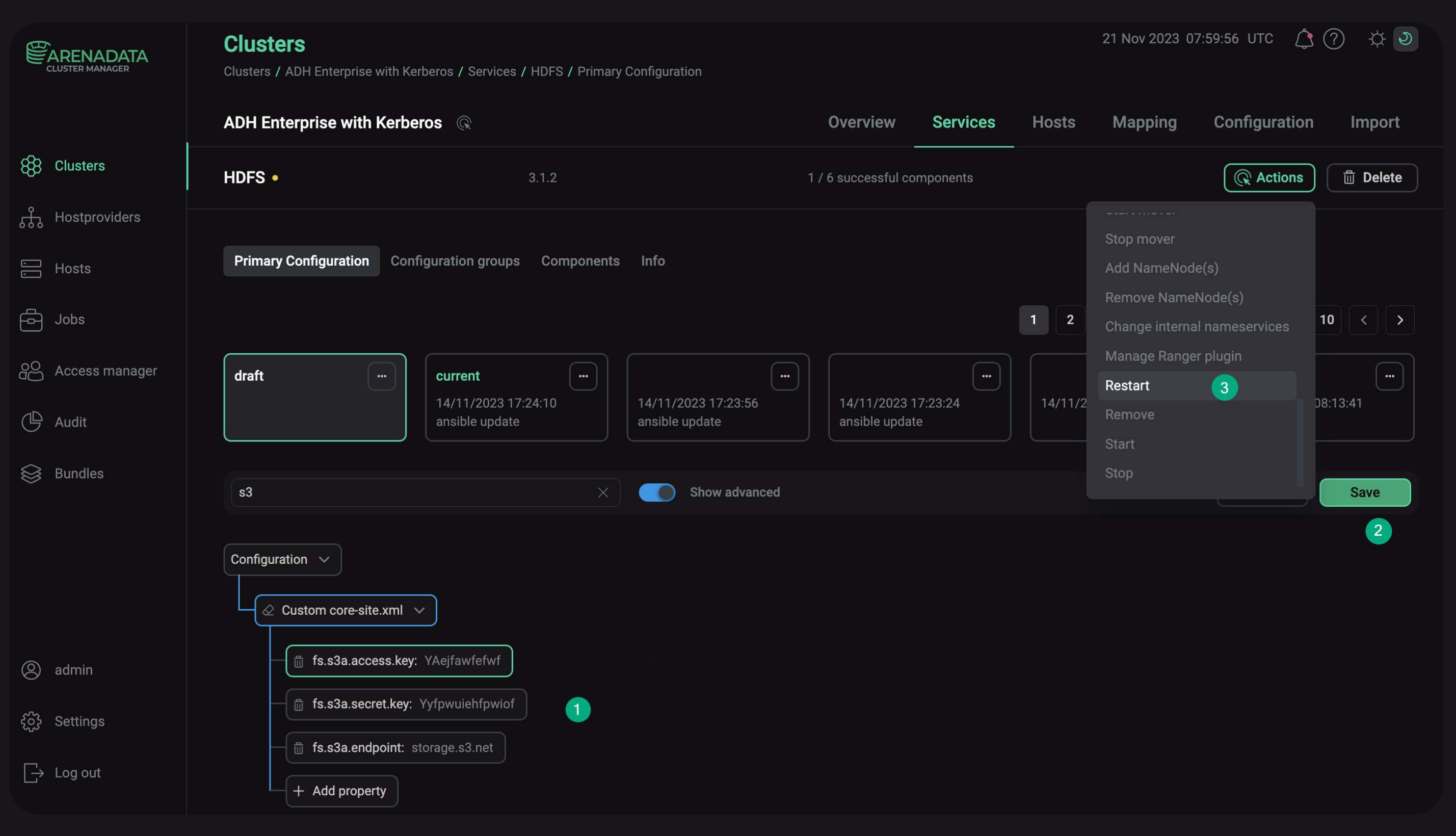
To authenticate using HDFS configuration, update the *core-site.xml* according to the table below.

Parameter	Description	Default value
fs.s3a.access.key	Access key ID. Do not specify for anonymous connection	—
fs.s3a.secret.key	Secret access key. Do not specify for anonymous connection	—
fs.s3a.endpoint	S3 endpoint to connect to. Consult with your S3 provider for information on what endpoint to use	s3.amazonaws.com

Additional properties are described in the Apache Hadoop documentation — Hadoop-AWS module: Integration with Amazon Web Services.

To configure these parameters using ADCM:

1. Navigate to the ***Clusters*** section and select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Toggle the ***Show advanced*** option, expand the ***Custom core-site.xml*** properties list, and select ***Add property***.
4. Add the required parameter with its value and click ***Apply***.
5. Confirm changes to HDFS configuration by clicking ***Save***.
6. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / ADH Enterprise with Kerberos / Services / HDFS / Primary Configuration
21 Nov 2023 07:59:56 UTC
ADH Enterprise with Kerberos
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
HDFS
3.1.2
1 / 6 successful components
Actions
Delete
Stop mover
Add NameNode(s)
Remove NameNode(s)
Change internal nameservices
Manage Ranger plugin
Restart
3
Remove
Start
Stop
Primary Configuration
Configuration groups
Components
Info
1
2
10
draft
current
14/11/2023 17:24:10
ansible update
14/11/2023 17:23:56
ansible update
14/11/2023 17:23:24
ansible update
08:13:41
s3
Show advanced
Save
2
Configuration
Custom core-site.xml
1
Add property

fs.s3a.access.key: YAejfawfefwf

fs.s3a.secret.key: Yyfpwuiehfpwiof

fs.s3a.endpoint: storage.s3.net

Edit S3 connection properties in ADCM

Anonymous connection

You can connect to a public S3 bucket anonymously. To do this, specify `org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider` as the value of the `fs.s3a.aws.credentials.provider` parameter.

This option can be used to verify that an object storage does not permit unauthenticated access. To do this, run:

```
$ hadoop fs -D fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.AnonymousAWSCredentialsProvider \
 -ls s3a://<bucket>/
```

Where `<bucket>` is the name of the bucket with which to establish a connection.

If unauthenticated access is prohibited, the connection attempt should fail like this:

```
ls: s3a://develop-test-bucket/: getFileStatus on s3a://develop-test-bucket/: com.amazonaws.services.s3.model.AmazonS3Exception:
Access Denied (Service: Amazon S3; Status Code: 403; Error Code: AccessDenied; Request ID: dc82c760b5ab707e; S3 Extended Request
ID: null; Proxy: null), S3 Extended Request ID: null:AccessDenied
```

Connect to the storage

To test the connection to the storage, run the `-ls` command:

```
$ hadoop fs -ls s3a://<bucket>/
```

Where `<bucket>` is the name of the bucket with which to establish a connection.

Output example:

```
Found 3 items
drwxrwxrwx   - admin admin          0 2023-11-14 17:28 s3a://develop-test-bucket/adbm_n1
drwxrwxrwx   - admin admin          0 2023-11-14 17:28 s3a://develop-test-bucket/adbm_n2
drwxrwxrwx   - admin admin          0 2023-11-14 17:28 s3a://develop-test-bucket/repo
```

Copy data

You can copy data between HDFS and S3 using the distcp command. To copy data from S3, run:

```
$ hadoop distcp \
      s3a://<bucket>/<path>/ \
      hdfs://<host>/<path>/
```

Where:

- `<bucket>` — the name of the bucket with which to establish a connection;
- `<host>` — the cluster nameservice or the NameNode host FQDN/IP address;
- `<path>` — the source and destination paths.

Logging in HDFS

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Grep logs
- Logging configuration
- Logging levels
- Archive logs

Overview

HDFS records file system processes output and events in text logs. This information can be useful when diagnosing technical issues in the system.

Each HDFS component stores logs locally, on the host it occupies. The logs have the *.log* extension and are located in the */var/log/hadoop-hdfs/* directory. In the same directory, you can find *.out* files that contain component restart information.

The logs naming convention is `hadoop-<user>-<component>-<host>.log`.

Where:

- `<user>` — user running Hadoop;
- `<component>` — component name, for example, DataNode;
- `<host>` — FQDN of the component host.

To look through HDFS logs in a browser, see the NameNodes or DataNodes UIs. The UIs can be accessed via links:

- `http://<HOST>:9864/logs/` — DataNode UI. Where `<HOST>` is the DataNode's host IP or FQDN.
- `http://<HOST>/logs/` — NameNode UI. Where `<HOST>` is the NameNode's host IP or FQDN.

HDFS employs the Log4j library for logging. For more information about it, see the Apache Logging Services documentation.

Grep logs

You can search through the logs for a specific information, like error messages. To do this, connect to the host with the component whose logs you want to inspect, and use a `grep` command.

For example:

```
$ cat /var/log/hadoop-hdfs/hadoop-hdfs-datanode-elenas-adh3.ru-central1.internal.log | grep -i -A3 -B1 error
```

This command searches for messages containing the word `error` in a log of the DataNode, located on the `elenas-adh3.ru-central1.internal` host. The `i` option allows you to ignore case distinctions. The `-A3 -B1` options expand the output to one line before and three lines after the line containing the error.

The example output:

```
2023-11-29 12:20:12,902 INFO org.apache.hadoop.ipc.Client: Retrying connect to server: elenas-adh2.ru-central1.internal/127.0.0.
1:8020. Already tried 2 time(s); maxRetries=45
2023-11-29 12:20:15,588 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: RECEIVED SIGNAL 15: SIGTERM
2023-11-29 12:20:15,591 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down DataNode at elenas-adh3.ru-central1.internal/127.0.0.1
--
        at org.apache.hadoop.ipc.Client$Connection.run(Client.java:1069)
2023-11-30 12:30:30,031 ERROR org.apache.hadoop.hdfs.server.datanode.DataNode: RECEIVED SIGNAL 15: SIGTERM
2023-11-30 12:30:30,035 INFO org.apache.hadoop.hdfs.server.datanode.DataNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down DataNode at elenas-adh3.ru-central1.internal/127.0.0.1
```

Logging configuration

You can configure logging settings for Hadoop via the *log4j.properties* configuration file located on each cluster's host at *etc/hadoop/conf/*.

The *log4j.properties* file contains logger settings, appender settings, and other logging properties. The parameters you set for a logger influence what a program reports to the system. The parameters you set for the appenders influence the information from loggers that ends up in the log files.

The default ADH log4j.properties file

```
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

# Define some default values that can be overridden by system properties
hadoop.root.logger=INFO,console
hadoop.log.dir=.
hadoop.log.file=hadoop.log

# Define the root logger to the system property "hadoop.root.logger".
log4j.rootLogger=${hadoop.root.logger}, EventCounter

# Logging Threshold
log4j.threshold=ALL

# Null Appender
log4j.appender.NullAppender=org.apache.log4j.varia.NullAppender

#
# Rolling File Appender - cap space usage at 5gb.
#
hadoop.log.maxfilesize=256MB
hadoop.log.maxbackupindex=20
log4j.appender.RFA=org.apache.log4j.RollingFileAppender
log4j.appender.RFA.File=${hadoop.log.dir}/${hadoop.log.file}

log4j.appender.RFA.MaxFileSize=${hadoop.log.maxfilesize}
log4j.appender.RFA.MaxBackupIndex=${hadoop.log.maxbackupindex}

log4j.appender.RFA.layout=org.apache.log4j.PatternLayout

# Pattern format: Date LogLevel LoggerName LogMessage
log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
# Debugging Pattern format
#log4j.appender.RFA.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} (%F:%M(%L)) - %m%n


#
# Daily Rolling File Appender
#

log4j.appender.DRFA=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DRFA.File=${hadoop.log.dir}/${hadoop.log.file}

# Rollover at midnight
log4j.appender.DRFA.DatePattern=.yyyy-MM-dd

log4j.appender.DRFA.layout=org.apache.log4j.PatternLayout

# Pattern format: Date LogLevel LoggerName LogMessage
log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
# Debugging Pattern format
#log4j.appender.DRFA.layout.ConversionPattern=%d{ISO8601} %-5p %c{2} (%F:%M(%L)) - %m%n


#
# console
# Add "console" to rootlogger above if you want to use this
#

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n

#
# TaskLog Appender
#
log4j.appender.TLA=org.apache.hadoop.mapred.TaskLogAppender

log4j.appender.TLA.layout=org.apache.log4j.PatternLayout
log4j.appender.TLA.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n

#
# HDFS block state change log from block manager
#
# Uncomment the following to log normal block state change
# messages from BlockManager in NameNode.
#log4j.logger.BlockStateChange=DEBUG

#
#Security appender
#
hadoop.security.logger=INFO,NullAppender
hadoop.security.log.maxfilesize=256MB
hadoop.security.log.maxbackupindex=20
log4j.category.SecurityLogger=${hadoop.security.logger}
hadoop.security.log.file=SecurityAuth-${user.name}.audit
log4j.appender.RFAS=org.apache.log4j.RollingFileAppender
log4j.appender.RFAS.File=${hadoop.log.dir}/${hadoop.security.log.file}
log4j.appender.RFAS.layout=org.apache.log4j.PatternLayout
log4j.appender.RFAS.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.RFAS.MaxFileSize=${hadoop.security.log.maxfilesize}
log4j.appender.RFAS.MaxBackupIndex=${hadoop.security.log.maxbackupindex}

#
# Daily Rolling Security appender
#
log4j.appender.DRFAS=org.apache.log4j.DailyRollingFileAppender
log4j.appender.DRFAS.File=${hadoop.log.dir}/${hadoop.security.log.file}
log4j.appender.DRFAS.layout=org.apache.log4j.PatternLayout
log4j.appender.DRFAS.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
log4j.appender.DRFAS.DatePattern=.yyyy-MM-dd

#
# hadoop configuration logging
#

# Uncomment the following line to turn off configuration deprecation warnings.
# log4j.logger.org.apache.hadoop.conf.Configuration.deprecation=WARN

#
# hdfs audit logging
#
hdfs.audit.logger=INFO,NullAppender
hdfs.audit.log.maxfilesize=256MB
hdfs.audit.log.maxbackupindex=20
log4j.logger.org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit=${hdfs.audit.logger}
log4j.additivity.org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit=false
log4j.appender.RFAAUDIT=org.apache.log4j.RollingFileAppender
log4j.appender.RFAAUDIT.File=${hadoop.log.dir}/hdfs-audit.log
log4j.appender.RFAAUDIT.layout=org.apache.log4j.PatternLayout
log4j.appender.RFAAUDIT.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n
log4j.appender.RFAAUDIT.MaxFileSize=${hdfs.audit.log.maxfilesize}
log4j.appender.RFAAUDIT.MaxBackupIndex=${hdfs.audit.log.maxbackupindex}

#
# NameNode metrics logging.
# The default is to retain two namenode-metrics.log files up to 64MB each.
#
namenode.metrics.logger=INFO,NullAppender
log4j.logger.NameNodeMetricsLog=${namenode.metrics.logger}
log4j.additivity.NameNodeMetricsLog=false
log4j.appender.NNMETRICSRFA=org.apache.log4j.RollingFileAppender
log4j.appender.NNMETRICSRFA.File=${hadoop.log.dir}/namenode-metrics.log
log4j.appender.NNMETRICSRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.NNMETRICSRFA.layout.ConversionPattern=%d{ISO8601} %m%n
log4j.appender.NNMETRICSRFA.MaxBackupIndex=1
log4j.appender.NNMETRICSRFA.MaxFileSize=64MB

#
# DataNode metrics logging.
# The default is to retain two datanode-metrics.log files up to 64MB each.
#
datanode.metrics.logger=INFO,NullAppender
log4j.logger.DataNodeMetricsLog=${datanode.metrics.logger}
log4j.additivity.DataNodeMetricsLog=false
log4j.appender.DNMETRICSRFA=org.apache.log4j.RollingFileAppender
log4j.appender.DNMETRICSRFA.File=${hadoop.log.dir}/datanode-metrics.log
log4j.appender.DNMETRICSRFA.layout=org.apache.log4j.PatternLayout
log4j.appender.DNMETRICSRFA.layout.ConversionPattern=%d{ISO8601} %m%n
log4j.appender.DNMETRICSRFA.MaxBackupIndex=1
log4j.appender.DNMETRICSRFA.MaxFileSize=64MB

# Custom Logging levels

#log4j.logger.org.apache.hadoop.mapred.JobTracker=DEBUG
#log4j.logger.org.apache.hadoop.mapred.TaskTracker=DEBUG
#log4j.logger.org.apache.hadoop.hdfs.server.namenode.FSNamesystem.audit=DEBUG


# AWS SDK & S3A FileSystem
#log4j.logger.com.amazonaws=ERROR
log4j.logger.com.amazonaws.http.AmazonHttpClient=ERROR
#log4j.logger.org.apache.hadoop.fs.s3a.S3AFileSystem=WARN

#
# Event Counter Appender
# Sends counts of logging messages at different severity levels to Hadoop Metrics.
#
log4j.appender.EventCounter=org.apache.hadoop.log.metrics.EventCounter

#
# Job Summary Appender
#
# Use following logger to send summary to separate file defined by
# hadoop.mapreduce.jobsummary.log.file :
# hadoop.mapreduce.jobsummary.logger=INFO,JSA
#
hadoop.mapreduce.jobsummary.logger=${hadoop.root.logger}
hadoop.mapreduce.jobsummary.log.file=hadoop-mapreduce.jobsummary.log
hadoop.mapreduce.jobsummary.log.maxfilesize=256MB
hadoop.mapreduce.jobsummary.log.maxbackupindex=20
log4j.appender.JSA=org.apache.log4j.RollingFileAppender
log4j.appender.JSA.File=${hadoop.log.dir}/${hadoop.mapreduce.jobsummary.log.file}
log4j.appender.JSA.MaxFileSize=${hadoop.mapreduce.jobsummary.log.maxfilesize}
log4j.appender.JSA.MaxBackupIndex=${hadoop.mapreduce.jobsummary.log.maxbackupindex}
log4j.appender.JSA.layout=org.apache.log4j.PatternLayout
log4j.appender.JSA.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n
log4j.logger.org.apache.hadoop.mapred.JobInProgress$JobSummary=${hadoop.mapreduce.jobsummary.logger}
log4j.additivity.org.apache.hadoop.mapred.JobInProgress$JobSummary=false

#
# shuffle connection log from shuffleHandler
# Uncomment the following line to enable logging of shuffle connections
# log4j.logger.org.apache.hadoop.mapred.ShuffleHandler.audit=DEBUG

#
# Yarn ResourceManager Application Summary Log
#
# Set the ResourceManager summary log filename
yarn.server.resourcemanager.appsummary.log.file=rm-appsummary.log
# Set the ResourceManager summary log level and appender
yarn.server.resourcemanager.appsummary.logger=${hadoop.root.logger}
#yarn.server.resourcemanager.appsummary.logger=INFO,RMSUMMARY

# To enable AppSummaryLogging for the RM,
# set yarn.server.resourcemanager.appsummary.logger to
# <LEVEL>,RMSUMMARY in hadoop-env.sh

# Appender for ResourceManager Application Summary Log
# Requires the following properties to be set
#    - hadoop.log.dir (Hadoop Log directory)
#    - yarn.server.resourcemanager.appsummary.log.file (resource manager app summary log filename)
#    - yarn.server.resourcemanager.appsummary.logger (resource manager app summary log level and appender)

log4j.logger.org.apache.hadoop.yarn.server.resourcemanager.RMAppManager$ApplicationSummary=${yarn.server.resourcemanager.
appsummary.logger}
log4j.additivity.org.apache.hadoop.yarn.server.resourcemanager.RMAppManager$ApplicationSummary=false
log4j.appender.RMSUMMARY=org.apache.log4j.RollingFileAppender
log4j.appender.RMSUMMARY.File=${hadoop.log.dir}/${yarn.server.resourcemanager.appsummary.log.file}
log4j.appender.RMSUMMARY.MaxFileSize=256MB
log4j.appender.RMSUMMARY.MaxBackupIndex=20
log4j.appender.RMSUMMARY.layout=org.apache.log4j.PatternLayout
log4j.appender.RMSUMMARY.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n

# HS audit log configs
#mapreduce.hs.audit.logger=INFO,HSAUDIT
#log4j.logger.org.apache.hadoop.mapreduce.v2.hs.HSAuditLogger=${mapreduce.hs.audit.logger}
#log4j.additivity.org.apache.hadoop.mapreduce.v2.hs.HSAuditLogger=false
#log4j.appender.HSAUDIT=org.apache.log4j.DailyRollingFileAppender
#log4j.appender.HSAUDIT.File=${hadoop.log.dir}/hs-audit.log
#log4j.appender.HSAUDIT.layout=org.apache.log4j.PatternLayout
#log4j.appender.HSAUDIT.layout.ConversionPattern=%d{ISO8601} %p %c{2}: %m%n
#log4j.appender.HSAUDIT.DatePattern=.yyyy-MM-dd

# Http Server Request Logs
#log4j.logger.http.requests.namenode=INFO,namenoderequestlog
#log4j.appender.namenoderequestlog=org.apache.hadoop.http.HttpRequestLogAppender
#log4j.appender.namenoderequestlog.Filename=${hadoop.log.dir}/jetty-namenode-yyyy_mm_dd.log
#log4j.appender.namenoderequestlog.RetainDays=3

#log4j.logger.http.requests.datanode=INFO,datanoderequestlog
#log4j.appender.datanoderequestlog=org.apache.hadoop.http.HttpRequestLogAppender
#log4j.appender.datanoderequestlog.Filename=${hadoop.log.dir}/jetty-datanode-yyyy_mm_dd.log
#log4j.appender.datanoderequestlog.RetainDays=3

#log4j.logger.http.requests.resourcemanager=INFO,resourcemanagerrequestlog
#log4j.appender.resourcemanagerrequestlog=org.apache.hadoop.http.HttpRequestLogAppender
#log4j.appender.resourcemanagerrequestlog.Filename=${hadoop.log.dir}/jetty-resourcemanager-yyyy_mm_dd.log
#log4j.appender.resourcemanagerrequestlog.RetainDays=3

#log4j.logger.http.requests.jobhistory=INFO,jobhistoryrequestlog
#log4j.appender.jobhistoryrequestlog=org.apache.hadoop.http.HttpRequestLogAppender
#log4j.appender.jobhistoryrequestlog.Filename=${hadoop.log.dir}/jetty-jobhistory-yyyy_mm_dd.log
#log4j.appender.jobhistoryrequestlog.RetainDays=3

#log4j.logger.http.requests.nodemanager=INFO,nodemanagerrequestlog
#log4j.appender.nodemanagerrequestlog=org.apache.hadoop.http.HttpRequestLogAppender
#log4j.appender.nodemanagerrequestlog.Filename=${hadoop.log.dir}/jetty-nodemanager-yyyy_mm_dd.log
#log4j.appender.nodemanagerrequestlog.RetainDays=3


# WebHdfs request log on datanodes
# Specify -Ddatanode.webhdfs.logger=INFO,HTTPDRFA on datanode startup to
# direct the log to a separate file.
#datanode.webhdfs.logger=INFO,console
#log4j.logger.datanode.webhdfs=${datanode.webhdfs.logger}
#log4j.appender.HTTPDRFA=org.apache.log4j.DailyRollingFileAppender
#log4j.appender.HTTPDRFA.File=${hadoop.log.dir}/hadoop-datanode-webhdfs.log
#log4j.appender.HTTPDRFA.layout=org.apache.log4j.PatternLayout
#log4j.appender.HTTPDRFA.layout.ConversionPattern=%d{ISO8601} %m%n
#log4j.appender.HTTPDRFA.DatePattern=.yyyy-MM-dd


# Appender for viewing information for errors and warnings
yarn.ewma.cleanupInterval=300
yarn.ewma.messageAgeLimitSeconds=86400
yarn.ewma.maxUniqueMessages=250
log4j.appender.EWMA=org.apache.hadoop.yarn.util.Log4jWarningErrorMetricsAppender
log4j.appender.EWMA.cleanupInterval=${yarn.ewma.cleanupInterval}
log4j.appender.EWMA.messageAgeLimitSeconds=${yarn.ewma.messageAgeLimitSeconds}
log4j.appender.EWMA.maxUniqueMessages=${yarn.ewma.maxUniqueMessages}

#
# Fair scheduler state dump
#
# Use following logger to dump the state to a separate file

#log4j.logger.org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.statedump=DEBUG,FSSTATEDUMP
#log4j.additivity.org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler.statedump=false
#log4j.appender.FSSTATEDUMP=org.apache.log4j.RollingFileAppender
#log4j.appender.FSSTATEDUMP.File=${hadoop.log.dir}/fairscheduler-statedump.log
#log4j.appender.FSSTATEDUMP.layout=org.apache.log4j.PatternLayout
#log4j.appender.FSSTATEDUMP.layout.ConversionPattern=%d{ISO8601} %p %c: %m%n
#log4j.appender.FSSTATEDUMP.MaxFileSize=${hadoop.log.maxfilesize}
#log4j.appender.FSSTATEDUMP.MaxBackupIndex=${hadoop.log.maxbackupindex}

# Log levels of third-party libraries
log4j.logger.org.apache.commons.beanutils=WARN
```

The *log4j.properties* config begins with root logger default properties. The root logger logs all messages of all components and each component has its own dedicated logger. If some of the component logger variables are not set, they will be inherited from the root logger.

The logging threshold value (`log4j.threshold`) is equivalent to logging levels: it controls what type of messages get into the output and the logs.

NOTE

The logging threshold value will affect the output messages of some commands.

The appenders control the output destination of loggers. In HDFS case, it's the log files.

For more detailed information on Log4j architecture, see the Apache Logging Services documentation.

To edit *log4j.properties* on all hosts at once, use ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at HDFS.
3. Toggle the ***Show advanced*** option and find the ***Custom log4j.properties*** field.
4. Make the necessary changes to the default file.
5. Confirm changes to HDFS configuration by clicking ***Save***.
6. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

NOTE

The *log4j.properties* file contains default values for logging parameters and they can be overwritten by HDFS daemons on restart. If it happens, you can set the parameters using environmental variables in the *hadoop-env.sh* file.

Logging levels

Hadoop supports the following logging levels (from least to most informative):

1. `FATAL` — indicates that an operation can't continue execution and will terminate.
2. `ERROR` — notifies that a program is not working correctly or has stopped.
3. `WARN` — warns about potential problems. This doesn't mean that a program is not working, but raises a concern.
4. `INFO` — informs regarding the program lifecycle or state.
5. `DEBUG` — prints debugging information about internal states of the program.
6. `TRACE` — prints messages tracing the execution flow of a program.

The Log4j logger also accepts logging level values: `OFF` — for switching off the logging, and `ALL` — for allowing all types of messages.

Enabling one level of logging will enable this level and all levels above it. For example, if you set the logging level to `WARN`, then only warnings, errors, and fatal messages would get into the log files, but not `INFO`, `DEBUG`, and `TRACE`.

To check the current log level for a component or set a different logging level, use the daemonlog command.

For example, to set the log level for the NameNode to `WARN`, run:

```
$ hadoop daemonlog -setlevel 127.0.0.1:9870 org.apache.hadoop.hdfs.server.namenode.NameNode WARN
```

This setting is not permanent and will be reset when the service restarts.

Archive logs

Some Hadoop services, like YARN and Spark, store their logs in HDFS. To reduce the number of small files in the system, you can compress logs using Hadoop archives. Aggregated logs can still be read by the Job History Server and by the yarn logs command.

To archive YARN logs, use the archive-logs command.

For more information about Hadoop archives, see the Hadoop Archive Logs Guide.

Use Observer NameNode

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Start an Observer NameNode
  - Update HDFS configuration
  - Switch a NameNode to observer state
  - Check Observer NameNode status

Overview

Traditionally, a pair of NameNodes (Active and Standby) manage all client interactions: the Active handles all namespace operations and metadata requests, while the Standby remains synchronized but does not serve clients. In an ADH cluster with high availability (HA) enabled, the introduction of the Observer NameNode role addresses read-scalability and consistent metadata-read challenges.

In large clusters with heavy metadata demands (listing directories, fetching block locations, concurrent reads), the Active NameNode often becomes a performance bottleneck. The Observer NameNode is implemented to offload read-only operations onto an additional node, allowing read traffic to be balanced away from the Active NameNode.

At the same time it preserves consistency by tailing edit logs and rejects or forwards any accidental write requests to the Active NameNode. This architecture supports consistent reads, better load distribution, and reduced latency for high-volume metadata workloads.

By deploying the Observer role, HDFS clusters gain the ability to route large numbers of metadata-read requests (for example, from BI tools, directory listing, web interfaces) away from the Active NameNode, improving throughput and resilience.

Start an Observer NameNode

Convert one of the Standby NameNodes to observer state by following the instructions below.

> **IMPORTANT**
>
> Before converting one of the NameNodes to observer state, check that your ADH cluster has at least three NameNodes. This ensures cluster health in case of failures with Active NameNode.

Update HDFS configuration

To start using Observer NameNode in ADH, update the HDFS configuration via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click on ***HDFS***.
3. In ***hdfs-site.xml*** section, set the ***dfs.namenode.state.context.enabled*** and ***dfs.ha.tail-edits.in-progress*** options to `true`.
4. Confirm changes by clicking ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

Switch a NameNode to observer state

To switch the observer state for a NameNode:

1. On the host with the required NameNode, check its state by running:

```
$ hdfs haadmin -getAllServiceState
```

Example output:

```
elenas-adh-4-2.ru-central1.internal:8020 active
elenas-adh-4-1.ru-central1.internal:8020 standby
elenas-adh-4-3.ru-central1.internal:8020 standby
```

2. Set the state of any non-active node to Observer using the command:

```
$ sudo -u hdfs hdfs haadmin -transitionToObserver <service_id> --forcemanual
```

where `<service_id>` is the ID of a Standby NameNode. You can find this ID in the HDFS UI.

Example output:

```
2025-10-21 08:30:33,203 WARN ha.HAAdmin: Proceeding with manual HA state management even though
automatic failover is enabled for NameNode at elenas-adh-4-1.ru-central1.internal/10.92.43.218:8020
```

Check Observer NameNode status

To check if the Standby NameNode status has changed, run:

```
$ hdfs haadmin -getAllServiceState
```

Example output:

```
elenas-adh-4-2.ru-central1.internal:8020 active
elenas-adh-4-1.ru-central1.internal:8020 observer
elenas-adh-4-3.ru-central1.internal:8020 standby
```

> **TIP**
>
> You can also check the status of a NameNode via HDFS UI.

HDFS service management via ADCM

Elena Kostyuchenko

Contents

To Table of Contents

Overview
Balancer
Disk balancer
Mover
Internal nameservice

Overview

The ADCM UI provides actions to manage the HDFS service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the HDFS service are listed in the table below.

Action	Description
Start balancer	Starts the HDFS Balancer
Stop balancer	Stops the HDFS Balancer
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute HDFS components
DataNode Maintenance/Decommission	Allows you to put DataNodes into maintenance mode, decommission them, or reinstate DataNodes under maintenance. A decommissioned DataNode's data will be replicated to other DataNodes. This mode can be used to safely delete a DataNode or take it down for long-term maintenance. For short-term decommissioning, use the maintenance mode. In the maintenance mode, the DataNode does not accept changes and does not replicate or delete blocks
Check disk balancer	Gets the current Disk balancer status from the specified DataNodes. To view the report, go to the ***Jobs*** page
Report disk balancer	Reports volume information from the specified DataNodes. To view the report, go to the ***Jobs*** page
Start disk balancer	Starts the disk balancer
Stop disk balancer	Stops the disk balancer
Start mover	Starts the Mover. When starting the mover, enter the directories whose storage policies must be ensured
Stop mover	Stops the Mover
Change internal nameservices	Allows you to change the internal nameservices. The value must be alphanumeric without underscores
Manage Ranger plugin	Enables or disables Ranger plugin for HDFS
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM. The service supports the ***Rolling restart*** option that allows you to restart HDFS components one by one rather than all at once (DataNode components can be restarted in batches). This feature helps to avoid the service downtime during restarts, keeping the entire cluster operable. The ***Rolling restart*** option has the following parameters: - ***batch_size*** — the number of DataNode components to be restarted in one iteration. Using batches is effective only if rack awareness is configured and the minimum number of racks is $\text{replication\_factor} - 1$. It is recommended to set the batch size less than or equal to the number of hosts per rack so that only hosts belonging to a single rack get restarted at a time, while others stay active. If rack awareness is not configured, it is recommended to set the batch size to `1`. - ***batch_delay*** — the delay in seconds between restarting batches of components. - ***health_checks*** — indicates whether to perform health checks on the restarted components. - ***max_failed_batches_number*** — the maximum number of component batches allowed to fail during restart. Upon reaching this value, the ***Restart*** action fails. The ***Rolling restart*** action restarts HDFS components in the following order: 1. HDFS DataNode 2. HDFS JournalNode 3. HDFS ZKFC and Standby NameNode 4. HDFS ZKFC and the Active NameNode 5. HDFS HttpFS

HDFS components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Balancer

The Balancer helps managing DataNodes' load in a cluster. You can start the balancer whenever there's an uneven distribution of data between DataNodes, for example, if a new DataNode has been created. The balancer stops when the DataNodes' load is at the acceptable threshold or lower.

The threshold represents how much the load of a specific DataNode may diverge from the load of the whole cluster, specified in percentage of the disk space.

After you select the ***Start balancer*** action, fill in the following fields in the window that appears (or leave empty to use the default values):

- ***Threshold*** — a percentage value between 1 and 100. The default value is 10%. Smaller values make a more balanced cluster but the balancing will take longer. If a value is too small and the DataNodes' load changes concurrently, the cluster may not be able to reach the balanced state.
- ***Hosts to exclude*** — FDQN of the hosts, whose DataNodes should be ignored by the balancer.
- ***Hosts to include*** — FDQN of the hosts, whose DataNodes should be included in the balancing process. By default, all hosts are included.
- ***Source hosts*** — FDQN of the hosts, whose DataNodes require balancing in the first place. The balancer will move blocks from only those specified DataNodes. By default, all hosts count as source hosts.
- ***Idle iterations*** — the number of iterations a balancer can remain idle before it stops. The default value is 5.

You can run balancer with additional parameters by using the balancer CLI command or by changing its parameters in the hdfs-site.xml configuration file.

Disk balancer

Disk balancer helps managing load in a single DataNode between directories. You can add data directories in the `dfs.datanode.data.dir` parameter of HDFS configuration.

You can run disk balancer with additional parameters by using the disk balancer CLI command or by changing its parameters in the hdfs-site.xml configuration file.

Mover

The Mover is a data migration tool that checks if the data in the specified directory complies with the storage policy and if it doesn't, moves the replicas to a different storage in order to fulfill the storage policy requirement.

After changing a data storage policy, it is not applied automatically. Use the mover action to ensure the new data storage policy is fulfilled.

You can run mover with additional parameters by using the mover CLI command or by changing its parameters in the hdfs-site.xml configuration file.

Internal nameservice

Internal nameservice is an additional (internal) name for an HDFS cluster that allows to query another HDFS cluster from the current one. For example, to transfer data between clusters or create tasks.

You can query any nameservice specified in the `dfs.internal.nameservices` parameter of the hdfs-site.xml configuration file. This cluster's DataNodes will report to all the nameservices in this list.

Use the Beeline shell with Hive

Mikhail Serov

Contents

To Table of Contents

- Examples
  - Beeline basics
  - Beeline with NoSASL connection
- Beeline output formats
- Cancelling the query
- Background query in terminal script

HiveServer2 supports the Beeline command shell. It is a JDBC client based on the SQLLine CLI. There is detailed documentation of SQLLine which applies to Beeline as well.

For information on replacing the implementation of Hive CLI with Beeline and the reasons to do so, see Apache Hive documentation page.

The Beeline shell works in both embedded and remote modes. In the embedded mode, it runs an embedded Hive (similar to Hive CLI) whereas the remote mode is for connecting to a separate HiveServer2 process over Thrift. Starting Hive 0.14, when Beeline is used with HiveServer2, it also prints the log messages from HiveServer2 for queries it executes to STDERR. The remote HiveServer2 mode is recommended for production use, as it is more secure and doesn't require direct HDFS/metastore access to be granted for users.

In the remote mode, HiveServer2 only accepts valid Thrift calls – even in HTTP mode, the message body should contain Thrift payloads.

Examples

Beeline basics

This section demonstrates basic interactions with Beeline.

To open the Beeline shell, run the following:

```
$ bin/beeline
```

The sample Beeline output is below.

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Beeline version 3.1.1 by Apache Hive
```

To connect to Hive, execute the following command in the Beeline shell:

```
!connect jdbc:hive2://localhost:10000 scott tiger
```

This produces an output like the one below.

```
Connecting to jdbc:hive2://localhost:10000
Connected to: Apache Hive (version 3.1.1)
Driver: Hive JDBC (version 3.1.1)
Transaction isolation: TRANSACTION_REPEATABLE_READ
```

NOTE

If Hive runs in the high availability mode, use the special JDBC string format.

To view all Hive tables, run the following in the Beeline shell:

```
show tables;
```

The command produces the following output:

```
+-------------------+
|     tab_name      |
+-------------------+
| primitives        |
| src               |
| src1              |
| src_json          |
| src_sequencefile  |
| src_thrift        |
| srcbucket         |
| srcbucket2        |
| srcpart           |
+-------------------+
9 rows selected (1.079 seconds)
```

You can also specify the connection parameters in the command line. This means you can find the command with the connection string from your UNIX shell history.

```
$ beeline -u jdbc:hive2://localhost:10000/default -n scott -w password_file
```

Beeline with NoSASL connection

If you would like to connect using the NOSASL mode, specify the authentication mode explicitly:

```
!connect jdbc:hive2://<host>:<port>/<db>;auth=noSasl hiveuser pass
```

See the available Beeline commands at the Beeline command line page.

Beeline output formats

In Beeline, the results can be displayed in different formats. The format mode can be set with the `outputformat` option.

Output formats types

Type	Description
table	The result is displayed in a table. A row of the result corresponds to a row in the table and the values in one row are displayed in separate columns in the table. This is the default format mode
vertical	Each row of the result is displayed in a block of key/value format, where the keys are the names of the columns
xmlattr	The result is displayed in an XML format where each row is a `result` element in the XML. The values of a row are displayed as attributes on the `result` element. The names of the attributes are the names of the columns
xmlelements	The result is displayed in an XML format where each row is a `result` element in the XML. The values of a row are displayed as child elements of the `result` element
json	(Hive 4.0) The result is displayed in JSON format where each row is a `result` element in the `resultset` JSON array
jsonfile	(Hive 4.0) The result is displayed in JSON format where each row is a distinct JSON object. This matches the expected format for a table created as JSONFILE format
sv output formats	The values of a row are separated by different delimiters. There are five separated-value output formats available: csv, tsv, csv2, tsv2, and dsv

Cancelling the query

When you press `CTRL+C` on the Beeline shell and there is a running query, Beeline attempts to cancel the query by closing the socket connection to HiveServer2. This behavior is enabled only when `hive.server2.close.session.on.disconnect` is set to `true`. Starting from Hive 2.2.0, Beeline does not exit the command line shell if a running query is cancelled as a result of `CTRL+C`. If you wish to exit the shell, you can press `CTRL+C` for the second time while the query is being cancelled. However, if there is no query currently running, the first `CTRL+C` will exit the Beeline shell. This behavior is similar to how the Hive CLI handles `CTRL+C`.

TIP

`!quit` is the recommended command to exit the Beeline shell.

Background query in terminal script

You can run Beeline disconnected from a terminal for batch processing and automation scripts via such commands as `nohup` and `disown`.

Some versions of Beeline client may require a workaround to allow the `nohup` command to correctly put the Beeline process in the background without stopping it.

The following environment variable can be updated:

```
export HADOOP_CLIENT_OPTS="$HADOOP_CLIENT_OPTS -Djline.terminal=jline.UnsupportedTerminal"
```

Running with `nohup` and `&` will place the process in the background and allow the terminal to disconnect while keeping the Beeline process running.

```
nohup beeline --silent=true --showHeader=true --outputformat=dsv -f query.hql </dev/null > /tmp/output.log 2> /tmp/error.log &
```

Use JDBC to connect to Hive

Konstantin Alpashkin

Contents

To Table of Contents

JDBC connection string

JDBC data types mapping

Examples

- Java
- Connect using DBeaver

This section describes ways to connect to HiveServer2 using the JDBC interface which is the recommended way for client interaction with HiveServer2. The section contains a sample Java program that opens a connection to HiveServer2 and also shows how to connect to Hive using DBeaver which is a database tool that relies on JDBC.

To connect to Hive that runs in a kerberized cluster, be sure to complete Kerberos prerequisites, like installing Kerberos client, getting a ticket, configuring *krb5.conf*, etc. For more information, see Kerberos overview.

JDBC connection string

Below are examples of JDBC connection strings depending on the security mechanism used in an ADH cluster. You can find the up-to-date JDBC connection string on the Hive ***Info*** page in ADCM (***Clusters*** → ***Services*** → ***Hive*** → ***Info***).

Without SSL/Kerberos

Connects to Hive in the high availability mode, the connection is insecure:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,
<cluster_host_N>:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/<cluster_id>/<namespace>
```

HTTP mode

Connects to Hive in the HTTP mode, the connection is insecure:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,<cluster_host_N>:2181/;transportMode=http;httpPath=<hive_endpoint>
```

Where `<hive_endpoint>` is the HTTP endpoint defined as `hive.server2.thrift.http.path` in *hive-site.xml*.

SSL enabled

Connects to Hive in the high availability mode, the connection is secured with SSL:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,
<cluster_host_N>:2181;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/<cluster_id>/<namespace>;ssl=true;ss
lTrustStore=/tmp/truststore.jks;trustStorePassword=bigdata
```

SSL+Kerberos

Connects to Hive in the high availability mode, the connection is secured with SSL+Kerberos:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,
<cluster_host_N>:2181;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/<cluster_id>/<namespace>;ssl=true;ss
lTrustStore=/tmp/truststore.jks;trustStorePassword=bigdata;principal=hive/_HOST@EXAMPLE.COM
```

Where `EXAMPLE.COM` is your Kerberos realm, for example `RU-CENTRAL1.INTERNAL`.

The sample JDBC connection strings above are intended to connect to Hive in the high availability (HA) mode. However, you can still connect to a HiveServer2 instance directly in the non-HA mode using the Thrift server port (`10000` by default). In this case, the JDBC string looks like `jdbc:hive2://<cluster_host>:10000/`.

JDBC data types mapping

The following table describes the mapping between Hive/Java data types.

Hive data type	Java data type	Comment
TINYINT	byte	Signed or unsigned 1-byte integer
SMALLINT	short	Signed 2-byte integer
INT	int	Signed 4-byte integer
BIGINT	long	Signed 8-byte integer
FLOAT	double	Single-precision number
DOUBLE	double	Double-precision number
DECIMAL	java.math.BigDecimal	Fixed-precision decimal value
BOOLEAN	boolean	Single bit (0 or 1)
STRING	String	Character string or variable-length character string
TIMESTAMP	java.sql.Timestamp	Date and time values
BINARY	String	Binary data
ARRAY	String (json-encoded)	Values of the same data type
MAP	String (json-encoded)	Key/value pairs
STRUCT	String (json-encoded)	Structured data

Examples

Java

The following snippet shows how to connect to HiveServer2 using standard JDBC tools. This program uses the `org.apache.hive.jdbc.HiveDriver` driver class from *hive-jdbc-3.1.1-arenadata-standalone.jar*. You can pull this JAR from Maven repository. This standalone JAR also contains all the dependencies required to use the driver on a standalone (non-ADH) host.

```
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.DriverManager;

public class HiveJdbcClient {
  private static String driverName = "org.apache.hive.jdbc.HiveDriver"; ①

  public static void main(String[] args) throws SQLException {
      try {
      Class.forName(driverName);
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
      System.exit(1);
    }

    Connection con = DriverManager.getConnection("jdbc:hive2://ka-adh-1.ru-central1.internal:2181," +
"ka-adh-3.ru-central1.internal:2181,ka-adh-2.ru-central1.internal:2181/;serviceDiscoveryMode=zooKeeper;" +
"zooKeeperNamespace=arenadata/cluster/2/hiveserver2", "hive", ""); ②

    // Connection con = DriverManager.getConnection("jdbc:hive2://ka-adh-1.ru-central1.internal:10000/test_db", "hive", ""); ③

    Statement stmt = con.createStatement();
    String tableName = "testHiveDriverTable";
    stmt.execute("drop table if exists " + tableName);
    stmt.execute("create table " + tableName + " (key int, value string)");
    // show tables
    String sql = "show tables '" + tableName + "'";
    System.out.println("Running query: " + sql);
    ResultSet res = stmt.executeQuery(sql);
    while (res.next()) {
      System.out.println(res.getString(1));
    }

    // other Hive queries
  }
}
```

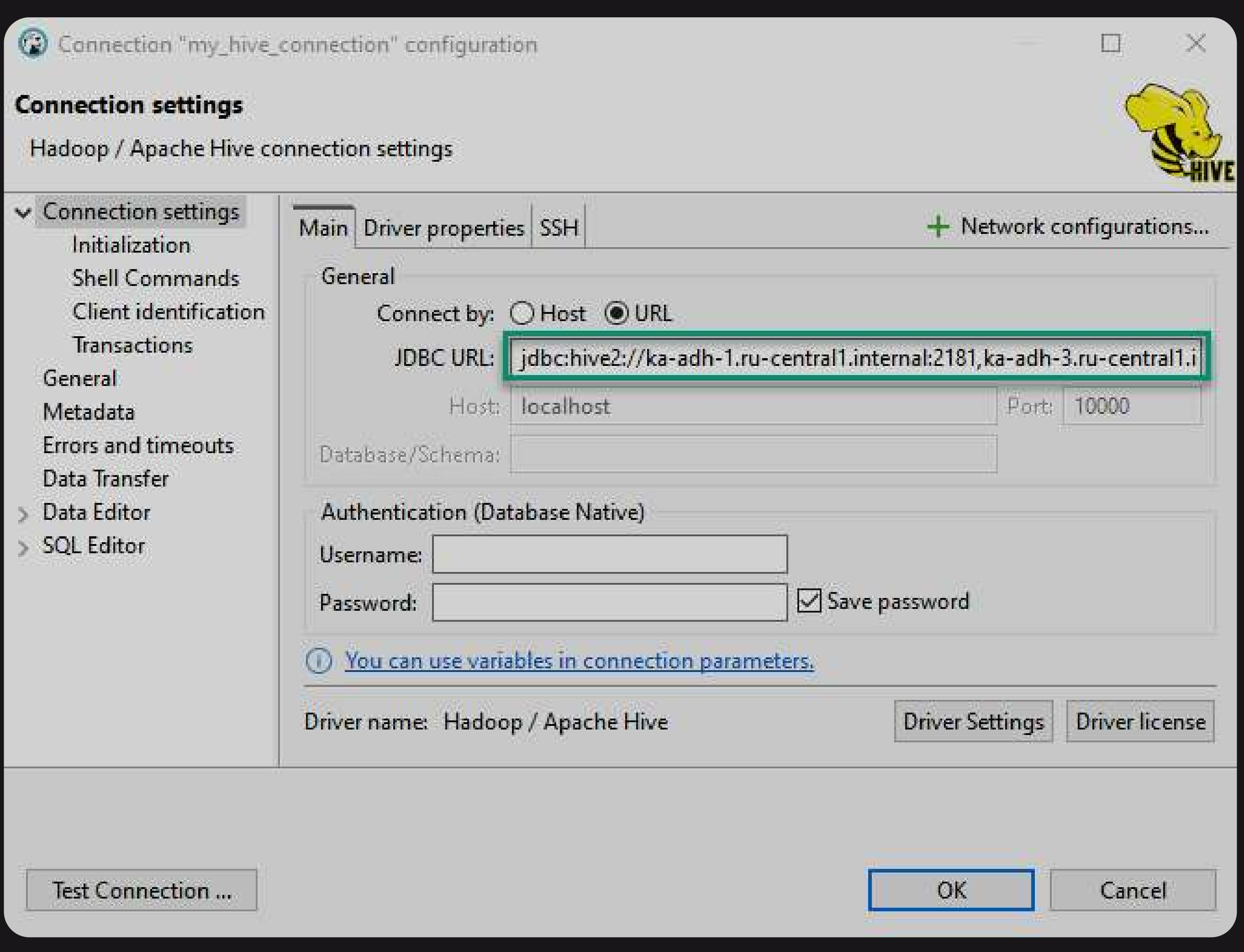
① The JDBC driver class name.

② The JDBC connection string to connect to Hive in the HA mode.

③ The JDBC connection string to connect to Hive in non-HA mode.

Connect using DBeaver

DBeaver is an open-source database tool that connects to Hive using JDBC. To connect to HiveServer2 from DBeaver, all you need is to provide an appropriate JDBC connection string, and Hive will download the default driver `org.apache.hive.jdbc.HiveDriver` automatically.



Connect by: ○ Host ● URL

JDBC URL: jdbc:hive2://ka-adh-1.ru-central1.internal:2181,ka-adh-3.ru-central1.i

Host: localhost

Port: 10000

Database/Schema:

Username:

Password:

☑ Save password

Driver name: Hadoop / Apache Hive

Hive connection settings

HiveServer2 UI overview

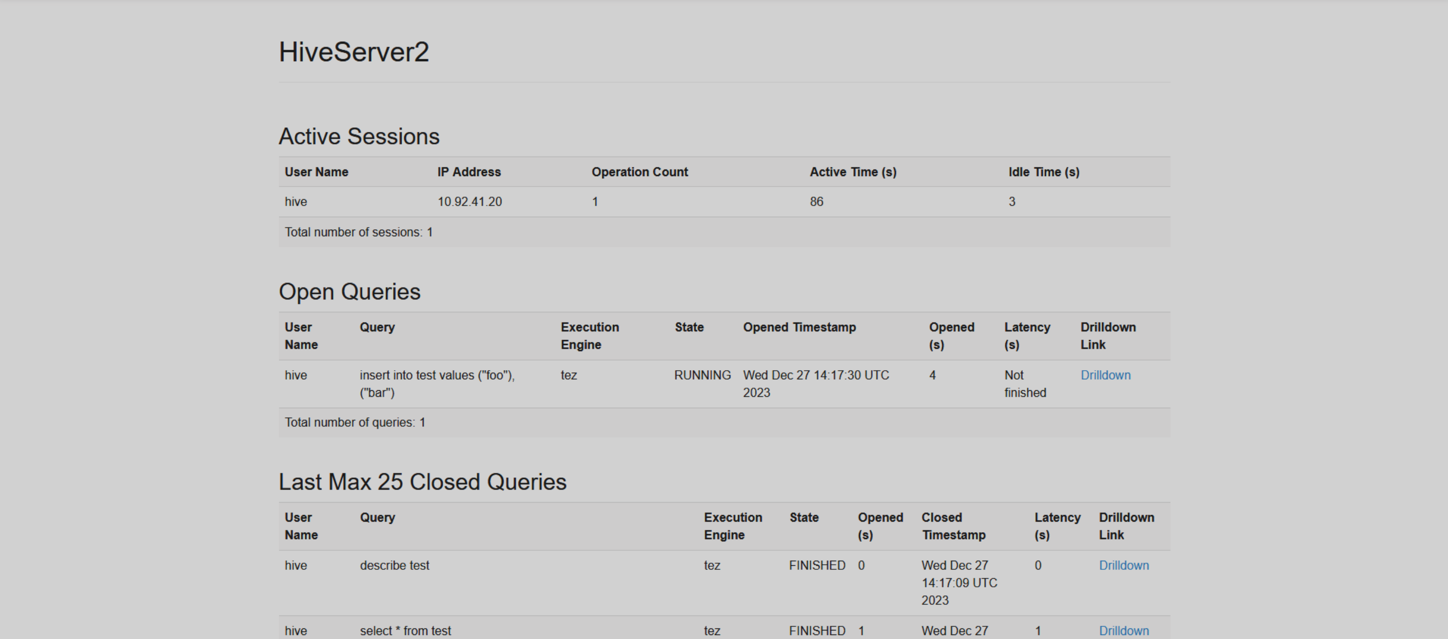
Konstantin Alpashkin

Contents

To Table of Contents

- Home
- Local logs
- Metrics Dump
- Hive Configuration
- Stack Trace
- Llap Daemons

HiveServer2 provides a web interface where you can monitor Hive configuration, logging, metrics, and active session information. The up-to-date link to the web UI can be found in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Hive*** → ***Info***). The HiveServer2 web UI has the following view.



HiveServer2

Active Sessions

User Name	IP Address	Operation Count	Active Time (s)	Idle Time (s)
hive	10.92.41.20	1	86	3
Total number of sessions: 1				

Open Queries

User Name	Query	Execution Engine	State	Opened Timestamp	Opened (s)	Latency (s)	Drilldown Link
hive	insert into test values ("foo"), ("bar")	tez	RUNNING	Wed Dec 27 14:17:30 UTC 2023	4	Not finished	Drilldown
Total number of queries: 1							

Last Max 25 Closed Queries

User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
hive	describe test	tez	FINISHED	0	Wed Dec 27 14:17:09 UTC 2023	0	Drilldown
hive	select * from test	tez	FINISHED	1	Wed Dec 27 14:16:36 UTC 2023	1	Drilldown

HiveServer2 UI

By default, the HiveServer2 UI is available on the `10002` port number. To change the port number:

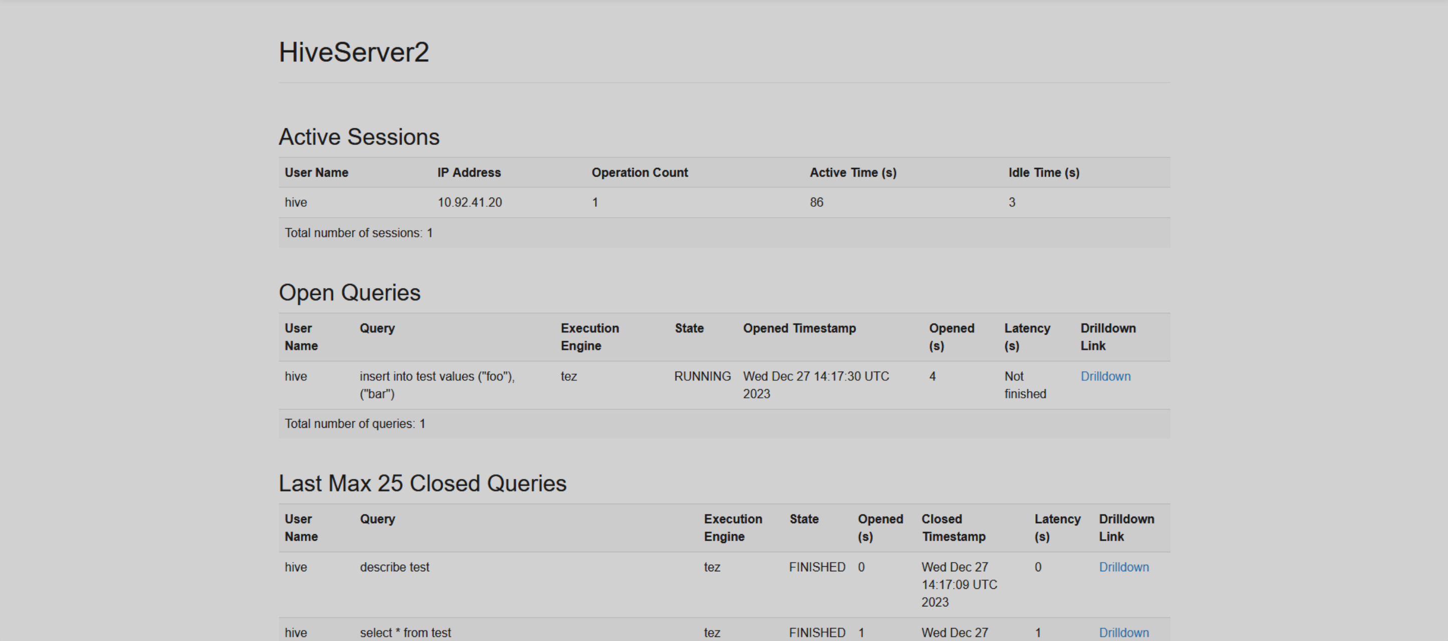
1. In ADCM, go to ***Clusters*** → ***<clusterName>*** → ***Services*** → ***Hive*** → ***Primary configuration***, and then enable ***Show advanced***.
2. In the ***Custom hive-site.xml*** section, add the new property `hive.server2.webui.port=<port>`.
3. Save the configuration and restart your Hive service. The UI will be available on the specified port after the restart.

HiveServer2 UI includes the following pages:

- Home
- Local Logs
- Metrics Dump
- Hive Configuration
- Stack Trace
- Llap Daemons

Home

The HiveServer2 ***Home*** page has the following view.


Home
Local logs
Metrics Dump
Hive Configuration
Stack Trace
Llap Daemons

HiveServer2

Active Sessions

User Name	IP Address	Operation Count	Active Time (s)	Idle Time (s)
hive	10.92.41.20	1	86	3
Total number of sessions: 1				

Open Queries

User Name	Query	Execution Engine	State	Opened Timestamp	Opened (s)	Latency (s)	Drilldown Link
hive	insert into test values ("foo"), ("bar")	tez	RUNNING	Wed Dec 27 14:17:30 UTC 2023	4	Not finished	Drilldown
Total number of queries: 1							

Last Max 25 Closed Queries

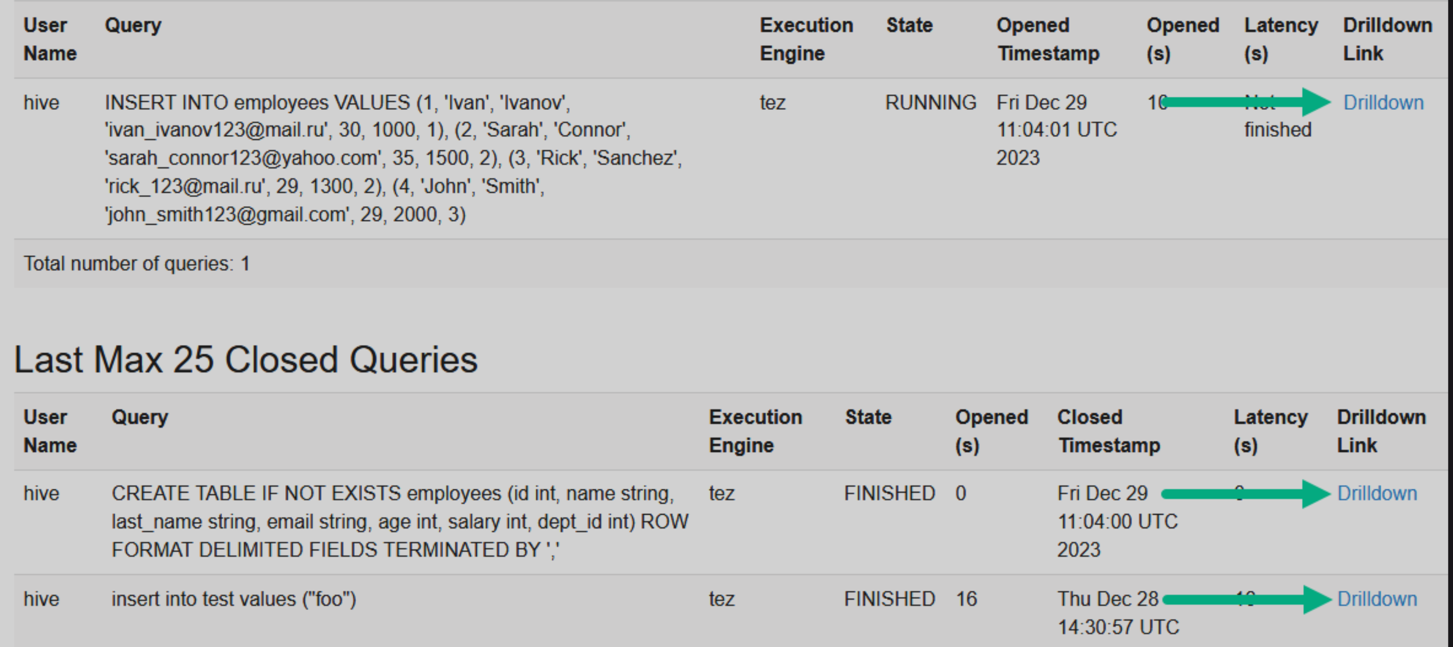
User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
hive	describe test	tez	FINISHED	0	Wed Dec 27 14:17:09 UTC 2023	0	Drilldown
hive	select * from test	tez	FINISHED	1	Wed Dec 27 14:16:36 UTC 2023	1	Drilldown

HiveServer2 Home page

This page has the following information:

- active sessions;
- queries currently processed by Hive;
- recent queries submitted to HiveServer2;
- Hive attributes, such as version, build number, uptime, etc.

To view detailed information about running and completed Hive queries, use the ***Drilldown*** link for the corresponding query as shown below.



Open Queries

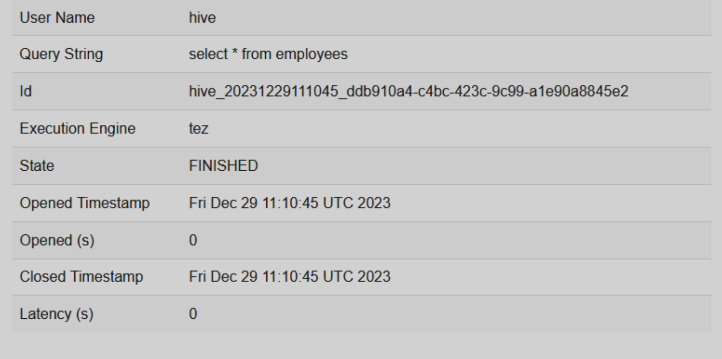
User Name	Query	Execution Engine	State	Opened Timestamp	Opened (s)	Latency (s)	Drilldown Link
hive	INSERT INTO employees VALUES (1, 'Ivan', 'Ivanov', 'ivan_ivanov123@mail.ru', 30, 1000, 1), (2, 'Sarah', 'Connor', 'sarah_connor123@yahoo.com', 35, 1500, 2), (3, 'Rick', 'Sanchez', 'rick_123@mail.ru', 29, 1300, 2), (4, 'John', 'Smith', 'john_smith123@gmail.com', 29, 2000, 3)	tez	RUNNING	Fri Dec 29 11:04:01 UTC 2023	10	Not finished	Drilldown
Total number of queries: 1							

Last Max 25 Closed Queries

User Name	Query	Execution Engine	State	Opened (s)	Closed Timestamp	Latency (s)	Drilldown Link
hive	CREATE TABLE IF NOT EXISTS employees (id int, name string, last_name string, email string, age int, salary int, dept_id int) ROW FORMAT DELIMITED FIELDS TERMINATED BY ','	tez	FINISHED	0	Fri Dec 29 11:04:00 UTC 2023	0	Drilldown
hive	insert into test values ("foo")	tez	FINISHED	16	Thu Dec 28 14:30:57 UTC 2023	16	Drilldown

Drilldown links

The ***Query Information*** page has the following view.



Query Information: select * from employees

User Name	hive
Query String	select * from employees
Id	hive_20231229111045_ddb910a4-c4bc-423c-9c99-a1e90a8845e2
Execution Engine	tez
State	FINISHED
Opened Timestamp	Fri Dec 29 11:10:45 UTC 2023
Opened (s)	0
Closed Timestamp	Fri Dec 29 11:10:45 UTC 2023
Latency (s)	0

Query Information page

The detailed query information is presented on the tabs:

- ***Base Profile***. Displays generic information about a Hive query, like ID, state, execution time, etc.
- ***Stages***. Displays Map/Reduce stages of the query.
- ***Query Plan***. Displays the query execution plan (`EXPLAIN <query>` output).
- ***Performance Logging***. Lists performance metrics from several Hive loggers.

Local logs

On this page, you can view log files generated by Hive Metastore/HiveServer2 components.

Metrics Dump

This page displays raw Hive metrics in the JSON format.

Hive Configuration

This page lists all the configuration settings from *hive-site.xml* plus effective Hadoop properties.

Stack Trace

This page displays the Java stacktrace reflecting the activity of Hive components.

Llap Daemons

Displays existing LLAP daemons (if any).

Work with Hive tables

Konstantin Alpashkin

Contents

To Table of Contents

Run Hive

Create a table

Insert data into a table

Select from a table

Delete a table

Hive provides several ways to work with tables.

You can use the DML (Data Manipulation Language) queries to import or add data to a table. Also, you can directly put a table into Hive using HDFS commands. In the examples throughout this section, we use SQL to load and insert data into a Hive table.

Run Hive

Before you start, make sure that your Hive is up and running. The Hive shell is available on the machine where the *Hive HiveServer2* component is installed.

1. Use the ADCM web-interface to check the host with the *Hive HiveServer2* component.
2. Connect via SSH to the Hive server host.
3. Run the `hive` command:

```
$ sudo -u hive hive
```

The command output is below.

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-hive.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-pig.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-thin-client.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = c93e27de-24b5-4a3c-8488-925926d54d78

Logging initialized using configuration in jar:file:/usr/lib/hive/lib/hive-common-3.1.1.jar!/hive-log4j2.properties Async: true
Hive Session ID = 88e81126-ac59-497e-bac2-709a665d4cf2
hive>
```

Create a table

Use the `CREATE` command to create a table. The following command creates the `solar_objects` table using the `,` delimiter to separate each column:

```
CREATE TABLE solar_objects(
    Name STRING,
    Mass INT,
    Diameter INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';
```

The output looks as follows:

```
OK
Time taken: 0.074 seconds
```

Insert data into a table

Use `INSERT` to populate the table with data:

```
INSERT INTO TABLE
    solar_objects
VALUES
    ('Sun',1989100000,1392000),
    ('Mercury',330,4879),
    ('Venus',4867,12104),
    ('Earth',5972,12742),
    ('Mars',642,6780),
    ('Jupiter',1898187,139822),
    ('Saturn',568317,116464),
    ('Uranus',86813,50724),
    ('Neptune',102413,49244);
```

The output looks as follows:

```
Query ID = aai_20211116141418_4a40a2c4-1d21-422c-80ee-cf0f95f35000
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1637045461848_0005)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5,42 s
----------------------------------------------------------------------------------------------
Loading data to table default.solar_objects
OK
Time taken: 15.12 seconds
```

Select from a table

Get the data using `SELECT`:

```
SELECT * FROM solar_objects;
```

The output looks as follows:

```
OK
Sun     1989100000      1392000
Mercury 330     4879
Venus   4867    12104
Earth   5972    12742
Mars    642     6780
Jupiter 1898187 139822
Saturn  568317  116464
Uranus  86813   50724
Neptune 102413  49244
Time taken: 0.17 seconds, Fetched: 9 row(s)
```

Delete a table

Delete a table using `DROP`:

```
DROP TABLE solar_objects;
```

The output looks as follows:

```
hive> hive> DROP TABLE solar_objects;
OK
Time taken: 1.991 seconds
```

Hive computation example

Konstantin Alpashkin

Contents

To Table of Contents

Dataset
Preparations
Computations

In this article, we use Hive to process criminal statistics based on the real incident reports provided by Boston Police Department (BPD). The dataset is provided by Analyze Boston. The fresh dataset is available on the Kaggle page.

Dataset

The dataset includes records starting from June 14, 2015 up to September 3, 2018. Each row represents a crime report including the type of crime, date and time, and location.

We have two files for processing:

- *crime.csv* — the main file with crime records;
- *offense_codes.csv* — the data file that contains crime codes.

After the processing we get the following information:

- the most common types of crime;
- the distribution of crimes by districts;
- the frequency of different types of crimes.

Preparations

Before the processing, complete the steps below:

1. Create directories for the data files on the Hive server host using these commands:

```
$ hdfs dfs -mkdir /user/user1/crime
$ hdfs dfs -mkdir /user/user1/offense_codes
```

2. Copy the data files into the directories you have created:

```
$ hdfs dfs -put ~/crime.csv /user/user1/crime
$ hdfs dfs -put ~/offense_codes.csv /user/user1/offense_codes
```

Computations

1. Create the `offense_codes` table:

```
CREATE EXTERNAL TABLE offense_codes (
  CODE int,
  NAME string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/user1/offense_codes' tblproperties ("skip.header.line.count"="1");
```

2. Create the `crime` table:

```
CREATE EXTERNAL TABLE crime (
    INCIDENT_NUMBER string,
    OFFENSE_CODE int,
    OFFENSE_CODE_GROUP string,
    OFFENSE_DESCRIPTION string,
    DISTRICT string,
    REPORTING_AREA string,
    SHOOTING string,
    OCCURRED_ON_DATE string,
    YEAR int,
    MONTH int,
    DAY_OF_WEEK string,
    HOUR int,
    UCR_PART string,
    STREET string,
    Lat DECIMAL,
    Long DECIMAL,
    Location string
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
LOCATION '/user/user1/crime' tblproperties ("skip.header.line.count"="1");
```

TIP

Do not forget to change the path where you store tables and data files. In this example, the sample path is */user/user1/*.

3. Run the computations with this SQL script:

```
with filtered_crimes as (
    select *
    from crime cr
    where cr.district is not null
),
     crimes_w_codes as (
         select cr.*,
                oc.*,
                trim(case
                    when instr(oc.name, '-') != 0 then
                        substr(oc.name, 1, instr(oc.name, '-') - 1)
                    else oc.name end) as crime_type
         from filtered_crimes cr
                  join offense_codes oc
                       on cr.offense_code = oc.code
     ),
     crimes_district_analytics as (
         select cr.district, count(incident_number) as crimes_total, avg(cr.lat) as lat, avg(cr.long) as lng
         from filtered_crimes cr
         group by cr.district
     ),
     crimes_by_district_by_month as (
         select cr.district, cr.month, count(cr.incident_number) as crimes_cnt
         from filtered_crimes cr
         group by cr.district, cr.month
     ),
     crimes_district_median as (
         select cr.district, percentile(cr.crimes_cnt,0.5) as crimes_monthly
         from crimes_by_district_by_month cr
         group by cr.district
     ),
     crimes_by_district_by_crime_type as (
         select cr.district, cr.crime_type, count(cr.INCIDENT_NUMBER) as crimes_cnt
         from crimes_w_codes cr
         group by cr.district, cr.crime_type
     ),
     crimes_sort as (
         select pre.district, collect_list(pre.crime_type) as frequent_crime_types
         from (
                  select cr.district,
                         cr.crime_type,
                         row_number() over (partition by cr.district order by cr.crimes_cnt desc) as rn
                  from crimes_by_district_by_crime_type cr) pre
         where pre.rn <= 3
         group by pre.district
     )
select cda.district, cda.crimes_total, cdm.crimes_monthly, cs.frequent_crime_types, cda.lat, cda.lng
from crimes_district_analytics cda
         join crimes_district_median cdm on cda.district = cdm.district
         join crimes_sort cs on cda.district = cs.district;
```

Where:

- `crimes_total` — the total crimes number in the specified district.
- `crimes_monthly` — the median crimes number per month in the specified district.
- `frequent_crime_types` — the three most frequent crime types in the entire history of observations in the specified district, separated by a comma with a single space (`, `), arranged in the descending order of frequency.
- `crime_type` — the first part of the `NAME` from the *offense_codes.csv* table, split by the `-` separator (for example, if `NAME` is `BURGLARY - COMMERICAL - ATTEMPT`, then `crime_type` is `BURGLARY`).
- `lat` — the latitude of the district coordinates, average for all incidents.
- `lng` — the longitude of the district coordinates, average for all incidents.

After the processing, we get the following results:

```
No Stats for default@crime, Columns: district, incident_number, lat, long
No Stats for default@crime, Columns: month
No Stats for default@crime, Columns: offense_code
No Stats for default@offense_codes, Columns: code, name
Query ID = user1_20211112125439_22d654c1-caeb-47f8-adc9-8e2028602653
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1636705252031_0039)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      3          3        0        0       0       0
Map 4 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 5 ...... container     SUCCEEDED      3          3        0        0       0       0
Map 10 ......... container     SUCCEEDED      1          1        0        0       0       0
Map 6 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 7 ...... container     SUCCEEDED      3          3        0        0       0       0
Reducer 8 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 9 ...... container     SUCCEEDED      3          3        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      3          3        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 10/10  [==========================>>] 100%  ELAPSED TIME: 9,21 s
----------------------------------------------------------------------------------------------
OK
cda.district    cda.crimes_total        cdm.crimes_monthly      cs.frequent_crime_types cda.lat cda.lng
 ETC"   1522    369.5   ["\"WEAPON","FIREARM/WEAPON"]   NULL    41.9364
 ETC. " 1514    382.0   ["\"DRUGS"]     NULL    41.8329
A7      13151   1062.5  ["SICK/INJURED/MEDICAL","INVESTIGATE PERSON","VANDALISM"]       41.9898 -70.9834
B2      48669   3876.0  ["VERBAL DISPUTE","SICK/INJURED/MEDICAL","INVESTIGATE PERSON"]  41.9943 -70.9907
C11     41460   3212.5  ["SICK/INJURED/MEDICAL","INVESTIGATE PERSON","VERBAL DISPUTE"]  41.9925 -70.9877
D4      41020   3221.5  ["PROPERTY","INVESTIGATE PERSON","SICK/INJURED/MEDICAL"]        41.9977 -70.9963
E5      12966   1022.5  ["SICK/INJURED/MEDICAL","INVESTIGATE PERSON","PROPERTY"]        41.9178 -70.8662
 BRUSH  525     525.0   ["\"FIRE REPORT"]       NULL    NULL
 BUILDING       1269    1269.0  ["\"FIRE REPORT"]       NULL    NULL
A15     6378    491.5   ["INVESTIGATE PERSON","VANDALISM","SICK/INJURED/MEDICAL"]       41.8010 -70.6760
        1693    107.0   ["M/V ACCIDENT","M/V","INVESTIGATE PROPERTY"]   24.8485 -43.0789
 ETC."  2597    680.0   ["\"DRUGS"]     NULL    41.8254
 FEET   1       1.0     ["\"A&B HANDS"] NULL    NULL
 NON-NEGLIGIENT MANSLAUGHTER"   161     40.0    ["\"MURDER","MURDER NON"]       NULL    41.7190
A1      34834   2698.5  ["PROPERTY","SICK/INJURED/MEDICAL","WARRANT ARREST"]    41.9741 -70.9578
B3      34618   2696.0  ["VERBAL DISPUTE","INVESTIGATE PERSON","MISSING PERSON"]        41.9961 -70.9936
C6      22903   1826.5  ["SICK/INJURED/MEDICAL","INVESTIGATE PERSON","TOWED MOTOR VEHICLE"]     41.8786 -70.8023
D14     19785   1581.0  ["TOWED MOTOR VEHICLE","SICK/INJURED/MEDICAL","INVESTIGATE PERSON"]     41.9931 -70.9888
E13     17010   1323.0  ["SICK/INJURED/MEDICAL","INVESTIGATE PERSON","TOWED MOTOR VEHICLE"]     41.9947 -70.9913
E18     16997   1289.5  ["SICK/INJURED/MEDICAL","INVESTIGATE PERSON","VERBAL DISPUTE"]  42.0000 -71.0000
Time taken: 12.691 seconds, Fetched: 20 row(s)
```

Multi-table inserts in Hive

Konstantin Alpashkin

Contents

To Table of Contents

Test database

Syntax

Examples

- Insert into multiple Hive tables
- Insert into table partitions
- Write to multiple files

Hive supports multi-table inserts that allow writing data to several Hive tables or to several HDFS files at a time. This is an effective technique as the source table is read only once and the retrieved data is reused to populate multiple targets.

Test database

Throughout this article, the following Hive table `transactions` is used to demonstrate multi-table insert examples.

```
SELECT * FROM transactions;
+----------------------+----------------------+--------------------------+------------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  |
+----------------------+----------------------+--------------------------+------------------------+
| 1                    | 1002                 | 10.00                    | 2023-01-01             |
| 8                    | 1003                 | 50.00                    | 2023-01-01             |
| 3                    | 1002                 | 30.00                    | 2023-01-02             |
| 4                    | 1001                 | 100.50                   | 2023-01-02             |
| 2                    | 1002                 | 20.00                    | 2023-01-03             |
| 6                    | 1001                 | 200.50                   | 2023-01-03             |
| 7                    | 1003                 | 50.00                    | 2023-01-03             |
| 5                    | 1001                 | 150.50                   | 2023-01-04             |
| 9                    | 1003                 | 75.00                    | 2023-01-04             |
+----------------------+----------------------+--------------------------+------------------------+
```

SQL for creating test table

The following SQL can be used to create and populate the test Hive table using */bin/beeline*.

```
CREATE DATABASE IF NOT EXISTS test_multi_inserts;
USE test_multi_inserts;

DROP TABLE IF EXISTS transactions;

CREATE TABLE transactions(
    txn_id int,
    acc_id int,
    txn_amount decimal(10,2),
    txn_date date)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

INSERT INTO transactions VALUES
(1, 1002, 10.00, '2023-01-01'),
(2, 1002, 20.00, '2023-01-03'),
(3, 1002, 30.00, '2023-01-02'),
(4, 1001, 100.50, '2023-01-02'),
(5, 1001, 150.50, '2023-01-04'),
(6, 1001, 200.50, '2023-01-03'),
(7, 1003, 50.00, '2023-01-03'),
(8, 1003, 50.00, '2023-01-01'),
(9, 1003, 75.00, '2023-01-04');
```

Syntax

The multi-insert syntax has two forms presented below:

```
FROM <src_tbl>
INSERT OVERWRITE
    (TABLE <target_tbl> [PARTITION(<partition_spec>)] | DIRECTORY [LOCAL] '<path/to/hdfs>')
<SELECT ... >


INSERT OVERWRITE
    (TABLE <target_tbl> [PARTITION(<partition_spec>)] | DIRECTORY [LOCAL] '<path/to/hdfs')
<SELECT ... FROM <src_tbl>. ... >
```

Where:

- `<src_tbl>` — the source Hive table to read data from.
- `<SELECT … >` — the `SELECT` clause to read data from `<src_tbl>`.
- `<target_tbl>` — the target table to write data to. Target tables are not created automatically and have to be pre-created manually.
- `<partition_spec>` — specifies one or more table partitions to write data into.
- `<path/to/hdfs>` — the path to an HDFS file to write data. If a relative path is provided, Hive stores files under the */user/hive/* HDFS directory. Adding the `LOCAL` keyword writes the data to the local file system.

Examples

Insert into multiple Hive tables

In the following example, the query calculates minimum and maximum transaction amount per account and writes the results into the corresponding Hive tables.

NOTE

The target tables must be created before running `INSERT`.

```
CREATE TABLE max_txn_per_acc (account integer, max_txn_value decimal(10,2));
CREATE TABLE min_txn_per_acc (account integer, min_txn_value decimal(10,2));

FROM transactions
    INSERT INTO TABLE max_txn_per_acc
        SELECT acc_id, MAX(txn_amount)
        GROUP BY acc_id
    INSERT INTO TABLE min_txn_per_acc
        SELECT acc_id, MIN(txn_amount)
        GROUP BY acc_id;
```

To verify the insertion results, query the tables:

```
SELECT * FROM max_txn_per_acc;
SELECT * FROM min_txn_per_acc;
```

The output:

```
SELECT * FROM max_txn_per_acc;
+--------------------------+--------------------------------+
| max_txn_per_acc.account  | max_txn_per_acc.max_txn_value  |
+--------------------------+--------------------------------+
| 1001                     | 200.50                         |
| 1002                     | 30.00                          |
| 1003                     | 75.00                          |
+--------------------------+--------------------------------+

SELECT * FROM min_txn_per_acc;
+--------------------------+--------------------------------+
| min_txn_per_acc.account  | min_txn_per_acc.min_txn_value  |
+--------------------------+--------------------------------+
| 1001                     | 100.50                         |
| 1002                     | 10.00                          |
| 1003                     | 50.00                          |
+--------------------------+--------------------------------++
```

Insert into table partitions

Hive allows you to insert data into specific table partitions using multi-insert operations. The following sample query writes specific source columns into multiple known partitions.

```
CREATE TABLE txns_partitioned_acc_date(txn_id int, txn_amount decimal(10, 2))
PARTITIONED BY (account_id int, txn_date date);

FROM transactions txns
    INSERT OVERWRITE TABLE txns_partitioned_acc_date
    PARTITION(account_id='1001', txn_date='2023-01-02')
SELECT txn_id, txn_amount
WHERE txns.acc_id=1001 AND txns.txn_date='2023-01-02';

SELECT * FROM test_multi_inserts.txns_partitioned_acc_date;
```

The result:

```
+-----------------------------------+---------------------------------------+---------------------------------------+------------
-------------------------+
| txns_partitioned_acc_date.txn_id  | txns_partitioned_acc_date.txn_amount  | txns_partitioned_acc_date.account_id  |
txns_partitioned_acc_date.txn_date  |
+-----------------------------------+---------------------------------------+---------------------------------------+------------
-------------------------+
| 4                                 | 100.50                                | 1001                                  | 2023-01-02
|
+-----------------------------------+---------------------------------------+---------------------------------------+------------
-------------------------+
```

By default, the `PARTITION(<spec>)` clause must include at least one static partition like `PARTITION(account_id='1001')` in the example above. To use dynamic partitions with multi-inserts (e.g. `PARTITION('acount_id')`) and thereby let Hive decide the destination partition automatically, set the following properties:

```
SET hive.exec.dynamic.partition=true;
SET hive.exec.dynamic.partition.mode=nonstrict;
```

Then, you can specify dynamic partitions as shown in the following example.

```
CREATE TABLE txns_partitioned_acc(txn_amount decimal(10, 2))
PARTITIONED BY (account_id int);

SET hive.exec.dynamic.partition.mode=nonstrict;

FROM transactions txns
    INSERT OVERWRITE TABLE txns_partitioned_acc
    PARTITION(account_id)
SELECT txn_amount, acc_id;

SELECT * FROM txns_partitioned_acc;
```

The output suggests that Hive has automatically routed the data to corresponding partitions.

```
+----------------------------------+----------------------------------+
| txns_partitioned_acc.txn_amount  | txns_partitioned_acc.account_id  |
+----------------------------------+----------------------------------+
| 100.50                           | 1001                             |
| 150.50                           | 1001                             |
| 200.50                           | 1001                             |
| 10.00                            | 1002                             |
| 20.00                            | 1002                             |
| 30.00                            | 1002                             |
| 50.00                            | 1003                             |
| 50.00                            | 1003                             |
| 75.00                            | 1003                             |
+----------------------------------+----------------------------------+
```

Write to multiple files

Hive allows writing data to multiple HDFS/local files at a time. The query below instructs Hive to write data from the source table into multiple HDFS files.

```
FROM transactions
    INSERT OVERWRITE DIRECTORY 'multi-insert-demo/transactions_20230101'
    SELECT acc_id, txn_amount WHERE txn_date='2023-01-01'
    INSERT OVERWRITE DIRECTORY 'multi-insert-demo/transactions_20230102'
    SELECT acc_id, txn_amount WHERE txn_date='2023-01-02';
```

To verify the new directories created in HDFS, use the following command:

```
$ hdfs dfs -ls /user/hive/multi-insert-demo
```

The output indicates two HDFS directories that store files with the corresponding selection results.

```
...
drwxr-xr-x   - hive hadoop          0 2023-11-29 10:51 /user/hive/multi-insert-demo/transactions_20230101
drwxr-xr-x   - hive hadoop          0 2023-11-29 10:51 /user/hive/multi-insert-demo/transactions_20230102
```

To write to files on a local file system rather than HDFS, use the `LOCAL` keyword as shown below.

```
FROM transactions
    INSERT OVERWRITE LOCAL DIRECTORY '/tmp/test.out'
SELECT transactions.*;
```

Analyze queries in Hive

Konstantin Alpashkin

Contents

To Table of Contents

- Test database
- EXPLAIN
  - EXPLAIN
  - EXPLAIN EXTENDED
  - EXPLAIN DEPENDENCY
  - EXPLAIN AUTHORIZATION
  - EXPLAIN VECTORIZATION
- Collect statistics with ANALYZE
  - Compute statistics

Before running a query Hive Cost-Based Optimizer (CBO) creates an **execution plan** — a sequence of tasks that have to be completed in order to fetch the requested data in the most efficient manner. Using Hive `EXPLAIN` and `ANALYZE` commands, you can view execution plans and get detailed statistics about Hive execution to optimize your Hive queries.

Test database

Throughout this article, the following Hive tables `transactions` and `accounts` are used to demonstrate usage examples.

```
SELECT * FROM transactions;
+----------------------+----------------------+--------------------------+------------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  |
+----------------------+----------------------+--------------------------+------------------------+
| 1                    | 1002                 | 10.00                    | 2023-01-01             |
| 8                    | 1003                 | 50.00                    | 2023-01-01             |
| 3                    | 1002                 | 30.00                    | 2023-01-02             |
| 4                    | 1001                 | 100.50                   | 2023-01-02             |
| 2                    | 1002                 | 20.00                    | 2023-01-03             |
| 6                    | 1001                 | 200.50                   | 2023-01-03             |
| 7                    | 1003                 | 50.00                    | 2023-01-03             |
| 5                    | 1001                 | 150.50                   | 2023-01-04             |
| 9                    | 1003                 | 75.00                    | 2023-01-04             |
+----------------------+----------------------+--------------------------+------------------------+

SELECT * FROM accounts;
+--------------+---------------------+
| accounts.id  | accounts.full_name  |
+--------------+---------------------+
| 1001         | John Smith          |
| 1002         | Sarah Connor        |
| 1003         | Rick Sanchez        |
+--------------+---------------------+
```

SQL for creating test table

The following SQL can be used to create and populate the test Hive tables using */bin/beeline*.

```sql
CREATE DATABASE IF NOT EXISTS explain_demo;
USE explain_demo;

DROP TABLE IF EXISTS transactions;
CREATE TABLE transactions(txn_id int, acc_id int, txn_amount decimal(10,2)) PARTITIONED BY (txn_date date);
INSERT INTO transactions VALUES
(1, 1002, 10.00, '2023-01-01'),
(2, 1002, 20.00, '2023-01-03'),
(3, 1002, 30.00, '2023-01-02'),
(4, 1001, 100.50, '2023-01-02'),
(5, 1001, 150.50, '2023-01-04'),
(6, 1001, 200.50, '2023-01-03'),
(7, 1003, 50.00, '2023-01-03'),
(8, 1003, 50.00, '2023-01-01'),
(9, 1003, 75.00, '2023-01-04');

DROP TABLE IF EXISTS accounts;
CREATE TABLE accounts(id int, full_name string);
INSERT INTO accounts VALUES
(1001, 'John Smith'),
(1002, 'Sarah Connor'),
(1003, 'Rick Sanchez');
```

EXPLAIN

The `EXPLAIN` command shows the execution plan for a given Hive query. The result includes a detailed breakdown of how Hive plans to execute the query.

The general syntax is as follows:

```
EXPLAIN [EXTENDED|DEPENDENCY|AUTHORIZATION|VECTORIZATION|FORMATTED] <query>
```

Where:

- `EXTENDED` — returns extended information about the execution plan.
- `DEPENDENCY` — provides details about tables and partitions that the query depends on.
- `AUTHORIZATION` — returns information about all entities that have to be authorized, including any permission issues (if any).
- `VECTORIZATION` — returns information about non-vectorized Map/Reduce stages.
- `FORMATTED` — formats the output as JSON.
- `<query>` — the query to analyze.

EXPLAIN

At its basic, `EXPLAIN` returns general information about the query execution plan.

```sql
CREATE TABLE IF NOT EXISTS top_txns_per_acc(acc_id int, max_txn_value decimal(10,2));

EXPLAIN
    FROM transactions INSERT OVERWRITE TABLE top_txns_per_acc
    SELECT acc_id, MAX(txn_amount) GROUP BY acc_id;
```

The result:

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| Plan optimized by CBO.                             |
|                                                    |
| Vertex dependency in root stage                    |
| Reducer 2 <- Map 1 (SIMPLE_EDGE)                   |
| Reducer 3 <- Reducer 2 (CUSTOM_SIMPLE_EDGE)        |
|                                                    |
| Stage-3                                            |
|   Stats Work{}                                     |
|     Stage-0                                        |
|       Move Operator                                |
|         table:{"name:":"explain_demo.top_txns_per_acc"} |
|         Stage-2                                    |
|           Dependency Collection{}                  |
|             Stage-1                                |
|               Reducer 3                            |
|               File Output Operator [FS_13]         |
|                 Group By Operator [GBY_11] (rows=1 width=552) |
|                   Output:["_col0","_col1"],aggregations:["compute_stats(VALUE._col0)","compute_stats(VALUE._col1)"] |
|                 <-Reducer 2 [CUSTOM_SIMPLE_EDGE]   |
|                   File Output Operator [FS_6]      |
|                     table:{"name:":"explain_demo.top_txns_per_acc"} |
|                     Group By Operator [GBY_4] (rows=4 width=12) |
|                       Output:["_col0","_col1"],aggregations:["max(VALUE._col0)"],keys:KEY._col0 |
|                     <-Map 1 [SIMPLE_EDGE] vectorized |
|                       SHUFFLE [RS_16]              |
|                         PartitionCols:_col0        |
|                         Group By Operator [GBY_15] (rows=9 width=12) |
|                           Output:["_col0","_col1"],aggregations:["max(txn_amount)"],keys:acc_id |
|                           Select Operator [SEL_14] (rows=9 width=12) |
|                             Output:["acc_id","txn_amount"] |
|                             TableScan [TS_0] (rows=9 width=12) |
|                               explain_demo@transactions,transactions,Tbl:COMPLETE,Col:NONE,Output:["acc_id","txn_amount"] |
|                   PARTITION_ONLY_SHUFFLE [RS_10]   |
|                     Group By Operator [GBY_9] (rows=1 width=536) |
|                       Output:["_col0","_col1"],aggregations:["compute_stats(acc_id, 'hll')","compute_stats(max_txn_value,
'hll')"] |
|                       Select Operator [SEL_8] (rows=4 width=12) |
|                         Output:["acc_id","max_txn_value"] |
|                          Please refer to the previous Group By Operator [GBY_4] |
|                                                    |
+----------------------------------------------------+
```

The `EXPLAIN` output may significantly vary depending on the actual query, however, the major information blocks are the following:

- **Stage dependencies**. Includes a list of dependent stages that make up the entire execution plan. Each stage can be a Map/Reduce job, an HDFS I/O operation, a Metastore access action, etc.
- **Stage plans**. Each block describes the details of individual stage, including involved operators, sort orders, etc.

EXPLAIN EXTENDED

The `EXPLAIN EXTENDED` command outputs more details about Hive execution plans. The output includes additional information about HDFS I/O operations, data formats, bucketing, advanced statistics, and so on.

```sql
EXPLAIN EXTENDED
    SELECT * FROM transactions
    WHERE txn_amount > 100;
```

EXPLAIN EXTENDED sample output

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| STAGE DEPENDENCIES:                                |
|   Stage-0 is a root stage                          |
|                                                    |
| STAGE PLANS:                                       |
|   Stage: Stage-0                                   |
|     Fetch Operator                                 |
|       limit: -1                                    |
|       Partition Description:                       |
|           Partition                                |
|             input format: org.apache.hadoop.mapred.TextInputFormat |
|             output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|             partition values:                      |
|               txn_date 2023-01-01                  |
|             properties:                            |
|               COLUMN_STATS_ACCURATE {"BASIC_STATS":"true","COLUMN_STATS":
{"acc_id":"true","txn_amount":"true","txn_id":"true"}} |
|               bucket_count -1                      |
|               column.name.delimiter ,             |
|               columns txn_id,acc_id,txn_amount     |
|               columns.comments                     |
|               columns.types int:int:decimal(10,2)  |
|               file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|               file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions/txn_date=2023-01-01 |
|               name explain_demo.transactions       |
|               numFiles 1                           |
|               numRows 2                            |
|               partition_columns txn_date           |
|               partition_columns.types date         |
|               rawDataSize 24                       |
|               serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|               serialization.format 1               |
|               serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               totalSize 26                         |
|               transient_lastDdlTime 1702847838     |
|             serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                                                    |
|               input format: org.apache.hadoop.mapred.TextInputFormat |
|               output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               properties:                          |
|                 bucket_count -1                    |
|                 bucketing_version 2                |
|                 column.name.delimiter ,           |
|                 columns txn_id,acc_id,txn_amount   |
|                 columns.comments                   |
|                 columns.types int:int:decimal(10,2) |
|                 file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|                 file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|                 location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions |
|                 name explain_demo.transactions     |
|                 partition_columns txn_date         |
|                 partition_columns.types date       |
|                 serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|                 serialization.format 1             |
|                 serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                 transient_lastDdlTime 1702846232   |
|               serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               name: explain_demo.transactions      |
|             name: explain_demo.transactions        |
|           Partition                                |
|             input format: org.apache.hadoop.mapred.TextInputFormat |
|             output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|             partition values:                      |
|               txn_date 2023-01-02                  |
|             properties:                            |
|               COLUMN_STATS_ACCURATE {"BASIC_STATS":"true","COLUMN_STATS":
{"acc_id":"true","txn_amount":"true","txn_id":"true"}} |
|               bucket_count -1                      |
|               column.name.delimiter ,             |
|               columns txn_id,acc_id,txn_amount     |
|               columns.comments                     |
|               columns.types int:int:decimal(10,2)  |
|               file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|               file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions/txn_date=2023-01-02 |
|               name explain_demo.transactions       |
|               numFiles 1                           |
|               numRows 2                            |
|               partition_columns txn_date           |
|               partition_columns.types date         |
|               rawDataSize 25                       |
|               serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|               serialization.format 1               |
|               serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               totalSize 27                         |
|               transient_lastDdlTime 1702847838     |
|             serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                                                    |
|               input format: org.apache.hadoop.mapred.TextInputFormat |
|               output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               properties:                          |
|                 bucket_count -1                    |
|                 bucketing_version 2                |
|                 column.name.delimiter ,           |
|                 columns txn_id,acc_id,txn_amount   |
|                 columns.comments                   |
|                 columns.types int:int:decimal(10,2) |
|                 file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|                 file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|                 location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions |
|                 name explain_demo.transactions     |
|                 partition_columns txn_date         |
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
|                 partition_columns.types date       |
|                 serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|                 serialization.format 1             |
|                 serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                 transient_lastDdlTime 1702846232   |
|               serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               name: explain_demo.transactions      |
|             name: explain_demo.transactions        |
|           Partition                                |
|             input format: org.apache.hadoop.mapred.TextInputFormat |
|             output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|             partition values:                      |
|               txn_date 2023-01-03                  |
|             properties:                            |
|               COLUMN_STATS_ACCURATE {"BASIC_STATS":"true","COLUMN_STATS":
{"acc_id":"true","txn_amount":"true","txn_id":"true"}} |
|               bucket_count -1                      |
|               column.name.delimiter ,             |
|               columns txn_id,acc_id,txn_amount     |
|               columns.comments                     |
|               columns.types int:int:decimal(10,2)  |
|               file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|               file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions/txn_date=2023-01-03 |
|               name explain_demo.transactions       |
|               numFiles 1                           |
|               numRows 3                            |
|               partition_columns txn_date           |
|               partition_columns.types date         |
|               rawDataSize 37                       |
|               serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|               serialization.format 1               |
|               serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               totalSize 40                         |
|               transient_lastDdlTime 1702847838     |
|             serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                                                    |
|               input format: org.apache.hadoop.mapred.TextInputFormat |
|               output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               properties:                          |
|                 bucket_count -1                    |
|                 bucketing_version 2                |
|                 column.name.delimiter ,           |
|                 columns txn_id,acc_id,txn_amount   |
|                 columns.comments                   |
|                 columns.types int:int:decimal(10,2) |
|                 file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|                 file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|                 location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions |
|                 name explain_demo.transactions     |
|                 partition_columns txn_date         |
|                 partition_columns.types date       |
|                 serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|                 serialization.format 1             |
|                 serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                 transient_lastDdlTime 1702846232   |
|               serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               name: explain_demo.transactions      |
|             name: explain_demo.transactions        |
|           Partition                                |
|             input format: org.apache.hadoop.mapred.TextInputFormat |
|             output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|             partition values:                      |
|               txn_date 2023-01-04                  |
|             properties:                            |
|               COLUMN_STATS_ACCURATE {"BASIC_STATS":"true","COLUMN_STATS":
{"acc_id":"true","txn_amount":"true","txn_id":"true"}} |
|               bucket_count -1                      |
|               column.name.delimiter ,             |
|               columns txn_id,acc_id,txn_amount     |
|               columns.comments                     |
|               columns.types int:int:decimal(10,2)  |
|               file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|               file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions/txn_date=2023-01-04 |
|               name explain_demo.transactions       |
|               numFiles 1                           |
|               numRows 2                            |
|               partition_columns txn_date           |
|               partition_columns.types date         |
|               rawDataSize 25                       |
|               serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|               serialization.format 1               |
|               serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               totalSize 27                         |
|               transient_lastDdlTime 1702847838     |
|             serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                                                    |
|               input format: org.apache.hadoop.mapred.TextInputFormat |
|               output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|               properties:                          |
|                 bucket_count -1                    |
|                 bucketing_version 2                |
|                 column.name.delimiter ,           |
|                 columns txn_id,acc_id,txn_amount   |
|                 columns.comments                   |
|                 columns.types int:int:decimal(10,2) |
|                 file.inputformat org.apache.hadoop.mapred.TextInputFormat |
|                 file.outputformat org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat |
|                 location hdfs://adh/apps/hive/warehouse/explain_demo.db/transactions |
|                 name explain_demo.transactions     |
|                 partition_columns txn_date         |
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
|                 partition_columns.types date       |
|                 serialization.ddl struct transactions { i32 txn_id, i32 acc_id, decimal(10,2) txn_amount} |
|                 serialization.format 1             |
|                 serialization.lib org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|                 transient_lastDdlTime 1702846232   |
|               serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |
|               name: explain_demo.transactions      |
|             name: explain_demo.transactions        |
|       Processor Tree:                              |
|         TableScan                                  |
|           alias: transactions                      |
|           GatherStats: false                       |
|           Filter Operator                          |
|             isSamplingPred: false                  |
|             predicate: (txn_amount > 100) (type: boolean) |
|             Select Operator                        |
|               expressions: txn_id (type: int), acc_id (type: int), txn_amount (type: decimal(10,2)), txn_date (type: date) |
|               outputColumnNames: _col0, _col1, _col2, _col3 |
|               ListSink                             |
|                                                    |
+----------------------------------------------------+
```

EXPLAIN DEPENDENCY

The `EXPLAIN DEPENDENCY` command returns extra information about all the input sources (Hive tables, partitions, HDFS files) that will be accessed in order to fetch the requested data.

```sql
EXPLAIN DEPENDENCY
    SELECT t.txn_id, a.full_name
    FROM transactions t
    JOIN accounts a ON (t.acc_id=a.id)
```

The output:

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| {"input_tables":[{"tablename":"explain_demo@transactions","tabletype":"MANAGED_TABLE"},
{"tablename":"explain_demo@accounts","tabletype":"MANAGED_TABLE"}],"input_partitions":
[{"partitionName":"explain_demo@transactions@txn_date=2023-01-01"},{"partitionName":"explain_demo@transactions@txn_date=2023-01-
02"},{"partitionName":"explain_demo@transactions@txn_date=2023-01-03"},{"partitionName":"explain_demo@transactions@txn_date=2023-
01-04"}]} |
+----------------------------------------------------+
```

EXPLAIN AUTHORIZATION

Using `EXPLAIN AUTHORIZATION`, you can get authorization information for a query, including permission issues (if any).

For example:

```sql
EXPLAIN AUTHORIZATION
    SELECT a.full_name, t.txn_date, t.txn_id from accounts a
    JOIN transactions t ON (a.id=t.acc_id);
```

The result:

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| INPUTS:                                            |
|   explain_demo@accounts                            |
|   explain_demo@transactions                        |
|   explain_demo@transactions@txn_date=2023-01-01    |
|   explain_demo@transactions@txn_date=2023-01-02    |
|   explain_demo@transactions@txn_date=2023-01-03    |
|   explain_demo@transactions@txn_date=2023-01-04    |
| OUTPUTS:                                           |
|   hdfs://adh/tmp/hive/hive/de6b68f3-0c62-40e9-84ac-8e34916558c9/hive_2023-12-18_09-26-13_342_1170854594803480484-104/-mr-10001
|
| CURRENT_USER:                                      |
|   hive                                             |
| OPERATION:                                         |
|   QUERY                                            |
+----------------------------------------------------+
```

EXPLAIN VECTORIZATION

The `VECTORIZATION` keyword adds details to the `EXPLAIN` output indicating why specific Map/Reduce jobs are not vectorized (if any).

The syntax is as follows:

```
EXPLAIN VECTORIZATION [ONLY] [SUMMARY|OPERATOR|EXPRESSION|DETAIL]
```

Where:

- `ONLY` — suppresses most non-vectorization elements.
- `SUMMARY` — shows if vectorization is enabled and adds a summary of Map and Reduce work.
- `OPERATOR` — shows vectorization information for operators.
- `EXPRESSION` — shows vectorization information for expressions.
- `DETAIL` — the most detailed information level. Includes all of the above.

Collect statistics with ANALYZE

When processing a query, Hive CBO optimizer makes extensive use of Hive statistics to choose the optimal query plan in terms of system resources. This statistics (number of rows, number of files, raw data size) is stored in Hive Metastore and by default is re-generated automatically when new data is inserted into the table (`hive.stats.autogather=true`).

Compute statistics

To collect statistics for a given table or a partition at any time, use the `ANALYZE` command. This command computes the statistics on demand and writes it to Hive Metastore. Hive supports statistics at the table, partition, and column level.

The syntax is as follows:

```
ANALYZE TABLE <tbl_name> [PARTITION(<partition>)] COMPUTE STATISTICS
[FOR COLUMNS c1, c2, ...]
[NOSCAN];
```

Where:

- `FOR COLUMNS` — computes column statistics either for all table columns (if no columns are specified) or only for specific columns of the table.
- `NOSCAN` — if specified, the command does not scan all the files. Instead, the statistics will include the number of bytes and physical size in bytes.

NOTE

By default, statistics gathering is enabled for newly created tables/partitions, so there is no need of computing the statistics manually unless you changed the default `hive.stats.autogather=true` property.

Once the statistics is written to Hive Metastore, you can view the stored statistics by issuing the `DESCRIBE` command. The syntax is as follows:

```
DESCRIBE [EXTENDED|FORMATTED] <tbl_name> [PARTITION(<partition>)]
```

The following example computes and shows Hive statistics for entire table:

```sql
ANALYZE TABLE transactions COMPUTE STATISTICS;
DESCRIBE EXTENDED transactions;
```

The output:

```
+-----------------------------+----------------------------------------------------+----------+
|          col_name           |                     data_type                      | comment  |
+-----------------------------+----------------------------------------------------+----------+
| txn_id                      | int                                                |          |
| acc_id                      | int                                                |          |
| txn_amount                  | decimal(10,2)                                      |          |
| txn_date                    | date                                               |          |
|                             | NULL                                               | NULL     |
| # Partition Information     | NULL                                               | NULL     |
| # col_name                  | data_type                                          | comment  |
| txn_date                    | date                                               |          |
|                             | NULL                                               | NULL     |
| Detailed Table Information  | Table(tableName:transactions, dbName:explain_demo, owner:hive, createTime:1702839122,
lastAccessTime:0, retention:0, sd:StorageDescriptor(cols:[FieldSchema(name:txn_id, type:int, comment:null),
FieldSchema(name:acc_id, type:int, comment:null), FieldSchema(name:txn_amount, type:decimal(10,2), comment:null),
FieldSchema(name:txn_date, type:date, comment:null)], location:hdfs://adh/apps/hive/warehouse/transactions, inputFormat:org.
apache.hadoop.mapred.TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false,
numBuckets:-1, serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:
{serialization.format=1}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:
[], skewedColValueLocationMaps:{}), storedAsSubDirectories:false), partitionKeys:[FieldSchema(name:txn_date, type:date,
comment:null)], parameters:{totalSize=120, numRows=9, rawDataSize=111, COLUMN_STATS_ACCURATE={\"BASIC_STATS\":\"true\"},
numFiles=4, numPartitions=4, transient_lastDdlTime=1702839122, bucketing_version=2}, viewOriginalText:null,
viewExpandedText:null, tableType:MANAGED_TABLE, rewriteEnabled:false, catName:hive, ownerType:USER) |          |
+-----------------------------+----------------------------------------------------+----------+
```

To gather statistics for a specific partition, consider the example below:

```sql
ANALYZE TABLE transactions PARTITION(txn_date='2023-01-03') COMPUTE STATISTICS;
DESCRIBE EXTENDED transactions PARTITION(txn_date='2023-01-03');
```

The partition statistics is present in the Detailed Partition Information block.

```
+---------------------------------+----------------------------------------------------+----------+
|            col_name             |                     data_type                      | comment  |
+---------------------------------+----------------------------------------------------+----------+
| txn_id                          | int                                                |          |
| acc_id                          | int                                                |          |
| txn_amount                      | decimal(10,2)                                      |          |
| txn_date                        | date                                               |          |
|                                 | NULL                                               | NULL     |
| # Partition Information         | NULL                                               | NULL     |
| # col_name                      | data_type                                          | comment  |
| txn_date                        | date                                               |          |
|                                 | NULL                                               | NULL     |
| Detailed Partition Information  | Partition(values:[2023-01-03], dbName:explain_demo, tableName:transactions, createTime:0,
lastAccessTime:0, sd:StorageDescriptor(cols:[FieldSchema(name:txn_id, type:int, comment:null), FieldSchema(name:acc_id, type:int,
comment:null), FieldSchema(name:txn_amount, type:decimal(10,2), comment:null), FieldSchema(name:txn_date, type:date,
comment:null)], location:hdfs://adh/apps/hive/warehouse/transactions/txn_date=2023-01-03, inputFormat:org.apache.hadoop.mapred.
TextInputFormat, outputFormat:org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat, compressed:false, numBuckets:-1,
serdeInfo:SerDeInfo(name:null, serializationLib:org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe, parameters:{serialization.
format=1}), bucketCols:[], sortCols:[], parameters:{}, skewedInfo:SkewedInfo(skewedColNames:[], skewedColValues:[],
skewedColValueLocationMaps:{}), storedAsSubDirectories:false), parameters:{totalSize=40, numRows=3, rawDataSize=37,
COLUMN_STATS_ACCURATE={"BASIC_STATS":"true","COLUMN_STATS":{"acc_id":"true","txn_amount":"true","txn_id":"true"}}, numFiles=1,
transient_lastDdlTime=1702844180}, catName:hive) |          |
+---------------------------------+----------------------------------------------------+----------+
```

Subqueries in Hive

Konstantin Alpashkin

Contents

To Table of Contents

- Test database
- Subqueries in the WHERE clause
  - Subqueries with IN
  - Subqueries with NOT IN
  - Subqueries with EXISTS
  - Subqueries with aggregate functions
- Subqueries in the FROM clause
  - Nested SELECT examples

A Hive subquery is a nested HiveQL statement that returns a result set during the execution of a parent (outer) query. As of ADH 3.1.2, Hive supports nesting subqueries inside FROM and WHERE clauses.

When Hive processes a query with a nested subquery, the subquery is evaluated first; then, its results are available in the parent query and can be referenced. The subquery mechanism is enabled by default and requires no manual activation.

Test database

Throughout this article, the following Hive tables `employees` and `departments` are used to demonstrate Hive subquery examples.

```
SELECT * FROM employees;
+--------------+----------------+---------------------+---------------------------+---------------+-------------------+
| employees.id | employees.name | employees.last_name |      employees.email      | employees.age | employees.dept_id |
+--------------+----------------+---------------------+---------------------------+---------------+-------------------+
| 1            | Ivan           | Ivanov              | ivan_ivanov123@mail.ru    | 30            | 1                 |
| 2            | Sarah          | Connor              | sarah_connor123@yahoo.com | 35            | 2                 |
| 3            | Rick           | Sanchez             | rick_123@mail.ru          | 29            | 2                 |
| 4            | John           | Smith               | john_smith123@gmail.com   | 50            | 3                 |
+--------------+----------------+---------------------+---------------------------+---------------+-------------------+

SELECT * FROM departments;
+-----------------+-----------------------+-------------------------+------------------------+
| departments.id  | departments.dep_name  | departments.dep_office  | departments.dep_phone  |
+-----------------+-----------------------+-------------------------+------------------------+
| 1               | sales                 | 110                     | 9379992                |
| 2               | it                    | 115                     | 12345678               |
| 3               | support               | 125                     | 880080080              |
+-----------------+-----------------------+-------------------------+------------------------+
```

SQL for creating test tables

The following SQL can be used to create and populate test Hive tables using */bin/beeline*.

```sql
CREATE DATABASE hr;

USE hr;

CREATE TABLE IF NOT EXISTS hr.employees (id int, name string, last_name string, email string, age int, salary int, dept_id int)
COMMENT 'Employee Table'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

INSERT INTO hr.employees VALUES
(1, 'Ivan', 'Ivanov', 'ivan_ivanov123@mail.ru', 30, 1000, 1),
(2, 'Sarah', 'Connor', 'sarah_connor123@yahoo.com', 35, 1500, 2),
(3, 'Rick', 'Sanchez', 'rick_123@mail.ru', 29, 1300, 2),
(4, 'John', 'Smith', 'john_smith123@gmail.com', 29, 2000, 3);

CREATE TABLE IF NOT EXISTS hr.departments (id int, dep_name string, dep_office int, dep_phone string)
COMMENT 'Departments Table'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

INSERT INTO hr.departments VALUES
(1, 'sales', 110, '9379992'),
(2, 'it', 115, '12345678'),
(3, 'support', 125, '880080080');
```

Subqueries in the WHERE clause

Only one subquery is allowed inside a `WHERE` clause. A subquery is called **correlated** if it contains a query predicate with the `=` operator and one side of the operator references at least one column from the parent query and the other side references at least one column from the subquery.

When used inside `WHERE`, the result of a subquery can be treated as input for `IN`/`NOT IN` statements.

Subqueries with IN

In the example below, the subquery gets department IDs from the `employees` table using a specific office number. Then, the outer query selects employees who work in this department.

```sql
SELECT e.name, e.last_name, e.email FROM employees e
WHERE e.dept_id IN
    (SELECT id FROM departments
    WHERE dep_office=110);
```

The result:

```
+---------+--------------+-------------------------+
| e.name  | e.last_name  |         e.email         |
+---------+--------------+-------------------------+
| Ivan    | Ivanov       | ivan_ivanov123@mail.ru  |
+---------+--------------+-------------------------+
```

Subqueries with NOT IN

In the example below, the subquery gets a department ID by a specific office number. Then, the outer query retrieves information about employees who are not located in the given office.

```sql
 SELECT e.name, e.last_name, e.email
 FROM employees e
 WHERE e.dept_id NOT IN
       (SELECT d.id
        FROM departments d
        WHERE d.dep_office=110)
```

The result:

```
+---------+--------------+----------------------------+
| e.name  | e.last_name  |          e.email           |
+---------+--------------+----------------------------+
| Sarah   | Connor       | sarah_connor123@yahoo.com  |
| Rick    | Sanchez      | rick_123@mail.ru           |
| John    | Smith        | john_smith123@gmail.com    |
+---------+--------------+----------------------------+
```

Subqueries with EXISTS

The following sample query returns office numbers of those employees who are of the same specific age. First, the subquery selects employees by age, and then the selection results are picked by `EXISTS`. For each record returned by the subquery, the outer query fetches the corresponding office number from the `departments` table.

```sql
SELECT d.dep_office
FROM departments d
WHERE EXISTS (
    SELECT * FROM employees e
    WHERE e.dept_id=d.id AND e.age=29
    );
```

The result:

```
+---------------+
| d.dep_office  |
+---------------+
| 115           |
| 125           |
+---------------+
```

Subqueries with aggregate functions

Hive provides limited support for aggregate functions within subqueries. The following sample query returns employees whose salary is higher than the average.

```sql
SELECT e.last_name, e.salary FROM employee e
WHERE salary > (
    SELECT AVG(salary)
    FROM employee);
```

The result:

```
+--------------+-----------+
| e.last_name  | e.salary  |
+--------------+-----------+
| Smith        | 2000      |
| Connor       | 1500      |
+--------------+-----------+
```

Subqueries in the FROM clause

Nested `SELECT` subqueries are typically used inside `FROM` clauses to fetch a draft dataset from one or more tables and then have it filtered/sort/modified in the outer query.

The number of subqueries nested inside a `FROM` clause can be arbitrary. `SELECT` subqueries nested inside `FROM` must be named since every table inside `FROM` must have a name. All columns in the subquery selection list must have unique names and are available in the outer query like columns of a table. For example:

```sql
SELECT t.name, t.email
FROM (SELECT e.name, e.email
      FROM employees e
      WHERE e.age=35) t; ①
```

① An alias (`t`) is required to qualify the nested `SELECT` query. This qualifier can be used in the parent query to reference columns selected by the subquery.

Nested SELECT examples

In the example below, the subquery fetches employees data from explicitly joined `employees` and `departments` tables. Then, the outer query assembles new email addresses using the columns fetched by the subquery.

```sql
SELECT t.old_email, concat(t.name, t.last_name, '_', t.department_name, '@new.domain.org') AS new_email
FROM
    (SELECT e.name, e.last_name, e.email AS old_email, d.dep_name AS department_name
     FROM employee e
        JOIN departments d ON e.dept_id=d.id
     ) t;
```

The result:

```
+----------------------------+-----------------------------------+
|        t.old_email         |            new_email              |
+----------------------------+-----------------------------------+
| sarah_connor123@yahoo.com  | SarahConnor_it@new.domain.org     |
| rick_123@mail.ru           | RickSanchez_it@new.domain.org     |
| john_smith123@gmail.com    | JohnSmith_support@new.domain.org  |
| ivan_ivanov123@mail.ru     | IvanIvanov_sales@new.domain.org   |
+----------------------------+-----------------------------------+
```

Aggregate functions in Hive

Konstantin Alpashkin

Contents

To Table of Contents

- Test database
- Basic aggregation
  - More Hive aggregation functions
- Advanced aggregation
  - GROUPING SETS
  - WITH CUBE
  - WITH ROLLUP

Hive aggregate functions are built-in functions that accept a set of values and return a single value. This section provides examples and tips on using basic and advanced Hive aggregation features. The full list of aggregate functions supported in Hive by default is available in Apache Hive documentation.

Test database

Throughout this article, the following Hive tables `employees` and `transactions` are used to demonstrate aggregation examples.

```
SELECT * FROM employees;
+---------------+-----------------+----------------------+---------------------------+----------------+--------------------+-----
---------------+
| employees.id  | employees.name  | employees.last_name  |      employees.email       | employees.age  | employees.salary  |
employees.dept_id  |
+---------------+-----------------+----------------------+---------------------------+----------------+--------------------+-----
---------------+
| 1             | Ivan            | Ivanov               | ivan_ivanov123@mail.ru     | 30             | 1000               | 1
|
| 2             | Sarah           | Connor               | sarah_connor123@yahoo.com  | 35             | 1500               | 2
|
| 3             | Rick            | Sanchez              | rick_123@mail.ru           | 29             | 1300               | 2
|
| 4             | John            | Smith                | john_smith123@gmail.com    | 29             | 2000               | 3
|
+---------------+-----------------+----------------------+---------------------------+----------------+--------------------+-----
---------------+


SELECT * FROM transactions;
+-----------------------+----------------------+--------------------------+------------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  |
+-----------------------+----------------------+--------------------------+------------------------+
| 1                     | 101                  | 10.20                    | 2023-10-26             |
| 2                     | 101                  | 102.30                   | 2023-10-26             |
| 3                     | 102                  | 95.00                    | 2023-10-26             |
| 4                     | 103                  | 900.50                   | 2023-10-27             |
| 5                     | 102                  | 910.50                   | 2023-10-27             |
| 6                     | 101                  | 850.00                   | 2023-10-27             |
+-----------------------+----------------------+--------------------------+------------------------+
```

SQL for creating test tables

The following SQL can be used to create and populate test Hive tables using */bin/beeline*.

```sql
CREATE TABLE IF NOT EXISTS employees (id int, name string, last_name string, email string, age int, salary int, dept_id int)
COMMENT 'Employee Table'
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';

INSERT INTO employees VALUES
(1, 'Ivan', 'Ivanov', 'ivan_ivanov123@mail.ru', 30, 1000, 1),
(2, 'Sarah', 'Connor', 'sarah_connor123@yahoo.com', 35, 1500, 2),
(3, 'Rick', 'Sanchez', 'rick_123@mail.ru', 29, 1300, 2),
(4, 'John', 'Smith', 'john_smith123@gmail.com', 29, 2000, 3);


CREATE TABLE IF NOT EXISTS transactions(txn_id int, acc_id int, txn_amount decimal(10,2), txn_date date)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe';

INSERT INTO transactions VALUES
(1, 101, 10.20, '2023-10-26'),
(2, 101, 102.30, '2023-10-26'),
(3, 102, 95.00, '2023-10-26'),
(4, 103, 900.50, '2023-10-27'),
(5, 102, 910.50, '2023-10-27'),
(6, 101, 850.00, '2023-10-27');
```

Basic aggregation

The `COUNT(col)` function returns the total number of rows in the result set.

```sql
SELECT COUNT(id) AS emp_total_cnt FROM employees;
```

The result:

```
+-----------------+
| emp_total_cnt  |
+-----------------+
| 4               |
+-----------------+
```

TIP

Using the `DISTINCT` keyword, you can perform aggregation only on unique column values.

The `MIN()` and `MAX()` functions return the minimum/maximum value of a numeric column.

```sql
SELECT MIN(salary) from employees;
```

The output:

```
+--------+
|  _c0  |
+-------+
| 1000  |
+-------+
```

The `AVG()` function returns an average value of a numeric column. Currently, aggregate functions are not supported inside the `WHERE` clause. For example, the following query will produce an error.

```sql
SELECT e.salary, e.age
FROM employees e
WHERE e.age > AVG(e.age);
```

The error message:

```
Error: Error while compiling statement: FAILED: SemanticException [Error 10128]: Line 3:14 Not yet supported place for UDAF 'avg'
(state=42000,code=10128)
```

As a workaround for this case, you can nest `AVG()` inside a subquery as shown below.

```sql
SELECT e.age, e.last_name
FROM employees e
WHERE e.age > (SELECT AVG(age) FROM employees);
```

The successful result:

```
+--------+--------------+
| e.age  | e.last_name  |
+--------+--------------+
| 35     | Connor       |
+--------+--------------+
```

More Hive aggregation functions

Apart from standard SQL aggregation functions like `SUM`, `AVG`, etc., Hive provides other functions that may come in handy for analyzing your datasets. For example, using `collect_*(col)`, you can collapse retrieved column values to a list or a set. Consider the example below.

```sql
SELECT collect_set(age) FROM employees; ①
SELECT d.dep_name AS department_name, collect_list(e.age) AS ages ②
FROM employees e
    JOIN departments d
    ON e.dept_id=d.id
GROUP BY d.dep_name;
```

① Returns a single row with a set of unique values.

② For each department, returns a list of values.

The result:

```
+-------------+
|     _c0     |
+-------------+
| [30,35,29]  |
+-------------+


+------------------+----------+
| department_name  |   ages   |
+------------------+----------+
| it               | [35,29]  |
| sales            | [30]     |
| support          | [29]     |
+------------------+----------+
```

Hive also provides special functions for calculating mathematical entities like deviation, variance, and many others. A few examples below.

```sql
SELECT variance(salary) FROM employees; ①
SELECT var_pop(salary) FROM employees; ①
SELECT var_samp(salary) FROM employees; ②
SELECT stddev_pop(salary) FROM employees; ③
SELECT stddev_samp(salary) FROM employees; ④
```

① The queries return the statistical variance of the numeric `salary` column.

② Returns the unbiased sample variance for the column.

③ Returns the statistical standard deviation of all values for the column.

④ Returns the unbiased sample standard deviation of the numeric column.

The result:

```
SELECT variance(salary) FROM employees;
+-----------+
|    _c0    |
+-----------+
| 132500.0  |
+-----------+

SELECT var_pop(salary) FROM employees;
+-----------+
|    _c0    |
+-----------+
| 132500.0  |
+-----------+

SELECT var_samp(salary) FROM employees;
+---------------------+
|         _c0         |
+---------------------+
| 176666.66666666666  |
+---------------------+

SELECT stddev_pop(salary) FROM employees;
+--------------------+
|        _c0         |
+--------------------+
| 364.0054944640259  |
+--------------------+

SELECT stddev_samp(salary) FROM employees;
+---------------------+
|         _c0         |
+---------------------+
| 420.31734043061635  |
+---------------------+
```

The full list of Hive aggregate functions is available in Apache Hive documentation.

Advanced aggregation

GROUPING SETS

The `GROUPING SETS` clause used with `GROUP BY` allows you to apply several `GROUP BY` criteria for the same record set. Using several grouping sets (e.g. `GROUPING SETS (a,b)`) is equivalent to running multiple `GROUP BY a`, `GROUP BY b` queries united by `UNION`. However, fetching results via grouping sets is done in one table scan as opposed to running several queries with different `GROUP BY` clauses.

The following example specifies a single grouping set (`GROUPING SETS (acc_id)`), thus, aggregating the results by account ID and not by transactions date.

```sql
SELECT acc_id, txn_date, SUM(txn_amount) as total_paid
FROM transactions
GROUP BY acc_id, txn_date
GROUPING SETS (acc_id);
```

The result of such query is the total amount paid by each account. `NULL` values in the `txn_date` column indicate that no aggregation by date was applied.

```
+---------+-----------+-------------+
| acc_id  | txn_date  | total_paid  |
+---------+-----------+-------------+
| 101     | NULL      | 962.50      |
| 102     | NULL      | 1005.50     |
| 103     | NULL      | 900.50      |
+---------+-----------+-------------+
```

The following example specifies multiple grouping sets (`GROUPING SETS (acc_id, txn_date)`).

```sql
SELECT acc_id, txn_date, SUM(txn_amount) as total_paid, grouping__id
FROM transactions
GROUP BY acc_id, txn_date
GROUPING SETS (acc_id, txn_date);
```

In this case, the result set consists of two subsets grouped by `txn_date` and `acc_id`:

```
+---------+-------------+-------------+
| acc_id  |  txn_date   | total_paid  |
+---------+-------------+-------------+
| NULL    | 2023-10-26  | 207.50      |
| NULL    | 2023-10-27  | 2661.00     |
| 101     | NULL        | 962.50      |
| 102     | NULL        | 1005.50     |
| 103     | NULL        | 900.50      |
+---------+-------------+-------------+
```

`NULL` values indicate that no aggregation was performed on the given column. Notice that the above result is a `UNION` of two result sets, returned by the queries like:

```sql
SELECT NULL, txn_date, SUM(txn_amount)
FROM transactions
GROUP BY txn_date
UNION
SELECT acc_id, NULL, SUM(txn_amount)
FROM transactions
GROUP BY acc_id;
```

IMPORTANT

Running individual queries with `UNION` will launch 2 jobs instead of 1 and can potentially take longer to execute.

WITH CUBE

The `CUBE` function is used with `GROUP BY` to calculate all possible combinations of column groups specified in the `GROUP BY` clause. The following HiveQL constructs are identical:

```sql
... GROUP BY a, b, c WITH CUBE
-- is equivalent to:
... GROUP BY a, b, c GROUPING SETS (
    (a, b, c), (a, b), (b, c), (a, c), (a), (b), (c), ( )
)
```

Consider the example below.

```sql
SELECT acc_id, txn_date, SUM(txn_amount) as total_sum
FROM transactions
GROUP BY acc_id, txn_date WITH CUBE;
```

The result includes records for all possible aggregation groups:

```
+---------+-------------+-------------+
| acc_id  |  txn_date   | total_sum   |
+---------+-------------+-------------+
| NULL    | NULL        | 2868.50     |
| NULL    | 2023-10-26  | 207.50      |
| NULL    | 2023-10-27  | 2661.00     |
| 101     | NULL        | 962.50      |
| 101     | 2023-10-26  | 112.50      |
| 101     | 2023-10-27  | 850.00      |
| 102     | NULL        | 1005.50     |
| 102     | 2023-10-26  | 95.00       |
| 102     | 2023-10-27  | 910.50      |
| 103     | NULL        | 900.50      |
| 103     | 2023-10-27  | 900.50      |
+---------+-------------+-------------+
```

In this example, using `WITH CUBE` is equivalent to the following query:

```sql
SELECT acc_id, txn_date, SUM(txn_amount) as total_sum
FROM transactions
GROUP BY acc_id, txn_date
GROUPING SETS ((acc_id, txn_date), acc_id, txn_date, ( ));
```

WITH ROLLUP

The `ROLLUP` function is used with `GROUP BY` to calculate multiple levels of subtotals across a specified group of dimensions. The function creates subtotals which "roll up" from the most detailed level to the most generic one, following a grouping list specified in the `GROUP BY` clause. The following HiveQL constructs are identical:

```sql
... GROUP BY a, b, c, WITH ROLLUP
-- is equivalent to:
... GROUP BY a, b, c GROUPING SETS ( (a, b, c), (a, b), (a), ( ))
```

Consider the example below.

```sql
SELECT acc_id, txn_date, SUM(txn_amount) as total_sum
FROM transactions
GROUP BY acc_id, txn_date WITH ROLLUP;
```

The result set includes records with different aggregation levels — starting from the most detailed groups (money paid by a specific account on a specific day) up to the most generic one (total money paid):

```
+---------+-------------+------------+
| acc_id  |  txn_date   | total_sum  |
+---------+-------------+------------+
| NULL    | NULL        | 2868.50    |
| 101     | NULL        | 962.50     |
| 101     | 2023-10-26  | 112.50     |
| 101     | 2023-10-27  | 850.00     |
| 102     | NULL        | 1005.50    |
| 102     | 2023-10-26  | 95.00      |
| 102     | 2023-10-27  | 910.50     |
| 103     | NULL        | 900.50     |
| 103     | 2023-10-27  | 900.50     |
+---------+-------------+------------+
```

In this example, using `ROLLUP` is equivalent to the following query:

```sql
SELECT acc_id, txn_date, SUM(txn_amount) as total_sum
FROM transactions
GROUP BY acc_id, txn_date
GROUPING SETS ((acc_id, txn_date), acc_id, ( ));
```

Window and analytical functions in Hive

Konstantin Alpashkin

Contents

To Table of Contents

- Test database
- Syntax
  - Window specification
- OVER keyword
  - OVER + PARTITION BY
  - OVER + ORDER BY
- Window functions
  - LEAD and LAG
  - ROW_NUMBER
  - FIRST_VALUE and LAST_VALUE
- Analytical functions
  - RANK and DENSE_RANK
  - PERCENT_RANK
  - NTILE

Hive window and analytical functions are used to perform calculations over a set of related rows (windows or partitions) in a result set. Unlike aggregate functions that collapse all column values to a single value, the outcome of a window function is one or more columns with calculation results added to the original result set. Window and analytical functions are extremely useful for analytical tasks and are essential solution to queries like "get top X products sold by category", "get maximum salary per department", "get first N rows from the result set", and alike.

All functions can be divided into the following groups:

- Window functions, such as `LEAD`, `LAG`, `FIRST_VALUE`, etc.
- Analytical functions like `RANK`, `ROW_NUMBER`, `NTILE`, etc.

This section provides examples and recommendations on using major functions, more details on supported Hive window and analytical functions are available in Apache Hive documentation.

Test database

Throughout this article, the following Hive table `transactions` is used to demonstrate the examples.

```
SELECT * FROM transactions;
+----------------------+----------------------+--------------------------+------------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  |
+----------------------+----------------------+--------------------------+------------------------+
| 1                    | 1002                 | 10.00                    | 2023-01-01             |
| 2                    | 1002                 | 20.00                    | 2023-01-03             |
| 3                    | 1002                 | 30.00                    | 2023-01-02             |
| 4                    | 1001                 | 100.50                   | 2023-01-02             |
| 5                    | 1001                 | 150.50                   | 2023-01-04             |
| 6                    | 1001                 | 200.50                   | 2023-01-03             |
| 7                    | 1003                 | 50.00                    | 2023-01-03             |
| 8                    | 1003                 | 50.00                    | 2023-01-01             |
| 9                    | 1003                 | 75.00                    | 2023-01-04             |
+----------------------+----------------------+--------------------------+------------------------+
```

SQL for creating test table

The following SQL can be used to create and populate the test Hive table using */bin/beeline*.

```
DROP TABLE IF EXISTS `transactions`;
CREATE TABLE `transactions`(`txn_id` int, `acc_id` int, `txn_amount` decimal(10,2), `txn_date` date);
INSERT INTO transactions VALUES
(1, 1002, 10.00, '2023-01-01'),
(2, 1002, 20.00, '2023-01-03'),
(3, 1002, 30.00, '2023-01-02'),
(4, 1001, 100.50, '2023-01-02'),
(5, 1001, 150.50, '2023-01-04'),
(6, 1001, 200.50, '2023-01-03'),
(7, 1003, 50.00, '2023-01-03'),
(8, 1003, 50.00, '2023-01-01'),
(9, 1003, 75.00, '2023-01-04');
```

Syntax

The basic syntax for using window and analytical functions is as follows:

```
SELECT <func>([[, ...]]) OVER ([<window_spec>]) [<window_name>]
FROM <table_name>;
```

Where:

- `<func>` — a standard SQL aggregation function (e.g. `AVG`) or a special window/analytical function.
- `OVER` — indicates that the function should be calculated for specific windows within the result set.
- `<window_spec>` — an expression that defines the criteria to split the result set into windows. More details on windows specification are available below.
- `<window_name>` — an alias for the column with the function calculation results.

Window specification

A window specification tells Hive how to divide the result set fetched by `SELECT` into windows. An empty specification indicates that the entire result set should be treated as a single window and the function should run for all the rows. The window specification syntax is as follows:

```
[PARTITION BY <col> [, ...]]
[ORDER BY <col> [ASC|DESC] [, ...]]
(ROWS | RANGE) BETWEEN (UNBOUNDED | [num]) PRECEDING AND ([num] PRECEDING | CURRENT ROW | (UNBOUNDED | [num]) FOLLOWING)
```

Where:

- `PARTITION BY <col>` — splits the result set into partitions based on the `<col>` values.
- `ORDER BY <col>` — orders the rows within each partition before running the function.
- `ROWS | RANGE …` — specifies the upper/lower window frame bounds.

There is also an alternative syntax with placing windows specification in a separate `WINDOW` clause:

```
SELECT <func>([[, ...]]) OVER <window_name>
FROM <table_name>
WINDOW <window_name> AS ([<window_spec>])
```

OVER keyword

At its simplest, a window function is defined by adding the `OVER` clause to a `SELECT` query. A typical selection may include window/analytical functions or standard SQL aggregate functions (`SUM`, `AVG`, etc.).

In the following example, the empty `OVER()` clause indicates that no explicit window is specified, so the entire result set returned by `SELECT` should be treated as a window to calculate the average transaction amount.

```
SELECT *, AVG(txn_amount)
OVER() AS average_txn
FROM transactions;
```

The result contains an extra `average_txn` column that holds the average value of all transactions:

```
+----------------------+----------------------+--------------------------+------------------------+--------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  | average_txn  |
+----------------------+----------------------+--------------------------+------------------------+--------------+
| 1                    | 1002                 | 10.00                    | 2023-01-01             | 76.277778    |
| 2                    | 1002                 | 20.00                    | 2023-01-03             | 76.277778    |
| 3                    | 1002                 | 30.00                    | 2023-01-02             | 76.277778    |
| 4                    | 1001                 | 100.50                   | 2023-01-02             | 76.277778    |
| 5                    | 1001                 | 150.50                   | 2023-01-04             | 76.277778    |
| 6                    | 1001                 | 200.50                   | 2023-01-03             | 76.277778    |
| 7                    | 1003                 | 50.00                    | 2023-01-03             | 76.277778    |
| 8                    | 1003                 | 50.00                    | 2023-01-01             | 76.277778    |
| 9                    | 1003                 | 75.00                    | 2023-01-04             | 76.277778    |
+----------------------+----------------------+--------------------------+------------------------+--------------+
```

OVER + PARTITION BY

By using the `OVER (PARTITION BY col)` construct, you can split the result set into partitions so the function logic is applied for each individual partition.

For example, the following query calculates average transaction amount with respect to each account.

```
SELECT *, AVG(txn_amount)
OVER (PARTITION BY acc_id) AS average_txn_per_acc
FROM transactions;
```

The result set now includes average values calculated for individual accounts:

```
+----------------------+----------------------+--------------------------+------------------------+----------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  | average_txn_per_acc  |
+----------------------+----------------------+--------------------------+------------------------+----------------------+
| 4                    | 1001                 | 100.50                   | 2023-01-02             | 150.500000           |
| 5                    | 1001                 | 150.50                   | 2023-01-04             | 150.500000           |
| 6                    | 1001                 | 200.50                   | 2023-01-03             | 150.500000           |
| 1                    | 1002                 | 10.00                    | 2023-01-01             | 20.000000            |
| 2                    | 1002                 | 20.00                    | 2023-01-03             | 20.000000            |
| 3                    | 1002                 | 30.00                    | 2023-01-02             | 20.000000            |
| 7                    | 1003                 | 50.00                    | 2023-01-03             | 58.333333            |
| 8                    | 1003                 | 50.00                    | 2023-01-01             | 58.333333            |
| 9                    | 1003                 | 75.00                    | 2023-01-04             | 58.333333            |
+----------------------+----------------------+--------------------------+------------------------+----------------------+
```

OVER + ORDER BY

With the `ORDER` keyword, you can add sorting to your function.

For example, the following query calculates the total amount paid by each account at a specific day, sorting the rows by date.

```
SELECT *, SUM(txn_amount)
OVER (PARTITION BY acc_id ORDER BY txn_date) AS running_total_paid
FROM transactions;
```

For each account ID, the rows in the final result set are sorted by transaction date. Notice that the `running_total_paid` column stores running total values. This occurs because adding `ORDER BY` to the window specification implicitly uses the default `RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW` boundary. Thus, when calculating an average for each row, the function uses the current row value + all previous values. Without `ORDER BY` the total sum in `running_total_paid` would be the same for all the records.

```
+----------------------+----------------------+--------------------------+------------------------+---------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  | running_total_paid  |
+----------------------+----------------------+--------------------------+------------------------+---------------------+
| 4                    | 1001                 | 100.50                   | 2023-01-02             | 100.50              |
| 6                    | 1001                 | 200.50                   | 2023-01-03             | 301.00              |
| 5                    | 1001                 | 150.50                   | 2023-01-04             | 451.50              |
| 1                    | 1002                 | 10.00                    | 2023-01-01             | 10.00               |
| 3                    | 1002                 | 30.00                    | 2023-01-02             | 40.00               |
| 2                    | 1002                 | 20.00                    | 2023-01-03             | 60.00               |
| 8                    | 1003                 | 50.00                    | 2023-01-01             | 50.00               |
| 7                    | 1003                 | 50.00                    | 2023-01-03             | 100.00              |
| 9                    | 1003                 | 75.00                    | 2023-01-04             | 175.00              |
+----------------------+----------------------+--------------------------+------------------------+---------------------+
```

Window functions

LEAD and LAG

The `LEAD`/`LAG` are similar functions that allow you to pull data from next/previous rows respectively. These can be useful for comparing adjacent column values or for calculating the difference between rows.

The following query groups the selection results by account, and for each row, returns an ID of the next transaction available in the partition group.

```
SELECT txn_id, txn_amount, txn_date, LEAD(txn_id)
OVER (PARTITION BY acc_id ORDER BY txn_date) AS next_txn_id
FROM transactions;
```

The result is shown below. `NULL` values in the `next_txn_id` indicate the window boundary i.e. no more next rows are available for the current partition group.

```
+---------+-------------+-------------+--------------+
| txn_id  | txn_amount  |  txn_date   | next_txn_id  |
+---------+-------------+-------------+--------------+
| 4       | 100.50      | 2023-01-02  | 6            |
| 6       | 200.50      | 2023-01-03  | 5            |
| 5       | 150.50      | 2023-01-04  | NULL         |
| 1       | 10.00       | 2023-01-01  | 3            |
| 3       | 30.00       | 2023-01-02  | 2            |
| 2       | 20.00       | 2023-01-03  | NULL         |
| 8       | 50.00       | 2023-01-01  | 7            |
| 7       | 50.00       | 2023-01-03  | 9            |
| 9       | 75.00       | 2023-01-04  | NULL         |
+---------+-------------+-------------+--------------+
```

By default, `LEAD` and `LAG` functions return column values that are 1 step away from the current row and return `NULL` if the partition boundary is reached. You can change the offset and the out-of-boundary indicating value using the following signature:

```
LEAD(col, i, j)
LAG(col, i, j)
```

Where:

- `col` — the name of a column to get the next/previous value.
- `i` — the offset to get the next/previous value.
- `j` — the value to be returned instead of `NULL` when reaching a partition boundary.

ROW_NUMBER

The `ROW_NUMBER()` function assigns a unique sequence number to each row within a partition according to the partition and order rules.

The following sample query shows how to get the first 2 transactions committed by each account.

```
SELECT * FROM
     (SELECT *, ROW_NUMBER()
      OVER (PARTITION BY acc_id ORDER BY txn_date) AS row_num
      FROM transactions) tx
WHERE tx.row_num < 3;
```

The result `tx.row_num` column indicates the first two transactions committed by each account chronologically:

```
+------------+------------+----------------+--------------+-------------+
| tx.txn_id  | tx.acc_id  | tx.txn_amount  | tx.txn_date  | tx.row_num  |
+------------+------------+----------------+--------------+-------------+
| 4          | 1001       | 100.50         | 2023-01-02   | 1           |
| 6          | 1001       | 200.50         | 2023-01-03   | 2           |
| 1          | 1002       | 10.00          | 2023-01-01   | 1           |
| 3          | 1002       | 30.00          | 2023-01-02   | 2           |
| 8          | 1003       | 50.00          | 2023-01-01   | 1           |
| 7          | 1003       | 50.00          | 2023-01-03   | 2           |
+------------+------------+----------------+--------------+-------------+
```

FIRST_VALUE and LAST_VALUE

The `FIRST_VALUE`/`LAST_VALUE` functions return a column value from the first/last row of a partition.

In the example below, the query gets the lowest transaction ID values for each account partition.

```
SELECT txn_id, acc_id, FIRST_VALUE(txn_id)
OVER (PARTITION BY acc_id) lowest_id_per_acc
FROM transactions;
```

The result:

```
+---------+---------+--------------------+
| txn_id  | acc_id  | lowest_id_per_acc  |
+---------+---------+--------------------+
| 4       | 1001    | 4                  |
| 5       | 1001    | 4                  |
| 6       | 1001    | 4                  |
| 1       | 1002    | 1                  |
| 2       | 1002    | 1                  |
| 3       | 1002    | 1                  |
| 7       | 1003    | 7                  |
| 8       | 1003    | 7                  |
| 9       | 1003    | 7                  |
+---------+---------+--------------------+
```

Both `FIRST_VALUE`/`LAST_VALUE` functions have an optional Boolean parameter that defaults to `false`. If set to `true`, Hive will ignore rows with `NULL` column values.

Analytical functions

RANK and DENSE_RANK

The `RANK` function assigns rank to each row in a partition. In case of duplicates, the function returns equal ranks for such rows, skipping the subsequent rank number.

In the following example, `RANK` calculates the rank of each transaction among all other transactions (no `PARTITION BY` specified). To calculate the transaction ranks for each account, the `PARTITION BY acc_id` expression should be added.

```
SELECT txn_amount, acc_id, RANK()
OVER (ORDER BY txn_amount DESC) rank_global
-- per-account rank:
-- OVER (PARTITION BY acc_id ORDER BY txn_amount DESC) rank_per_acc
FROM transactions;
```

The result:

```
+-------------+---------+--------------+
| txn_amount  | acc_id  | rank_global  |
+-------------+---------+--------------+
| 200.50      | 1001    | 1            |
| 150.50      | 1001    | 2            |
| 100.50      | 1001    | 3            |
| 75.00       | 1003    | 4            |
| 50.00       | 1003    | 5            |
| 50.00       | 1003    | 5            |
| 30.00       | 1002    | 7            |
| 20.00       | 1002    | 8            |
| 10.00       | 1002    | 9            |
+-------------+---------+--------------+
```

Notice that the result set has two transactions with the same rank (`5`) as their `txn_amount` values are equal. Since these two rows have been assigned the same rank `5`, the subsequent rank number `6` is absent in the result set, and the next available rank number is `7`. To avoid such "holes" in the ranks sequence, use the `DENSE_RANK` function. The previous example returns the following result if `DENSE_RANK` is used instead of `RANK`:

```
+-------------+---------+--------------+
| txn_amount  | acc_id  | rank_global  |
+-------------+---------+--------------+
| 200.50      | 1001    | 1            |
| 150.50      | 1001    | 2            |
| 100.50      | 1001    | 3            |
| 75.00       | 1003    | 4            |
| 50.00       | 1003    | 5            |
| 50.00       | 1003    | 5            |
| 30.00       | 1002    | 6            |
| 20.00       | 1002    | 7            |
| 10.00       | 1002    | 8            |
+-------------+---------+--------------+
```

PERCENT_RANK

The `PERCENT_RANK` function calculates a percentage rank (a number from 0 to 1 inclusive) for each row within a partition group. The `ORDER BY` clause is mandatory in the window specification. The function produces a column with an ascending numeric sequence that represents the position of each row within a partition. The first row in each partition group always has the value of `0`, and the last row is always `1`.

For example, the following query ranks the transaction committed by the same account.

```
SELECT txn_id, acc_id, txn_amount, PERCENT_RANK()
OVER (PARTITION BY acc_id ORDER BY txn_amount) position_within_acc
FROM transactions;
```

The result set contains an extra column `position_within_acc` with a ranking number:

```
+---------+---------+-------------+-------------------+
| txn_id  | acc_id  | txn_amount  |position_within_acc|
+---------+---------+-------------+-------------------+
| 4       | 1001    | 100.50      | 0.0               |
| 5       | 1001    | 150.50      | 0.5               |
| 6       | 1001    | 200.50      | 1.0               |
| 1       | 1002    | 10.00       | 0.0               |
| 2       | 1002    | 20.00       | 0.5               |
| 3       | 1002    | 30.00       | 1.0               |
| 7       | 1003    | 50.00       | 0.0               |
| 8       | 1003    | 50.00       | 0.0               |
| 9       | 1003    | 75.00       | 1.0               |
+---------+---------+-------------+-------------------+
```

NTILE

The `NTILE(n)` function divides an ordered result set into `n` roughly equal parts (buckets) and for each row, assigns an integer, indicating which bucket the given row falls into. The `ORDER BY` clause is mandatory in the window specification. With this universal function, you can calculate statistical notions like median, percentile, quartile, etc.

The following example splits all the transactions into 4 groups (quartiles) based on the transaction amount and identifies which quartile each row falls into. Notice that the partition gets sorted by `txn_amount` from lowest to highest, thus, smallest transactions belong to the first quartile.

```
SELECT txn_id, txn_amount, NTILE(4)
OVER (ORDER BY txn_amount) AS quartile
FROM transactions;
```

The result:

```
+---------+-------------+-----------+
| txn_id  | txn_amount  | quartile  |
+---------+-------------+-----------+
| 1       | 10.00       | 1         |
| 2       | 20.00       | 1         |
| 3       | 30.00       | 1         |
| 7       | 50.00       | 2         |
| 8       | 50.00       | 2         |
| 9       | 75.00       | 3         |
| 4       | 100.50      | 3         |
| 5       | 150.50      | 4         |
| 6       | 200.50      | 4         |
+---------+-------------+-----------+
```

Use materialized views in Hive

Konstantin Alpashkin

Contents

To Table of Contents

- Test database
- Syntax
- Query rewriting
- Rebuild materialized view
- Usage example

A materialized view (MV) is a database object that stores results of a query and serves these precomputed results later instead of fetching the data from the database. Typically, MVs are used to boost the performance by "caching" the results of long-running queries with complex aggregations, joins, filters, etc.

This section describes basic MV operations with usage examples. More information about MVs in Hive can be found in Apache Hive documentation.

Test database

Throughout this article, the following Hive tables `employees` and `departments` are used to demonstrate MV usage examples.

```
SELECT * FROM employees;
+---------------+-----------------+----------------------+----------------------------+----------------+-------------------+-----
---------------+
| employees.id  | employees.name  | employees.last_name  |      employees.email       | employees.age  | employees.salary  |
employees.dept_id  |
+---------------+-----------------+----------------------+----------------------------+----------------+-------------------+-----
---------------+
| 1             | Ivan            | Ivanov               | ivan_ivanov123@mail.ru     | 30             | 1000              | 1
|
| 2             | Sarah           | Connor               | sarah_connor123@yahoo.com  | 35             | 1500              | 2
|
| 3             | Rick            | Sanchez              | rick_123@mail.ru           | 29             | 1300              | 2
|
| 4             | John            | Smith                | john_smith123@gmail.com    | 29             | 2000              | 3
|
+---------------+-----------------+----------------------+----------------------------+----------------+-------------------+-----
---------------+

SELECT * FROM departments;
+-----------------+-----------------------+-------------------------+------------------------+
| departments.id  | departments.dep_name  | departments.dep_office  | departments.dep_phone  |
+-----------------+-----------------------+-------------------------+------------------------+
| 1               | sales                 | 110                     | 9379992                |
| 2               | it                    | 115                     | 12345678               |
| 3               | support               | 125                     | 880080080              |
+-----------------+-----------------------+-------------------------+------------------------+
```

SQL for creating test table

The following SQL can be used to create and populate the test Hive table using */bin/beeline*.

```
CREATE DATABASE IF NOT EXISTS mv_demo;
USE mv_demo;

DROP TABLE IF EXISTS employees;
CREATE TABLE employees (id int, name string, last_name string, email string, age int, salary int, dept_id int)
 STORED AS ORC
 TBLPROPERTIES ('transactional'='true');

INSERT INTO employees VALUES
(1, 'Ivan', 'Ivanov', 'ivan_ivanov123@mail.ru', 30, 1000, 1),
(2, 'Sarah', 'Connor', 'sarah_connor123@yahoo.com', 35, 1500, 2),
(3, 'Rick', 'Sanchez', 'rick_123@mail.ru', 29, 1300, 2),
(4, 'John', 'Smith', 'john_smith123@gmail.com', 29, 2000, 3);

DROP TABLE IF EXISTS departments;
CREATE TABLE departments (id int, dep_name string, dep_office int, dep_phone string)
STORED AS ORC
TBLPROPERTIES ('transactional'='true');

INSERT INTO departments VALUES
(1, 'sales', 110, '9379992'),
(2, 'it', 115, '12345678'),
(3, 'support', 125, '880080080');
```

Syntax

Since technically an MV is an ordinary Hive table (yet with several limitations), the basic syntax for creating MVs is similar to creating Hive tables. The MV creation syntax is as follows:

```
CREATE MATERIALIZED VIEW [IF NOT EXISTS] [<db_name>.]<mv_name>
  [DISABLE REWRITE]
  [COMMENT <comment>]
  [PARTITIONED ON (<col_name>, ...)]
  [CLUSTERED ON (<col_name>, ...) | DISTRIBUTED ON (<col_name>, ...) SORTED ON (<col_name>, ...)]
  [
    [ROW FORMAT <row_format>]
    [STORED AS <file_format>] | STORED BY '<storage.handler.class.name>' [WITH SERDEPROPERTIES (...)]
  ]
  [LOCATION <hdfs_path>]
  [TBLPROPERTIES (<property_name>=<property_value>, ...)]
AS
<query>;
```

Where:

- `<query>` — the query to run for results and populate the MV with data.
- `DISABLE REWRITE` — disables automatic query rewriting for the given MV.
- `PARTITIONED ON` — allows partitioning the MV by columns to improve the performance.
- `CLUSTERED ON` — used to divide the MV into roughly equal buckets for better maintainability and speed. Each bucket is stored as a separate file in HDFS.
- `ROW FORMAT` — defines row formatting using either delimiters or a custom SerDe class.
- `STORED AS` — sets the storage format. The default format is ORC.
- `STORED BY` — specifies the name of a custom storage handler class to store MV data using a non-native format.
- `LOCATION` — defines an HDFS path to store the MV data.
- `TBLPROPERTIES` — a list of key/value properties for tuning the MV behavior.

Query rewriting

Once an MV is created, the Hive optimizer automatically rewrites incoming queries to use the MV if it believes that this will improve the query execution plan. The query rewriting mechanism is enabled by default and can be disabled at MV creation time (`DISABLE REWRITE` clause) or by using the following command:

```
ALTER MATERIALIZED VIEW <mv_name> DISABLE REWRITE;
```

To disable the query rewriting feature globally, set the following property:

```
SET hive.materializedview.rewriting=false;
```

If an MV becomes stale (MV content is not updated for a long time), Hive will stop using this MV for query rewriting. The time interval to consider an MV as stale can be set with the `hive.materializedview.rewriting.time.window"="Nmin"` property in the ***Custom hive-site.xml*** ADCM settings section.

IMPORTANT

Automatic rewriting cannot be enabled if the MV was created from a non-transactional table. The source tables must be ACID, managed tables.

Rebuild materialized view

After an MV is created, any modifications to the source tables are not automatically reflected in the MV. The MV rebuild process should be triggered manually using the following command:

```
ALTER MATERIALIZED VIEW <mv_name> REBUILD;
```

If new data was added to source tables using `INSERT`, Hive runs an incremental rebuild. For `UPDATE` and `DELETE` operations, the full rebuild is performed.

Usage example

The steps below show basic MV operations:

1. Create a new MV from the test tables.

```
CREATE MATERIALIZED VIEW mv_emp_dp
  AS SELECT e.name, e.last_name, e.email, d.dep_name
  FROM employees e JOIN departments d
  ON (e.dept_id = d.id);
```

2. Get the list of available MVs:

```
SHOW MATERIALIZED VIEWS;
```

The output:

```
+------------+
|  tab_name  |
+------------+
| mv_emp_dp  |
+------------+
```

Also, check the HDFS warehouse location for new files.

```
$ sudo -u hdfs hdfs dfs -ls /apps/hive/warehouse/mv_demo.db/mv_emp_dp
```

A new ORC file with MV data is available in the Hive warehouse directory:

```
-rw-r--r--   3 hive hadoop        795 2023-11-20 13:23 /apps/hive/warehouse/mv_demo.db/mv_emp_dp/000000_0
```

3. Run a query similar to the one used for MV creation. Such a query gets the results from the precomputed MV instead of joining and scanning the two source tables.

```
SELECT e.last_name, e.email, d.dep_name
FROM employees e JOIN departments d
ON (e.dept_id = d.id)
WHERE d.dep_name='sales';
```

The result:

```
+--------------+-------------------------+-------------+
| e.last_name  |         e.email         | d.dep_name  |
+--------------+-------------------------+-------------+
| Ivanov       | ivan_ivanov123@mail.ru  | sales       |
+--------------+-------------------------+-------------++
```

4. Run the same query with `EXPLAIN EXTENDED` to ensure that query rewriting took place.

```
EXPLAIN EXTENDED SELECT e.last_name, e.email, d.dep_name
FROM employees e JOIN departments d
ON (e.dept_id = d.id)
WHERE d.dep_name='sales';
```

The output is shown below. Notice the alias `alias: mv_demo.mv_emp_dp` — this indicates that the results have been fetched from the `mv_demo.mv_emp_dp` MV with one table scan.

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| STAGE DEPENDENCIES:                                |
|   Stage-0 is a root stage                          |
|                                                    |
| STAGE PLANS:                                       |
|   Stage: Stage-0                                   |
|     Fetch Operator                                 |
|       limit: -1                                    |
|       Processor Tree:                              |
|         TableScan                                  |
|           alias: mv_demo.mv_emp_dp                 |
|           GatherStats: false                       |
|           Filter Operator                          |
|             isSamplingPred: false                  |
|             predicate: (dep_name = 'sales') (ty... |
|             Select Operator                        |
|               expressions: last_name (type: str... |
|               outputColumnNames: _col0, _col1, ... |
|               ListSink                             |
|                                                    |
+----------------------------------------------------+
```

5. Update the source table.

```
INSERT INTO employees VALUES
(5,'Anna','Petrova','ann_pett@yandxed.ru', 32, 1700, 1);
```

6. Run the `EXPLAIN EXTENDED` query again:

```
EXPLAIN EXTENDED SELECT e.last_name, e.email, d.dep_name
FROM employees e JOIN departments d
ON (e.dept_id = d.id)
WHERE d.dep_name='sales';
```

This time the execution plan includes joining and full scanning of the source tables since their contents is no more in sync with the MV.

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| STAGE DEPENDENCIES:                                |
|   Stage-1 is a root stage                          |
|   Stage-0 depends on stages: Stage-1               |
|                                                    |
| STAGE PLANS:                                       |
|   Stage: Stage-1                                   |
|     Tez                                            |
|       DagId: hive_20231120144209_2c92e022-a865-... |
|       Edges:                                       |
|         Map 1 <- Map 2 (BROADCAST_EDGE)            |
|       DagName: hive_20231120144209_2c92e022-a86... |
|       Vertices:                                    |
|         Map 1                                      |
|             Map Operator Tree:                     |
|                 TableScan                          |
|                   alias: e                         |
|                   Statistics: Num rows: 50 Dat ... |
|                   ...                              |
+----------------------------------------------------+
```

7. To refresh the `mv_emp_dp` MV and synchronize the updates made to the source tables, rebuild the MV:

```
ALTER MATERIALIZED VIEW mv_emp_dp REBUILD;
```

8. Run the `EXPLAIN EXTENDED` query again:

```
EXPLAIN EXTENDED SELECT e.last_name, e.email, d.dep_name
FROM employees e JOIN departments d
ON (e.dept_id = d.id)
WHERE d.dep_name='sales';
```

Now the execution plan indicates the use of MV for getting the data.

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| STAGE DEPENDENCIES:                                |
|   Stage-0 is a root stage                          |
|                                                    |
| STAGE PLANS:                                       |
|   Stage: Stage-0                                   |
|     Fetch Operator                                 |
|       limit: -1                                    |
|       Processor Tree:                              |
|         TableScan                                  |
|           alias: mv_demo.mv_emp_dp                 |
|           GatherStats: false                       |
|           Filter Operator                          |
|             isSamplingPred: false                  |
|             predicate: (dep_name = 'sales') (ty... |
|             Select Operator                        |
|               expressions: last_name (type: str... |
|               outputColumnNames: _col0, _col1, ... |
|               ListSink                             |
|                                                    |
+----------------------------------------------------+
```

9. To delete the materialized view, use the command:

```
DROP MATERIALIZED VIEW mv_emp_dp;
```

Hive on Spark

Konstantin Alpashkin

Contents

To Table of Contents

Configuration

Hive on Spark provides Hive with the ability to utilize Apache Spark as its execution engine. By default, Hive is configured to work via Tez.

Configuration

The easiest way to turn on the Spark execution engine is to use ADCM.

Use the ***Hive Configuration*** page to change the `hive.execution.engine` parameter value. Also, you can change this parameter in the *hive-site.xml* file located in the default Hive configuration directory */etc/hive/conf/*. The parameter `hive.execution.engine` is shown below.

```
<property>
    <name>hive.execution.engine</name>
    <value>spark</value>
</property>
```

See the Spark on Hive page to read more about Spark configuration.

There is a Hive configuration file in the home Spark directory (*/etc/spark3/conf/hive-site.xml*). You can change or add any parameter to this file if you want your Spark application to work with Hive in any custom way.

Default configuration file

```
<?xml version="1.0"?>
<configuration>
        <property>
                <name>hive.cbo.enable</name>
                <value>True</value>
        </property>
        <property>
                <name>hive.compute.query.using.stats</name>
                <value>False</value>
        </property>
        <property>
                <name>hive.execution.engine</name>
                <value>spark</value>
        </property>
        <property>
                <name>hive.log.explain.output</name>
                <value>True</value>
        </property>
        <property>
                <name>hive.metastore.event.db.notification.api.auth</name>
                <value>False</value>
        </property>
        <property>
                <name>hive.metastore.kerberos.keytab.file</name>
                <value>/etc/security/keytabs/hive.service.keytab</value>
        </property>
        <property>
                <name>hive.metastore.kerberos.principal</name>
                <value>hive/_HOST@EXAMPLE.COM</value>
        </property>
        <property>
                <name>hive.metastore.sasl.enabled</name>
                <value>False</value>
        </property>
        <property>
                <name>hive.metastore.uris</name>
                <value>thrift://aai-adh.ru-central1.internal:9083</value>
        </property>
        <property>
                <name>hive.metastore.warehouse.dir</name>
                <value>/apps/hive/warehouse</value>
        </property>
        <property>
                <name>hive.server2.authentication</name>
                <value>NONE</value>
        </property>
        <property>
                <name>hive.server2.authentication.kerberos.keytab</name>
                <value>/etc/security/keytabs/hive.service.keytab</value>
        </property>
        <property>
                <name>hive.server2.authentication.kerberos.principal</name>
                <value>hive/_HOST@EXAMPLE.COM</value>
        </property>
        <property>
                <name>hive.server2.authentication.spnego.keytab</name>
                <value>/etc/security/keytabs/HTTP.service.keytab</value>
        </property>
        <property>
                <name>hive.server2.authentication.spnego.principal</name>
                <value>HTTP/_HOST@EXAMPLE.COM</value>
        </property>
        <property>
                <name>hive.server2.enable.doAs</name>
                <value>False</value>
        </property>
        <property>
                <name>hive.stats.fetch.column.stats</name>
                <value>False</value>
        </property>
        <property>
                <name>hive.support.concurrency</name>
                <value>False</value>
        </property>
        <property>
                <name>hive.tez.container.size</name>
                <value>1024</value>
        </property>
        <property>
                <name>hive.txn.manager</name>
                <value>org.apache.hadoop.hive.ql.lockmgr.DummyTxnManager</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionDriverName</name>
                <value>com.mysql.jdbc.Driver</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionPassword</name>
                <value>admin</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionURL</name>
                <value>jdbc:mysql://aai-adh3.ru-central1.internal:3306/hive</value>
        </property>
        <property>
                <name>javax.jdo.option.ConnectionUserName</name>
                <value>APP</value>
        </property>
</configuration>
```

Hive with HBase

Eugenia Kuzina

Contents

To Table of Contents

Create a new HBase table in Hive

Use an existing HBase table in Hive

Hive/HBase integration allows Hive QL statements to access HBase tables for both read (`SELECT`) and write (`INSERT`) operations. You can create a new HBase table in Hive or access an existing table.

Create a new HBase table in Hive

To create a new HBase table that can be managed by Hive, use the `STORED BY` clause in the `CREATE TABLE` statement. It allows you to specify a storage handler required for integration.

Example:

```
$ sudo -u hive hive
```

```
CREATE TABLE hbase_table_test(key int, value1 string, value2 int, value3 int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,col_family1:val1,col_family1:val2,col_family2:val3")
TBLPROPERTIES ("hbase.table.name" = "hive_test", "hbase.mapred.output.outputtable" = "hive_test");
```

Where:

- `STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'` — specifies a storage handler that allows Hive to access data stored in HBase.
- `WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,col_family1:val1,col_family1:val2,col_family2:val3")` — controls the mapping of HBase columns to Hive. The `hbase.columns.mapping` can contain the following entries: `:key`, `:timestamp`, or `<column-family-name>:[<column-name>][#(binary|string)]`. The type specification is delimited by `#`. If `hbase.columns.mapping` does not include the type specification, the value from `hbase.table.default.storage.type` is used.
- `TBLPROPERTIES ("hbase.table.name" = "hive_test", "hbase.mapred.output.outputtable" = "hive_test")` — specifies table properties. The `hbase.table.name` property defines the table name in HBase. In this example, the table is known as `hbase_table_test` within Hive, and as `hive_test` within HBase. If `hbase.table.name` is not set, the table has the same name in Hive and HBase. The `hbase.mapred.output.outputtable` property is required if you plan to insert data into the table (it is used by the hbase.mapreduce.TableOutputFormat class).

Execute the `list` command to check if HBase has information about the new table:

```
$ hbase shell

$ list
```

The `list` output should contain the `hive_test` table.

In HBase shell, utilize the `put` command to add data to this table:

```
put 'hive_test', '1' , 'col_family1:val1', 'test'
put 'hive_test', '1' , 'col_family1:val2', '1'
put 'hive_test', '1' , 'col_family2:val3', '1'

put 'hive_test', '2' , 'col_family1:val1', 'test2'
put 'hive_test', '2' , 'col_family1:val2', '2'
put 'hive_test', '2' , 'col_family2:val3', '2'
```

Use the `scan` command to check the table data:

```
scan 'hive_test'
```

The result:

```
ROW                                                          COLUMN+CELL
1                                                           column=col_family1:val1, timestamp=1694606917613, value=test
1                                                           column=col_family1:val2, timestamp=1694606917647, value=1
1                                                           column=col_family2:val3, timestamp=1694606922996, value=1
2                                                           column=col_family1:val1, timestamp=1694607054746, value=test2
2                                                           column=col_family1:val2, timestamp=1694607054763, value=2
2                                                           column=col_family2:val3, timestamp=1694607056127, value=2
2 row(s)
Took 0.0175 seconds
```

The data is added successfully to the table created in Hive.

Insert a new row to the same table through the Hive shell:

```
INSERT INTO hbase_table_test VALUES(3,'test3', 3, 3);
```

Check the table data in the HBase shell:

```
scan 'hive_test'
```

The result:

```
ROW                    COLUMN+CELL
 1                     column=col_family1:val1, timestamp=1698334191476, value=test
 1                     column=col_family1:val2, timestamp=1698334191504, value=1
 1                     column=col_family2:val3, timestamp=1698334191538, value=1
 2                     column=col_family1:val1, timestamp=1698334191591, value=test2
 2                     column=col_family1:val2, timestamp=1698334191616, value=2
 2                     column=col_family2:val3, timestamp=1698334193906, value=2
 3                     column=col_family1:val1, timestamp=1698334255870, value=test3
 3                     column=col_family1:val2, timestamp=1698334255870, value=3
 3                     column=col_family2:val3, timestamp=1698334255870, value=3
3 row(s)
Took 0.0133 seconds
```

This example demonstrates that Hive and HBase can work with the same table.

Use an existing HBase table in Hive

You can also operate with an existing HBase table in Hive. The example below uses the `people_ages` table from the Bulk loading via built-in MapReduce jobs article.

For an existing HBase table, create an external table in the Hive shell and specify other properties according to the table structure:

```
CREATE EXTERNAL TABLE hbase_people_ages  (name string, age int)
STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,basic:age")
TBLPROPERTIES ("hbase.table.name" = "people_ages", "hbase.mapred.output.outputtable" = "people_ages");
```

Select data to check whether the table is available:

```
SELECT * FROM hbase_people_ages LIMIT 5;
```

The result:

```
OK
Abbott Delia    62
Abbott Howard   24
Abbott Jack     29
Adams Clyde     29
Aguilar Myrtie  23
Time taken: 0.161 seconds, Fetched: 5 row(s)
```

It is also possible to combine access to HBase tables with native Hive tables via `JOIN` and `UNION` clauses.

Create another table in Hive:

```
CREATE TABLE people (key int, name string, points int);

INSERT INTO people
    VALUES (1,'Phelps Rena', 35), (2,'Quinn Brian', 48), (1,'Reyes Danny', 52);
```

Execute the query with a `JOIN` clause:

```
SELECT hbase_people_ages.*, people.points FROM hbase_people_ages JOIN people ON (hbase_people_ages.name = people.name);
```

The result:

```
OK
Phelps Rena     18      35
Quinn Brian     20      48
Reyes Danny     41      52
Time taken: 10.003 seconds, Fetched: 3 row(s)
```

View a Hive job via TEZ

Konstantin Alpashkin

Contents

To Table of Contents

Job statistics
- DAG Details
- DAG Counters
- Graphical View
- All Vertices
- All Tasks
- All Tasks Attempts
- Vertex Swimlane

Each job that you run in Hive appears on the Tez UI. Complete the following steps on the Tez UI:

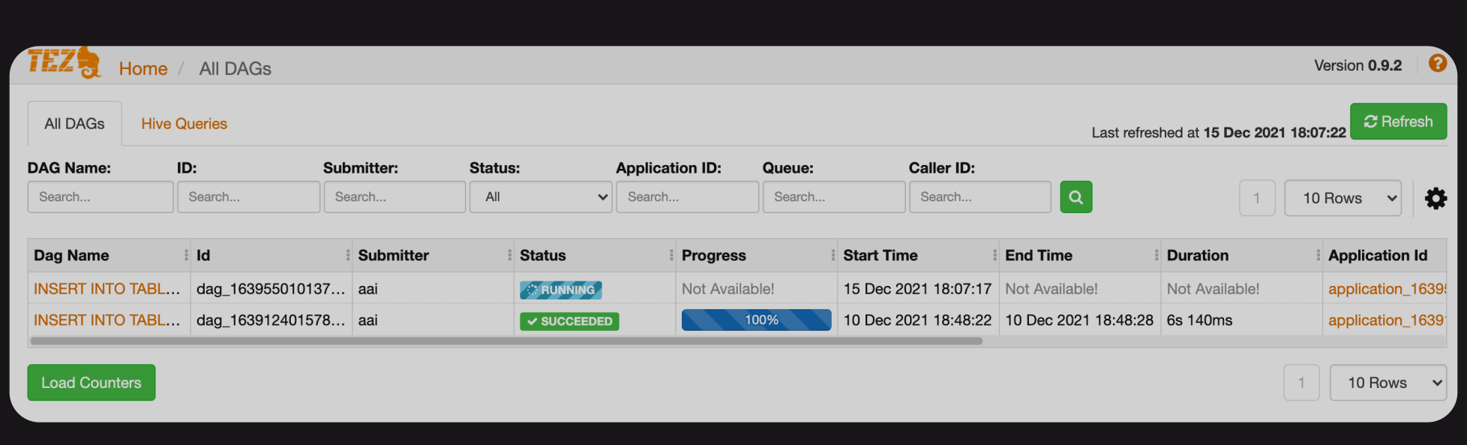
1. Open the Tez UI. You can find the link to the interface on the TEZ overview page.
2. Launch the Hive shell on the host with the Hive HiveServer2 component. More details on using the Hive shell are available on the Work with Hive tables page.
3. Create a table using the Hive shell:

```
CREATE TABLE customers
( customer_id int NOT NULL,
  customer_name char(50) NOT NULL,
  address char(50),
  city char(50),
  state char(25),
  zip_code char(10)
);
```

4. Insert data into the created table:

```
INSERT INTO TABLE customers VALUES
(1,'John Smith','Holliwood blvd','Los Angeles','CA','435267'),
(2,'Alice Brown','42 st','New York','NY','876234'),
(3,'Maria Jensen','Paris drv','Chicago','IL','293752');
```

5. Use Tez UI during the data insertion to monitor this process.


TEZ
Home / All DAGs
Version 0.9.2
All DAGs
Hive Queries
Last refreshed at 15 Dec 2021 18:07:22
Refresh
DAG Name:
ID:
Submitter:
Status:
Application ID:
Queue:
Caller ID:
Search...
All
10 Rows
Load Counters

Dag Name	Id	Submitter	Status	Progress	Start Time	End Time	Duration	Application Id
INSERT INTO TABL...	dag_163955010137...	aai	RUNNING	Not Available!	15 Dec 2021 18:07:17	Not Available!	Not Available!	application_1639
INSERT INTO TABL...	dag_163912401578...	aai	SUCCEEDED	100%	10 Dec 2021 18:48:22	10 Dec 2021 18:48:28	6s 140ms	application_1639

Adding data to Tez

You can further check the inserted data with the following query:

```
SELECT *
FROM customers
WHERE state = 'CA'
ORDER BY customer_id ASC;
```

The output looks like the one below.

```
Query ID = aai_20211215195428_e60e1dc5-f6f3-4c5e-896a-6e56407845c1
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1639550101374_0007)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 5,38 s
----------------------------------------------------------------------------------------------
OK
1       John Smith                                        Holliwood blvd                                        Los
Angeles                                         CA                      435267
4       Peter Stormik                                     International blv                                     San
Francisco                                       CA                      231987
5       Daniel Morrowetz                                  Bond rd                                               Elk Grove
CA                              390817
Time taken: 6.447 seconds, Fetched: 3 row(s)
```

The `SELECT` job appears in the Tez UI similarly to `INSERT`.

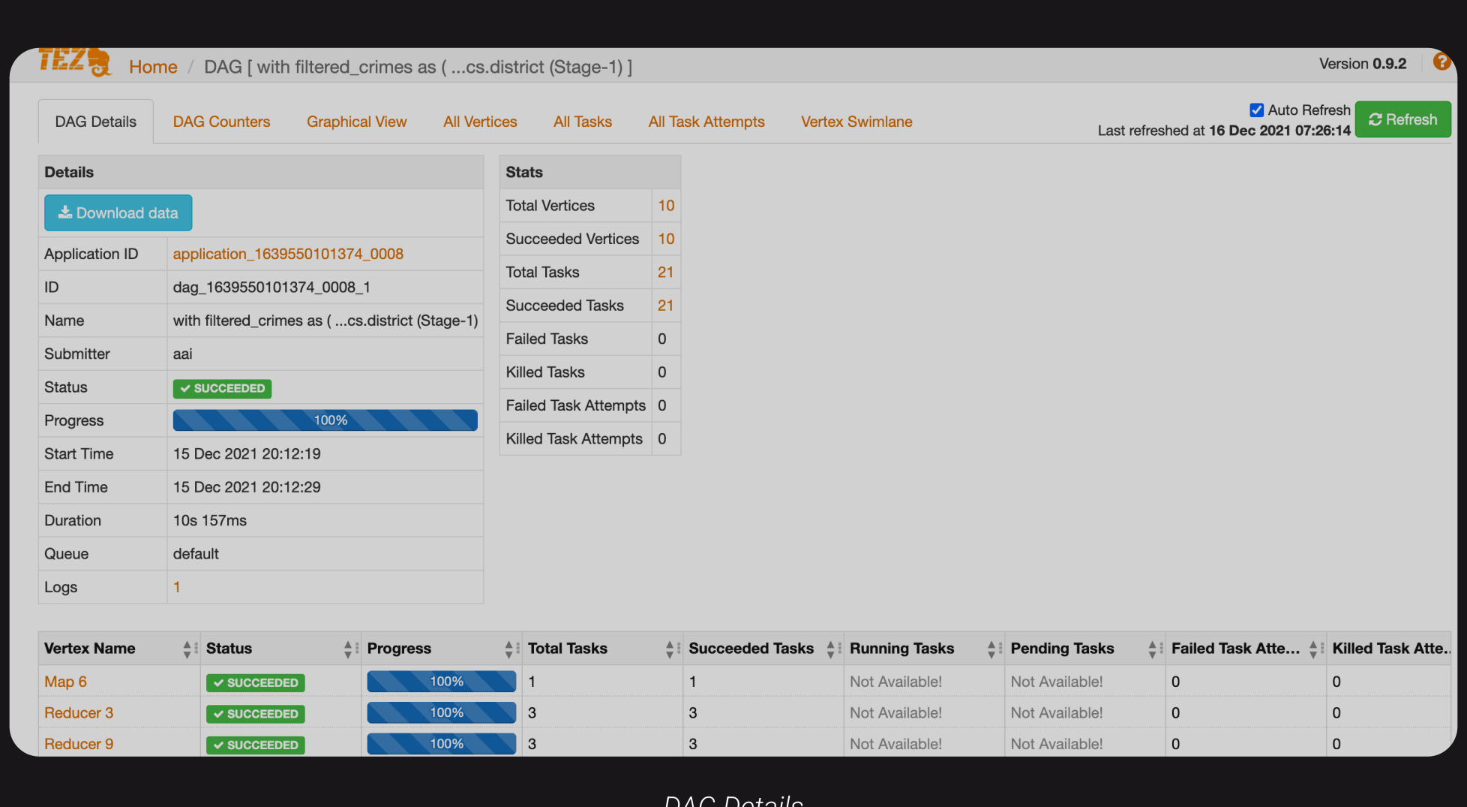
6. Run the Hive computation example to get the data about this computation on Tez UI. On the job completion, you can see the full statistics on the jobs page.

Job statistics

DAG Details

You can see all detailed information about the generated DAG. The ***Query Details*** tab provides information such as:

- Hive query ID;
- user who executed the query;
- start and end time;
- total time needed to execute the query;
- tables that were read and written;
- application ID;
- DAG IDs;
- session ID;
- thread ID;
- queue against which the query was run.


TEZ
Home / DAG [with filtered_crimes as (...cs.district (Stage-1)]
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 07:26:14
Refresh
Download data

Details	
Application ID	application_1639550101374_0008
ID	dag_1639550101374_0008_1
Name	with filtered_crimes as (...cs.district (Stage-1)
Submitter	aai
Status	SUCCEEDED
Progress	100%
Start Time	15 Dec 2021 20:12:19
End Time	15 Dec 2021 20:12:29
Duration	10s 157ms
Queue	default
Logs	1

Stats	
Total Vertices	10
Succeeded Vertices	10
Total Tasks	21
Succeeded Tasks	21
Failed Tasks	0
Killed Tasks	0
Failed Task Attempts	0
Killed Task Attempts	0

Vertex Name	Status	Progress	Total Tasks	Succeeded Tasks	Running Tasks	Pending Tasks	Failed Task Atte...	Killed Task Atte...
Map 6	SUCCEEDED	100%	1	1	Not Available!	Not Available!	0	0
Reducer 3	SUCCEEDED	100%	3	3	Not Available!	Not Available!	0	0
Reducer 9	SUCCEEDED	100%	3	3	Not Available!	Not Available!	0	0

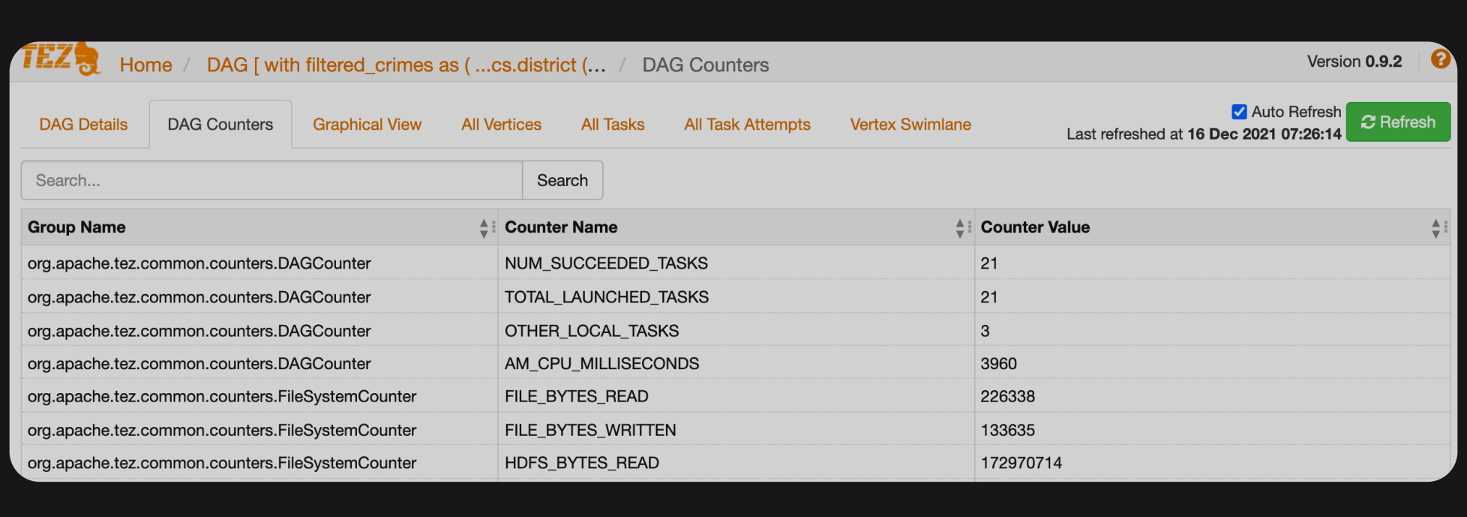
DAG Details

DAG Counters

DAG counters provide a way to measure the progress or the number of operations that occur within a generated DAG. Counters are used to gather statistics for quality control purposes or for problem diagnosis.

DAG counters provide the following details:

- Group name;
- Counter name;
- Counter value.

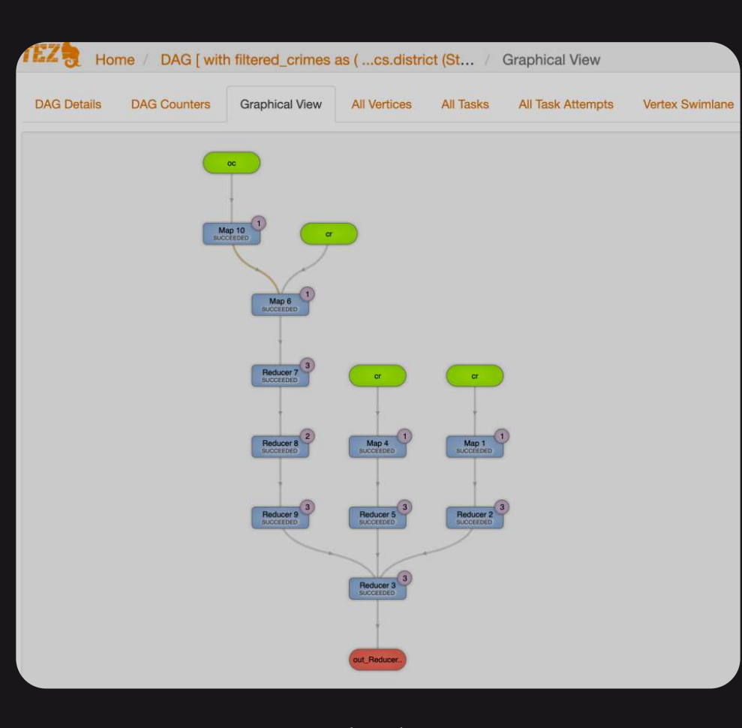

TEZ
Home / DAG [with filtered_crimes as (...cs.district (... / DAG Counters
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 07:26:14
Refresh
Search...
Search

Group Name	Counter Name	Counter Value
org.apache.tez.common.counters.DAGCounter	NUM_SUCCEEDED_TASKS	21
org.apache.tez.common.counters.DAGCounter	TOTAL_LAUNCHED_TASKS	21
org.apache.tez.common.counters.DAGCounter	OTHER_LOCAL_TASKS	3
org.apache.tez.common.counters.DAGCounter	AM_CPU_MILLISECONDS	3960
org.apache.tez.common.counters.FileSystemCounter	FILE_BYTES_READ	226338
org.apache.tez.common.counters.FileSystemCounter	FILE_BYTES_WRITTEN	133635
org.apache.tez.common.counters.FileSystemCounter	HDFS_BYTES_READ	172970714

DAG Counters

Graphical View

The ***DAG Graphical View*** tab displays the DAG that is generated by Hive in the form of a flowchart.

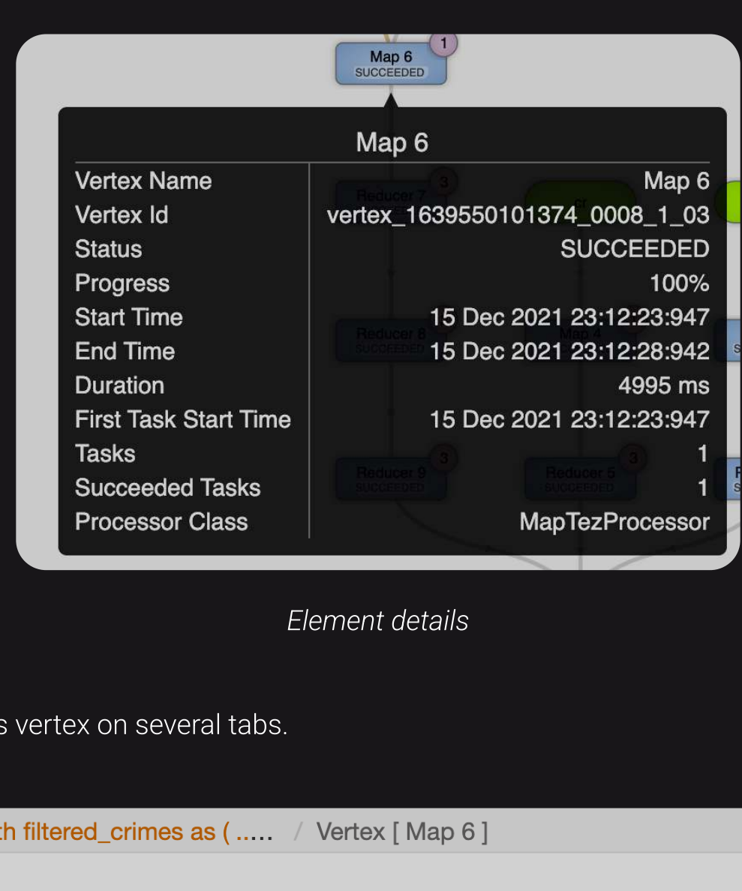

TEZ
Home / DAG [with filtered_crimes as (...cs.district (St... / Graphical View
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Map 10
Map 6
Reducer 7
Reducer 8
Reducer 9
Map 4
Reducer 5
Map 1
Reducer 2
Reducer 3
out_Reducer

Graphical View

Here, the inputs to vertices `Map 10`, `Map 6`, `Map 4`, and `Map 1` are the tables displayed in green boxes. Next, `Map 6` depends on the result set generated by `Map 10`. `Map 6` then passes its result as an input to `Reducer 7`. When the `Reducer 7` finishes its execution, the results are passed onto `Reducer 8` and then to `Reducer 9`. `Reducer 3` is the last vertex in the DAG flow. After `Reducer 3`_ successfully completes its execution, the query output is written to a file in HDFS.

There are a few options to change the layout of the DAG flow. You can hide the input and the output nodes to view the task vertices by clicking the ***Toggle source/sink visibility*** icon. You can switch between the horizontal and vertical orientation by clicking the ***Toggle orientation*** icon.

When you hover an element, you get the full information about vertex details, status and progress, duration, and tasks number.

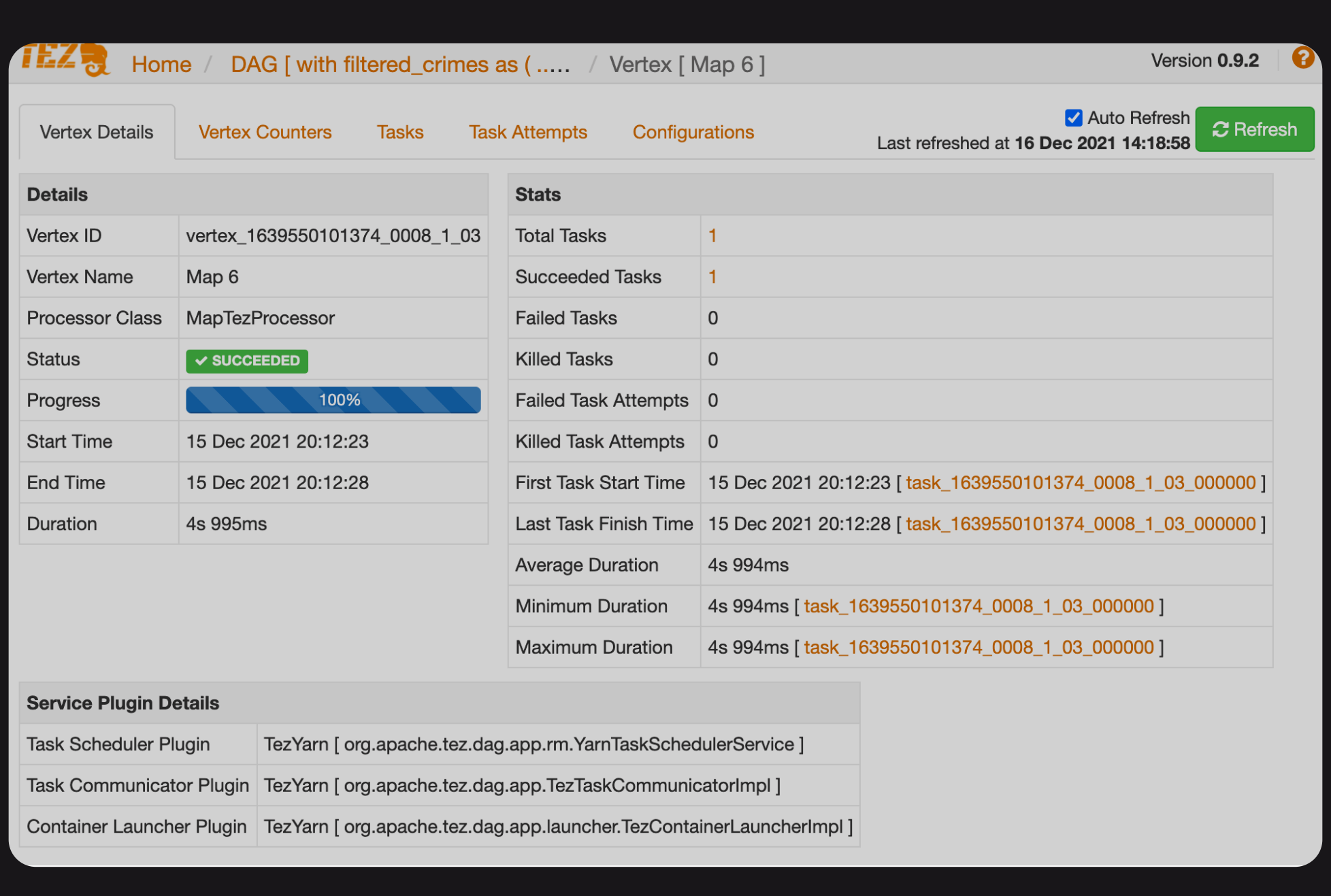

Map 6
SUCCEEDED
1

Map 6

Vertex Name	Map 6
Vertex Id	vertex_1639550101374_0008_1_03
Status	SUCCEEDED
Progress	100%
Start Time	15 Dec 2021 23:12:23:947
End Time	15 Dec 2021 23:12:28:942
Duration	4995 ms
First Task Start Time	15 Dec 2021 23:12:23:947
Tasks	1
Succeeded Tasks	1
Processor Class	MapTezProcessor

Element details

If you click on an element, you get the full statistics about this vertex on several tabs.


TEZ
Home / DAG [with filtered_crimes as (..... / Vertex [Map 6]
Version 0.9.2
Vertex Details
Vertex Counters
Tasks
Task Attempts
Configurations
Auto Refresh
Last refreshed at 16 Dec 2021 14:18:58
Refresh

Details	
Vertex ID	vertex_1639550101374_0008_1_03
Vertex Name	Map 6
Processor Class	MapTezProcessor
Status	SUCCEEDED
Progress	100%
Start Time	15 Dec 2021 20:12:23
End Time	15 Dec 2021 20:12:28
Duration	4s 995ms

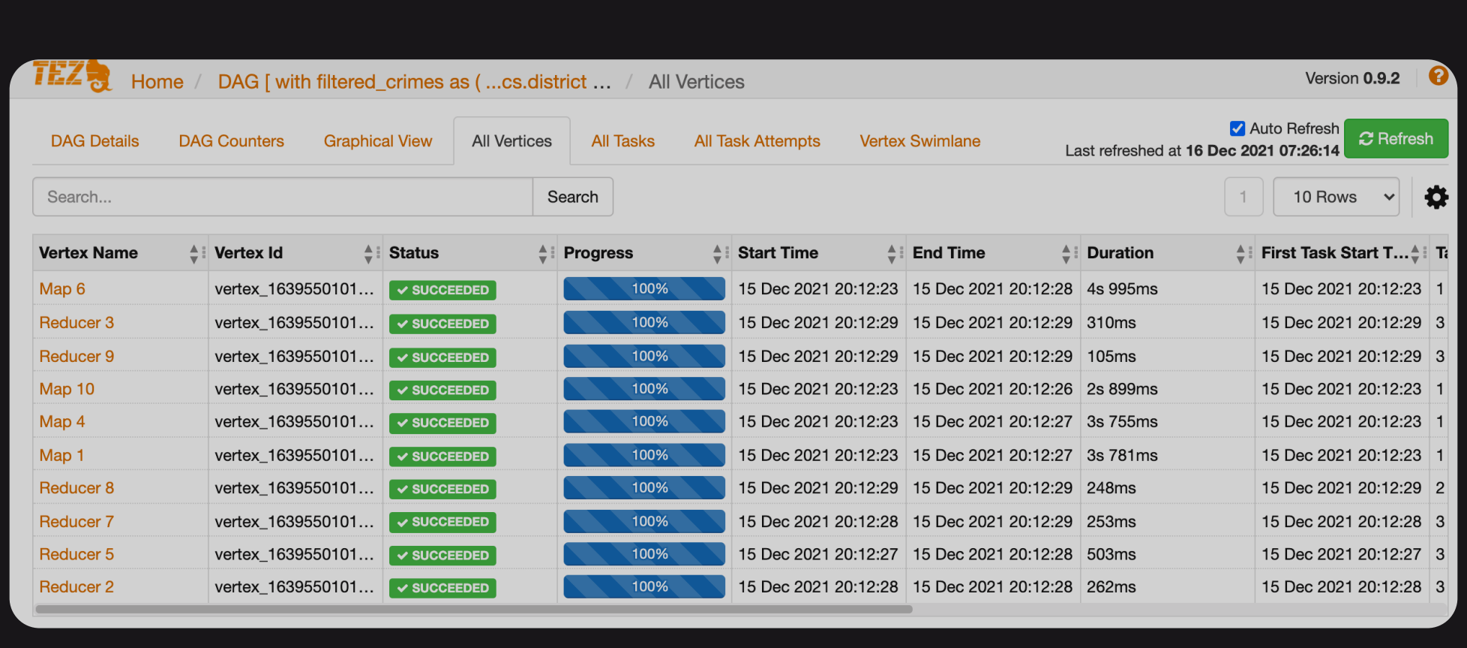
Stats	
Total Tasks	1
Succeeded Tasks	1
Failed Tasks	0
Killed Tasks	0
Failed Task Attempts	0
Killed Task Attempts	0
First Task Start Time	15 Dec 2021 20:12:23 [task_1639550101374_0008_1_03_000000]
Last Task Finish Time	15 Dec 2021 20:12:28 [task_1639550101374_0008_1_03_000000]
Average Duration	4s 994ms
Minimum Duration	4s 994ms [task_1639550101374_0008_1_03_000000]
Maximum Duration	4s 994ms [task_1639550101374_0008_1_03_000000]

Service Plugin Details	
Task Scheduler Plugin	TezYarn [org.apache.tez.dag.app.rm.YarnTaskSchedulerService]
Task Communicator Plugin	TezYarn [org.apache.tez.dag.app.TezTaskCommunicatorImpl]
Container Launcher Plugin	TezYarn [org.apache.tez.dag.app.launcher.TezContainerLauncherImpl]

Vertex Details

All Vertices

On this tab, you can see a table that shows full information about Maps and Reducers. Each Vertex name is a link that directs you to the detailed information on Maps and Reducers.

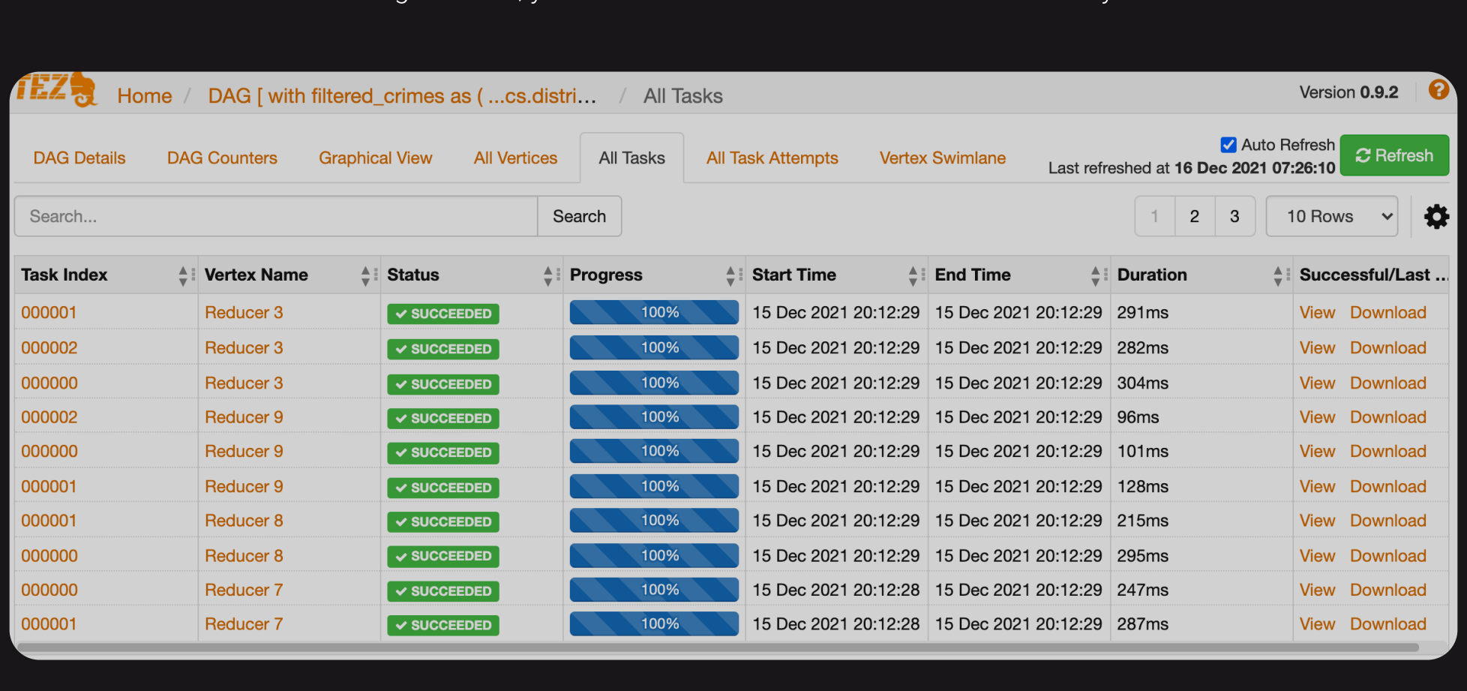

TEZ
Home / DAG [with filtered_crimes as (...cs.district ... / All Vertices
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 07:26:14
Refresh
Search...
Search
1
10 Rows

Vertex Name	Vertex Id	Status	Progress	Start Time	End Time	Duration	First Task Start T...	Ta
Map 6	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:23	15 Dec 2021 20:12:28	4s 995ms	15 Dec 2021 20:12:23	1
Reducer 3	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	310ms	15 Dec 2021 20:12:29	3
Reducer 9	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	105ms	15 Dec 2021 20:12:29	3
Map 10	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:23	15 Dec 2021 20:12:26	2s 899ms	15 Dec 2021 20:12:23	1
Map 4	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:23	15 Dec 2021 20:12:27	3s 755ms	15 Dec 2021 20:12:23	1
Map 1	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:23	15 Dec 2021 20:12:27	3s 781ms	15 Dec 2021 20:12:23	1
Reducer 8	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	248ms	15 Dec 2021 20:12:29	2
Reducer 7	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:28	15 Dec 2021 20:12:29	253ms	15 Dec 2021 20:12:28	3
Reducer 5	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:27	15 Dec 2021 20:12:28	503ms	15 Dec 2021 20:12:27	3
Reducer 2	vertex_1639550101...	SUCCEEDED	100%	15 Dec 2021 20:12:28	15 Dec 2021 20:12:28	262ms	15 Dec 2021 20:12:28	3

All Vertices

All Tasks

This tab shows the detailed statistics about Reducer tasks. Using the links, you can view the detailed information about any task or Reducer.

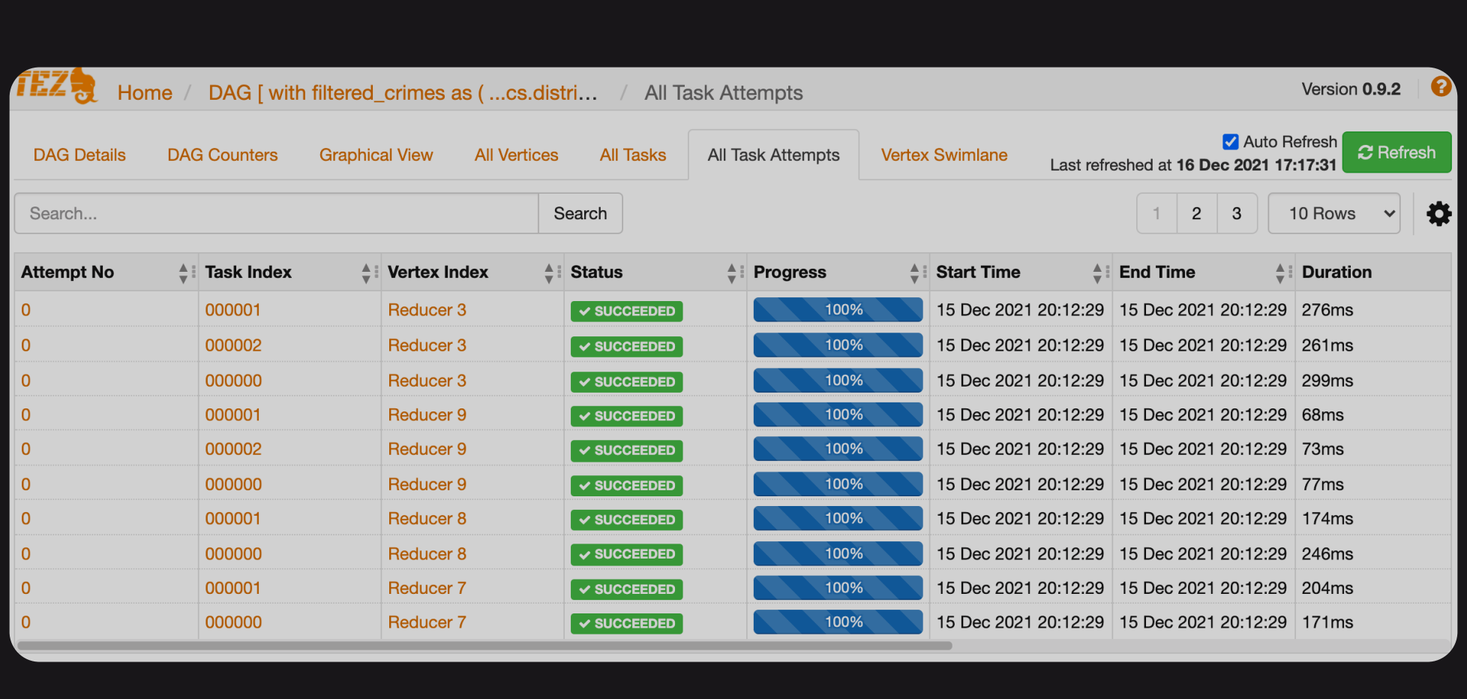

TEZ
Home / DAG [with filtered_crimes as (...cs.distri... / All Tasks
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 07:26:10
Refresh
Search...
Search
1
2
3
10 Rows

Task Index	Vertex Name	Status	Progress	Start Time	End Time	Duration	Successful/Last ...
000001	Reducer 3	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	291ms	View Download
000002	Reducer 3	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	282ms	View Download
000000	Reducer 3	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	304ms	View Download
000002	Reducer 9	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	96ms	View Download
000000	Reducer 9	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	101ms	View Download
000001	Reducer 9	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	128ms	View Download
000001	Reducer 8	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	215ms	View Download
000000	Reducer 8	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	295ms	View Download
000000	Reducer 7	SUCCEEDED	100%	15 Dec 2021 20:12:28	15 Dec 2021 20:12:29	247ms	View Download
000001	Reducer 7	SUCCEEDED	100%	15 Dec 2021 20:12:28	15 Dec 2021 20:12:29	287ms	View Download

All Tasks

All Tasks Attempts

This tab shows the detailed statistics about all Reducer tasks attempts. Using the links, you can view the full information about any attempt, task or reducer.

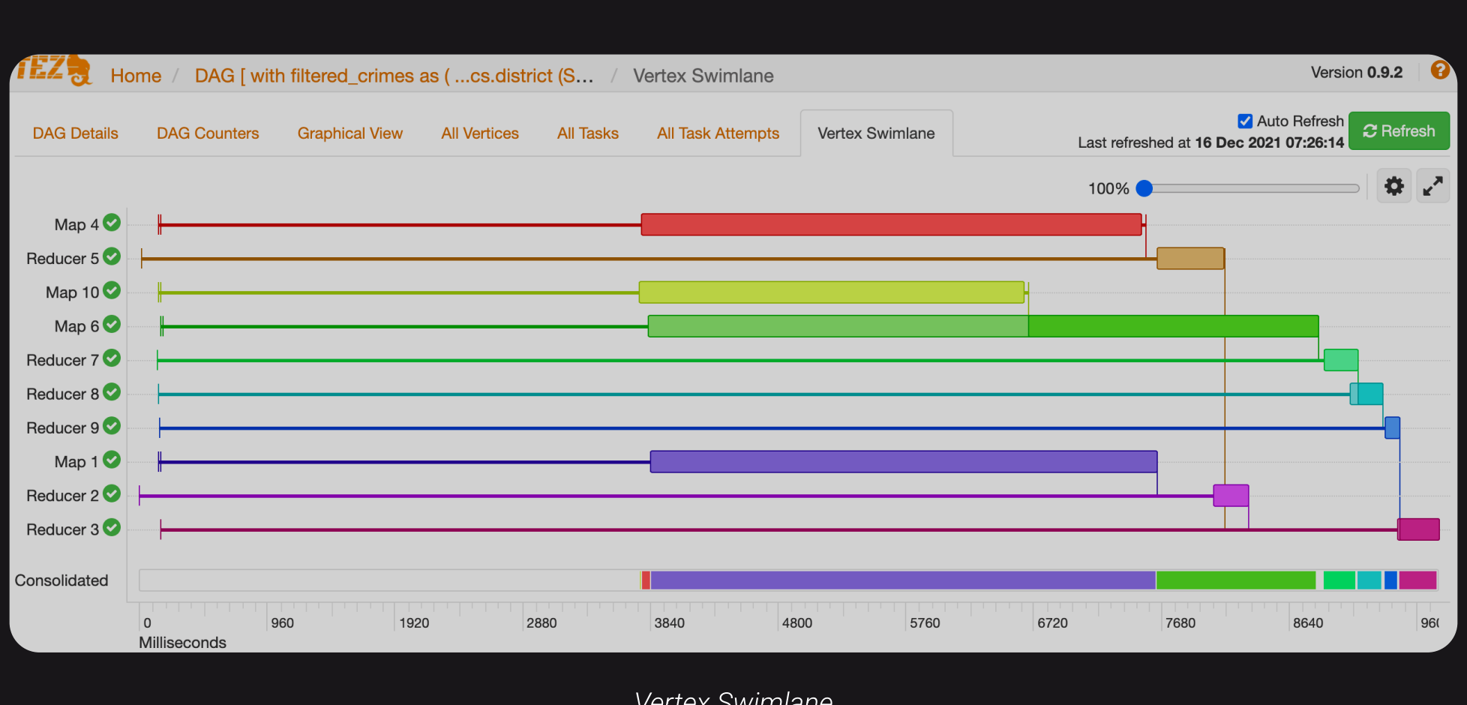

TEZ
Home / DAG [with filtered_crimes as (...cs.distri... / All Task Attempts
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 17:17:31
Refresh
Search...
Search
1
2
3
10 Rows

Attempt No	Task Index	Vertex Index	Status	Progress	Start Time	End Time	Duration
0	000001	Reducer 3	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	276ms
0	000002	Reducer 3	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	261ms
0	000000	Reducer 3	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	299ms
0	000001	Reducer 9	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	68ms
0	000002	Reducer 9	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	73ms
0	000000	Reducer 9	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	77ms
0	000001	Reducer 8	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	174ms
0	000000	Reducer 8	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	246ms
0	000001	Reducer 7	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	204ms
0	000000	Reducer 7	SUCCEEDED	100%	15 Dec 2021 20:12:29	15 Dec 2021 20:12:29	171ms

All Tasks Attempts

Vertex Swimlane

This tab shows the DAG of the vertices against time. Each mapping and reducing task is a vertex.


TEZ
Home / DAG [with filtered_crimes as (...cs.district (S... / Vertex Swimlane
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 07:26:14
Refresh
100%
Map 4
Reducer 5
Map 10
Map 6
Reducer 7
Reducer 8
Reducer 9
Map 1
Reducer 2
Reducer 3
Consolidated
0
960
1920
2880
3840
4800
5760
6720
7680
8640
96
Milliseconds

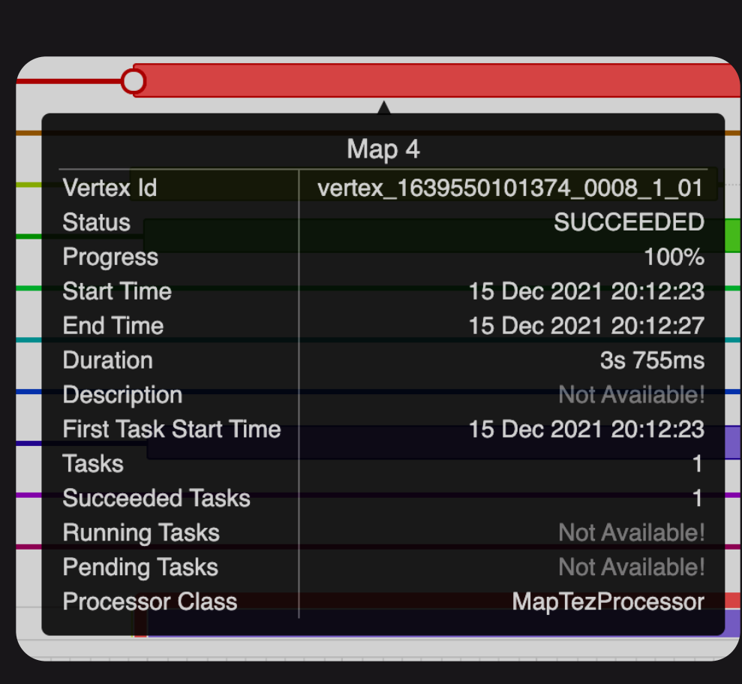
Vertex Swimlane

Each horizontal line of the swimlane represents the total time taken by the vertex to complete. The vertical lines indicate the time when:

- the vertex was initialized;
- the vertex was started;
- the time when the first task was started;
- the time when the last task was completed;
- the time when the vertex finished its execution.

When you hover the vertical line, the bubble displays the stage of the vertex execution and provides a timestamp.

To know more about a particular vertex, hover the mouse anywhere at the horizontal line.



Map 4

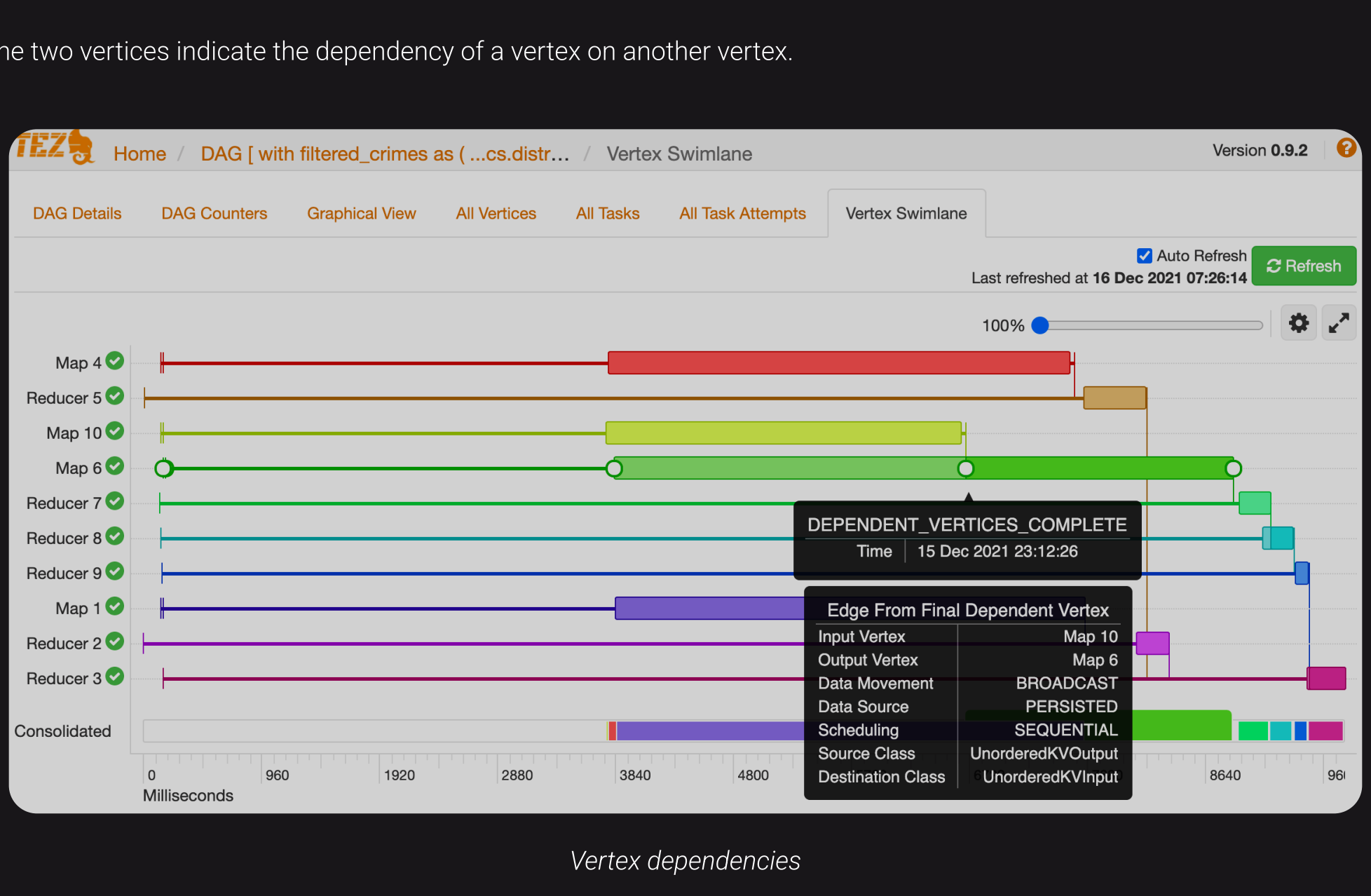
Vertex Id	vertex_1639550101374_0008_1_01
Status	SUCCEEDED
Progress	100%
Start Time	15 Dec 2021 20:12:23
End Time	15 Dec 2021 20:12:27
Duration	3s 755ms
Description	Not Available!
First Task Start Time	15 Dec 2021 20:12:23
Tasks	1
Succeeded Tasks	1
Running Tasks	Not Available!
Pending Tasks	Not Available!
Processor Class	MapTezProcessor

Vertex Information

The following details can help you to view the time taken by a particular task and debug the query:

- ***Vertex ID*** — a unique identifier for a particular vertex.
- ***Status*** — indicates the query status.
- ***Progress*** — the progress of the vertex, usually for long-running queries.
- ***Start time*** — indicates when a particular vertex started.
- ***End time*** — indicates when a particular vertex ended.
- ***Duration*** (in milliseconds) — the total time taken by the vertex to complete its execution.
- ***Description*** — the description of the vertex.
- ***First task start time*** — when the first task within that vertex started its execution.
- ***Tasks*** — the total number of the tasks executed in a particular vertex.
- ***Succeeded tasks*** — the number of the tasks that were executed successfully within that vertex.
- ***Running tasks*** — the tasks that are still running.
- ***Pending tasks*** — the tasks that have not yet started their execution.
- ***Processor class*** — the Hive processor for Tez that forms the vertices in Tez and processes the data.

The vertical lines that connect the two vertices indicate the dependency of a vertex on another vertex.


TEZ
Home / DAG [with filtered_crimes as (...cs.distr... / Vertex Swimlane
Version 0.9.2
DAG Details
DAG Counters
Graphical View
All Vertices
All Tasks
All Task Attempts
Vertex Swimlane
Auto Refresh
Last refreshed at 16 Dec 2021 07:26:14
Refresh
100%
Map 4
Reducer 5
Map 10
Map 6
Reducer 7
Reducer 8
Reducer 9
Map 1
Reducer 2
Reducer 3
Consolidated
0
960
1920
2880
3840
4800
8640
96
Milliseconds

DEPENDENT_VERTICES_COMPLETE

Time	15 Dec 2021 23:12:26

Edge From Final Dependent Vertex

Input Vertex	Map 10
Output Vertex	Map 6
Data Movement	BROADCAST
Data Source	PERSISTED
Scheduling	SEQUENTIAL
Source Class	UnorderedKVOutput
Destination Class	UnorderedKVInput

Vertex dependencies

In this example, `Map 6` depends on the results of `Map 10`. `Map 6` will finish its execution only when `Map 10` ends its execution successfully. Similarly, `Reducer 8` depends on `Reducer 7` to complete its execution.

Hive performance tuning

Konstantin Alpashkin

Contents

To Table of Contents

Partitioning

Bucketing

Vectorization

Cost-based optimization (CBO)

Use TEZ

Use ORC format

This article describes optimization techniques and best practices that can improve your Hive's performance.

Partitioning

In Hive parlance, **partitioning** is a way to split a single large table into smaller tables based on the column values. In a partitioned table, Hive creates a new partition for each distinct column value. For each partition, Hive creates a separate HDFS directory to store partition data instead of writing all the data to one HDFS file.

Partitioning can greatly speed up Hive operations by eliminating the need to scan unnecessary data that might be irrelevant for the current query. Partitioning is most effective when the data is evenly distributed across partitions. If the data is heavily skewed across several partitions, some worker nodes will have much more data to process than others, resulting in inefficient use of cluster resources. Also, the efficiency of partitioning depends on how the partition columns are used. For example, it is recommended to partition a table by those columns frequently used in the `GROUP BY` operations.

The syntax for creating a partitioned table is as follows:

```
CREATE TABLE transactions(
    txn_id int, acc_id int, txn_amount decimal(10,2)
 ) PARTITIONED BY (txn_date date);
```

Run the `DESCRIBE` command to view the table structure:

```
DESCRIBE transactions;
```

The command creates a table partitioned by the `txn_date` column as indicated in the output:

```
+--------------------------+----------------+----------+
|         col_name         |   data_type    | comment  |
+--------------------------+----------------+----------+
| txn_id                   | int            |          |
| acc_id                   | int            |          |
| txn_amount               | decimal(10,2)  |          |
| txn_date                 | date           |          |
|                          | NULL           | NULL     |
| # Partition Information  | NULL           | NULL     |
| # col_name               | data_type      | comment  |
| txn_date                 | date           |          |
+--------------------------+----------------+----------+
```

For every distinct value in the `txn_date` column, Hive will create a new partition to store rows related to that date. Check the HDFS files created by Hive to store the table data:

```
$ hdfs dfs -ls /apps/hive/warehouse/transactions
```

The output:

```
Found 3 items
drwxr-xr-x   - hive hadoop          0 2024-05-01 00:48 /apps/hive/warehouse/transactions/txn_date=2023-01-01
drwxr-xr-x   - hive hadoop          0 2024-05-01 00:49 /apps/hive/warehouse/transactions/txn_date=2023-01-02
drwxr-xr-x   - hive hadoop          0 2024-05-01 00:49 /apps/hive/warehouse/transactions/txn_date=2023-01-03
```

Hive creates a subdirectory for each partition and stores the related data there. Such partitioning is very beneficial for further reads since Hive will only need to scan specific partition directories instead of scanning the entire dataset.

> **NOTE**
>
> You should be careful when choosing a column for partitioning because too many partitions may load the NameNode.

You can also create multi-column partitions for more granular splitting of data. The example of creating a table with multiple partitions is shown below.

```
CREATE TABLE transactions_1 (
    id int, txn_amount decimal(10,2)
 ) PARTITIONED BY (txn_date date, acc_id int);
```

Check the table partitions using the command:

```
DESCRIBE transactions_1;
```

The output:

```
+--------------------------+----------------+----------+
|         col_name         |   data_type    | comment  |
+--------------------------+----------------+----------+
| id                       | int            |          |
| txn_amount               | decimal(10,2)  |          |
| txn_date                 | date           |          |
| acc_id                   | int            |          |
|                          | NULL           | NULL     |
| # Partition Information  | NULL           | NULL     |
| # col_name               | data_type      | comment  |
| txn_date                 | date           |          |
| acc_id                   | int            |          |
+--------------------------+----------------+----------+
```

When inserting data to this table, Hive stores the data in HDFS as shown below.

```
$ hdfs dfs -ls -R /apps/hive/warehouse/transactions_1
```

The output:

```
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-01
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-01/acc_id=1
-rw-r--r--   3 hive hadoop          8 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-01/acc_id=1/000000_0
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-01/acc_id=2
-rw-r--r--   3 hive hadoop          9 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-01/acc_id=2/000000_0
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-02
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-02/acc_id=3
-rw-r--r--   3 hive hadoop          8 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-02/acc_id=3/000000_0
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-03
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-03/acc_id=1
-rw-r--r--   3 hive hadoop          8 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-03/acc_id=1/000000_0
drwxr-xr-x   - hive hadoop          0 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-03/acc_id=2
-rw-r--r--   3 hive hadoop          8 2024-05-15 17:45 /apps/hive/warehouse/transactions_1/txn_date=2024-01-03/acc_id=2/000000_0
```

Notice that some of the *txn_date={date}* HDFS directories might contain the *acc_id=N* subdirectories that store data partitioned by the `acc_id` column values.

Bucketing

Similar to partitioning, Hive **bucketing** (also called clustering) is an optimization technique intended to split a large table into several manageable files which are faster to scan. The main distinction between partitioning and bucketing is that Hive creates new partitions based on table data — one partition per each distinct value in the partition column. Whereas with bucketing you can specify a predefined number of buckets at the time of Hive table creation.

The following table highlights major differences between partitions and buckets.

Partitioning	Bucketing
An HDFS directory is created for each partition	An HDFS file is created for each bucket
One or more columns can be used to split the table data into partitions	Only one column is used to split the table into buckets
The number of partitions depends on column values — one partition per one distinct value	You can manage the number of buckets to create by specifying the number explicitly
Created using the `PARTITIONED BY` clause	Created using the `CLUSTERED BY` clause

The use of Hive bucketing has the following impact on a Hive cluster:

- Reduced total data size.
- Reduced insert speed.
- Faster selection by the sort key.
- Joining two tables using sort merge join can be done without pre-sorting.
- Hive and Spark buckets are incompatible due to different hashing algorithms.

To create a bucketed table, use the following syntax:

```
CREATE TABLE employees(
    emp_id int,
    first_name string,
    last_name string,
    department_id int
)
CLUSTERED BY (department_id) SORTED BY (emp_id) INTO 5 BUCKETS;
```

The above command creates a Hive table whose data will be stored in 5 HDFS files. When writing data to the bucketed table, Hive evenly distributes the data among 5 HDFS files (`5` is specified in the `CLUSTERED BY` clause). The data in each bucket (file) is sorted by the `emp_id` column (`SORTED BY` clause).

Like with partitioning, keeping data in separate files can reduce total scan time. However, the number of buckets should be reasonable since too many small files can nullify the optimization efforts.

You can use buckets along with partitions for advanced splitting of your data. For example:

```
CREATE TABLE employees(
    emp_id int,
    first_name string,
    last_name string,
    office_num int
)
PARTITIONED BY (department_id int)
CLUSTERED BY (office_num) INTO 5 BUCKETS;
```

> **NOTE**
>
> The partition column is specified only in the `PARTITIONED BY` clause. The bucketing column is defined both in the table column specification and in the `CLUSTERED BY` clause.

Vectorization

Vectorized query execution is a built-in Hive optimization feature that helps to reduce the CPU load by processing blocks of rows rather than one row at a time. When processing blocks of rows, each column is stored in memory as an array (vector) of primitive data types. Simple arithmetic operations and comparisons are performed on the primitive values from the vector's contents. This approach reduces numerous condition checks for each column value, allows more efficient CPU utilization with heavy use of caching, and improves overall processing time.

Hive vectorization execution works only for specific data types and operations and has a limited support for user-defined functions.

By default, Hive vectorization is disabled. To use vectorized queries, the following requirements must be met:

- Use ORC as the storage format.
- Set `hive.vectorized.execution.enabled = true`.

To run a vectorized query, follow the steps:

1. Create a Hive table stored as ORC:

```
CREATE TABLE test_table_vectorized (
    id int, value string
) STORED AS ORC;
```

2. Enable Hive vectorization by setting the property:

```
SET hive.vectorized.execution.enabled=true;
```

3. Run `EXPLAIN EXTENDED` for an arbitrary aggregation command:

```
EXPLAIN EXTENDED
    SELECT COUNT(*) FROM test_table_vectorized;
```

The output:

```
+----------------------------------------------------+
|                      Explain                       |
+----------------------------------------------------+
| STAGE DEPENDENCIES:                                |
|   Stage-1 is a root stage                          |
|   Stage-0 depends on stages: Stage-1               |
|                                                    |
| STAGE PLANS:                                       |
|   Stage: Stage-1                                   |
|     Tez                                            |
|       DagId: hive_20240504214716_0b61644b-d042-4333-8566-05a678208479:12 |
|       Edges:                                       |
|         Reducer 2 <- Map 1 (CUSTOM_SIMPLE_EDGE)    |
|       DagName: hive_20240504214716_0b61644b-d042-4333-8566-05a678208479:12 |
|       Vertices:                                    |
|         Map 1                                      |
|             Map Operator Tree:                     |
|                 TableScan                          |
|                   alias: test_table_vectorized     |
|                   Statistics: Num rows: 2 Data size: 182 Basic stats: COMPLETE Column stats: COMPLETE |
|                   GatherStats: false               |
|                   Select Operator                  |
|                     Statistics: Num rows: 2 Data size: 182 Basic stats: COMPLETE Column stats: COMPLETE |
|                     Group By Operator              |
|                       aggregations: count()        |
|                       mode: hash                   |
|                       outputColumnNames: _col0     |
|                       Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE |
|                       Reduce Output Operator       |
|                         null sort order:           |
|                         sort order:                |
|                         Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: COMPLETE |
|                         tag: -1                    |
|                         value expressions: _col0 (type: bigint) |
|                         auto parallelism: false    |
|             Execution mode: vectorized             |
|             ...
|         Reducer 2                                  |
|             Execution mode: vectorized             |
|             ...
|                                                    |
|   Stage: Stage-0                                   |
|     Fetch Operator                                 |
|       limit: -1                                    |
|       Processor Tree:                              |
|         ListSink                                   |
|                                                    |
+----------------------------------------------------+
```

The output indicates that certain execution phases have the `Execution mode: vectorized` label. Setting `hive.vectorized.execution.enabled=false` will return the execution back to the row-at-a-time manner mode.

Cost-based optimization (CBO)

Hive cost-based optimization (CBO) is a built-in optimization feature that provides optimizations like query rewrites, JOINs re-ordering for more efficient processing, JOIN elimination on AST, etc.

The feature is based on Apache Calcite and comes disabled by default. To enable CBO, follow the steps below.

1. Set the following Hive configuration properties in ADCM (***Clusters → <clusterName> → Services → Hive → Primary configuration***):
   - `hive.cbo.enable=true`
   - `hive.compute.query.using.stats=true`
   - `hive.stats.fetch.column.stats=true`

2. Compute statistics for the table you need. This is required to prepare statistics data for those tables that should be optimized with CBO. The following command refreshes the table statistics in the metastore.

```
ANALYZE TABLE transactions COMPUTE STATISTICS;
```

Use TEZ

Apache TEZ is an execution engine for Hive. It is considered a much more flexible and powerful successor to the MapReduce framework. TEZ should be the preferred choice for Hive computation tasks as it reduces disk access count and improves overall execution time with DAGs.

In ADH, TEZ is available as a separate component (Hive Tez) of the Hive service. To use TEZ, your ADH cluster must have the Hive Tez component installed as a part of the Hive service. If it is installed, TEZ is used as the default Hive execution engine using the `hive.execution.engine` property.

Use ORC format

Using a suitable storage format is crucial for Hive performance. Optimized row columnar (ORC) is a file format developed specially for Hadoop workloads. A columnar format by nature, it is optimized for column-oriented operations like filtering and aggregations. Compared to other data formats supported by Hive (Parquet, Avro, TextFile, etc.), ORC excels at compression and speed, allowing to save petabytes of data storage.

The default storage format used by Hive is `TextFile`. To create a Hive table that stores data as ORC, ensure to use the `STORED AS ORC` clause, for example:

```
CREATE TABLE test_tbl_orc (
  id int,
  data String
) STORED AS ORC;
```

Hive backup and restore

Konstantin Alpashkin

Contents

To Table of Contents

Back up Hive table

Restore Hive table

Example

This article describes how to back up and restore Hive table data. At a high level, backing up a Hive table involves the following actions:

- Creating a copy of the Hive table data stored in the Hive warehouse in HDFS.
- Creating a dump of the Hive Metastore database.

Restoring a Hive table from a backup involves the reverse steps, namely:

- Copying the backup files to the corresponding Hive warehouse directory in HDFS.
- Importing the dump file into the Hive Metastore database.

More details about the backup/restore processes are presented below. Also, the article provides an end-to-end scenario that walks you through all the major operations required to back up/restore a Hive table.

Back up Hive table

To back up a managed Hive table, do the following:

1. Stop Hive via ADCM.
2. Back up Hive warehouse data in HDFS.
   Hive stores raw table data under the HDFS directory specified by the `hive.metastore.warehouse.dir` parameter (defaults to */apps/hive/warehouse*). To create a copy of the warehouse directory, use HDFS snapshots, distcp, or any other way to copy an HDFS directory.
3. Back up the Hive Metastore database.
   Apart from the raw table data stored in HDFS, Hive Metastore stores table metadata in a relational database. This metadata includes schema, partitions, buckets, indexes, and other system information which is vital for proper Hive table operation. All the metadata is stored in the `hive` database that should be dumped when creating a backup.

Restore Hive table

To restore a Hive table from the backup, do the following:

1. Stop Hive via ADCM.
2. Copy the HDFS backup files to the Hive warehouse location.
3. Import the SQL dump file to the database used by Hive Metastore.

> **NOTE**
>
> When you import a backup dump file to Hive Metastore, it overwrites the existing `hive` database with metadata. Thus, after the import of a backup dump, your Hive cluster will retain only those Hive tables that were present in the backup dump.

Example

The following scenario guides you through all the steps required to back up and restore a Hive table. The scenario assumes the full overwrite of Hive metadata when restoring the table. This means that only the tables from the backup will remain operable in the Hive cluster after the restore.

1. Create a test Hive table with some data:

```
SELECT * FROM transactions;
+----------------------+----------------------+--------------------------+------------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  |
+----------------------+----------------------+--------------------------+------------------------+
| 1                    | 1002                 | 10.00                    | 2023-01-01             |
| 8                    | 1003                 | 50.00                    | 2023-01-01             |
| 3                    | 1002                 | 30.00                    | 2023-01-02             |
| 4                    | 1001                 | 100.50                   | 2023-01-02             |
| 2                    | 1002                 | 20.00                    | 2023-01-03             |
| 6                    | 1001                 | 200.50                   | 2023-01-03             |
| 7                    | 1003                 | 50.00                    | 2023-01-03             |
| 5                    | 1001                 | 150.50                   | 2023-01-04             |
| 9                    | 1003                 | 75.00                    | 2023-01-04             |
+----------------------+----------------------+--------------------------+------------------------+
```

SQL for creating the test table

Use the following SQL to create and populate the test table using */bin/beeline*.

```
DROP TABLE IF EXISTS transactions;
CREATE TABLE transactions(txn_id int, acc_id int, txn_amount decimal(10,2), txn_date date);
INSERT INTO transactions VALUES
(1, 1002, 10.00, '2023-01-01'),
(2, 1002, 20.00, '2023-01-03'),
(3, 1002, 30.00, '2023-01-02'),
(4, 1001, 100.50, '2023-01-02'),
(5, 1001, 150.50, '2023-01-04'),
(6, 1001, 200.50, '2023-01-03'),
(7, 1003, 50.00, '2023-01-03'),
(8, 1003, 50.00, '2023-01-01'),
(9, 1003, 75.00, '2023-01-04');
```

2. In ADCM, stop the Hive service.
3. Create a snapshot of the Hive warehouse directory in HDFS. In this scenario, the required warehouse directory is */apps/hive/warehouse/transactions* (assuming `hive.metastore.warehouse.dir` points to *apps/hive/warehouse/*). To create a snapshot, follow the instructions:

   Make the */apps/hive/warehouse/transactions* directory snapshottable:

   ```
   $ sudo -u hdfs hdfs dfsadmin -allowSnapshot /apps/hive/warehouse/transactions
   ```

   The output:

   ```
   Allowing snapshot on /apps/hive/warehouse/transactions succeeded
   ```

   Then, create a snapshot of the warehouse directory:

   ```
   $ sudo -u hdfs hdfs dfs -createSnapshot /apps/hive/warehouse/transactions snapshot_transactions
   ```

   The output:

   ```
   Created snapshot /apps/hive/warehouse/{test-table-name}/.snapshot/snapshot_transactions
   ```

   Copy the snapshot to a temporary HDFS location since */apps/hive/warehouse/transactions* will be deleted on further steps. For example:

   ```
   $ sudo -u hdfs hadoop fs -mkdir /user/tmp
   $ sudo -u hdfs hadoop fs -chown hdfs:hdfs /user/tmp
   $ sudo -u hdfs hdfs dfs -cp /apps/hive/warehouse/transactions/.snapshot/snapshot_transactions /user/tmp
   ```

   Remove the snapshot from */apps/hive/warehouse/transactions*. This step is necessary because leaving the snapshot will otherwise prevent Hive from deleting the HDFS warehouse directory on further steps.

   ```
   $ sudo -u hdfs hdfs dfs -deleteSnapshot /apps/hive/warehouse/transactions/ snapshot_transactions
   ```

4. Dump the Hive Metastore database. In this scenario, Hive uses a MariaDB database that runs as an ADH service. For other database engines like Postgres, use the corresponding dump syntax. To dump the MariaDB database, on the ADH host with MariaDB server, run as root:

```
$ sudo mysqldump -uroot hive > hive_backup.sql
```

5. Start Hive.
6. Drop the test Hive table.

```
DROP TABLE transactions;
```

   Ensure that the HDFS directory */apps/hive/warehouse/transactions* has been removed.

7. Restore the test Hive table from the backup. For this, complete the steps below.

   Copy the snapshot files to the Hive warehouse location.

   ```
   $ sudo -u hdfs hdfs dfs -cp /user/tmp/snapshot_transactions /apps/hive/warehouse/transactions
   $ sudo -u hdfs hdfs dfs -ls /apps/hive/warehouse/transactions
   ```

   Then, import the SQL dump file into the Metastore database. For MariaDB, use the following command:

   ```
   $ sudo mysql hive < hive_backup.sql
   ```

8. Restart Hive.
9. Retrieve some data from the newly restored Hive table:

```
SELECT * FROM transactions;
```

   The output indicates that the test table has been successfully restored.

```
+----------------------+----------------------+--------------------------+------------------------+
| transactions.txn_id  | transactions.acc_id  | transactions.txn_amount  | transactions.txn_date  |
+----------------------+----------------------+--------------------------+------------------------+
| 1                    | 1002                 | 10.00                    | 2023-01-01             |
| 2                    | 1002                 | 20.00                    | 2023-01-03             |
| 3                    | 1002                 | 30.00                    | 2023-01-02             |
| 4                    | 1001                 | 100.50                   | 2023-01-02             |
| 5                    | 1001                 | 150.50                   | 2023-01-04             |
| 6                    | 1001                 | 200.50                   | 2023-01-03             |
| 7                    | 1003                 | 50.00                    | 2023-01-03             |
| 8                    | 1003                 | 50.00                    | 2023-01-01             |
| 9                    | 1003                 | 75.00                    | 2023-01-04             |
+----------------------+----------------------+--------------------------+------------------------+
```

High availability in Hive

Konstantin Alpashkin

Contents

To Table of Contents

JDBC connection string in HA

By default, the high availability (HA) mode for Hive is enabled out-of-the-box.

To be more specific, it is the HiveServer2 component that operates in the HA mode. The HA mode for the HiveServer2 component is enabled regardless of the number of components in the cluster. When new HiveServer2 components are added to the cluster, ZooKeeper automatically registers them and updates the JDBC connection string.

JDBC connection string in HA

In the HA mode, the JDBC connection string includes the ZooKeeper ensemble. You can find the up-to-date JDBC connection string on the Hive ***Info*** page in ADCM (***Clusters*** → ***Services*** → ***Hive*** → ***Info***).

Below are examples of JDBC connection strings depending on the security mechanism being used.

HA without SSL/Kerberos

High availability, insecure connection without SSL/Kerberos:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,
<cluster_host_N>:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/<cluster_id>/<namespace>
```

HA with SSL

High availability, secure connection with SSL:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,
<cluster_host_N>:2181;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/<cluster_id>/<namespace>;ssl=true;ss
lTrustStore=/tmp/truststore.jks;trustStorePassword=bigdata
```

HA with SSL+Kerberos

High availability, secure connection with SSL+Kerberos:

```
jdbc:hive2://<cluster_host_0>:2181,<cluster_host_1>:2181,
<cluster_host_N>:2181;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/<cluster_id>/<namespace>;ssl=true;ss
lTrustStore=/tmp/truststore.jks;trustStorePassword=bigdata;principal=hive/_HOST@EXAMPLE.COM
```

Where `EXAMPLE.COM` is your Kerberos realm, for example `RU-CENTRAL1.INTERNAL`.

NOTE

You can still connect to a HiveServer2 instance directly in the non-HA mode using the Thrift port (`10000` by default). In this case, the JDBC string looks like `jdbc:hive2://<cluster_host>:10000/`.

Logging in Hive

Konstantin Alpashkin

Contents

To Table of Contents

- Logs location and format
- Log types
  - HiveServer2 logs
  - Metastore logs
  - Hive Beeline logs
- Configure logging with ADCM
  - Common tasks
- View Hive logs in web UI

Under the hood Hive uses the Log4j 2 framework for logging. To configure Hive logging processes, use the ***Custom log4j.properties*** settings in ADCM. More details on using these settings are available in the Configure logging with ADCM section below.

Logs location and format

By default, Hive stores all logs in the local file system under the root logs directory */var/log/hive*. The root log directory is customizable and is defined by the `hive.log.dir` Log4j property. Hive logs are stored as plain text files with the *.log*/*.out* extensions. Most log events have the following structure:

```
[Timestamp] [Severity] [Thread that generates the event] [Event message]
```

For example:

```
2024-01-29T16:26:24,216  INFO [HiveServer2-Background-Pool: Thread-71] ql.Driver: Executing
command(queryId=hive_20240129162624_f4dfb3c4-3e27-4113-bbad-d524e54fcc26): drop table if exists
hive_main_through_beeline_check_75320
```

Log types

Hive logs are generated by separate Hive components, such as HiveServer2, Metastore, and Beeline. Each component has its own customizable Log4j configuration that defines the logging behavior for individual components. The major log types generated by Hive components are described below.

HiveServer2 logs

By default, Hive writes HiveServer2 logs to the files named *hive-server2.log*. These logs reflect the HiveServer2 activity including all the errors and exceptions raised in the HiveServer2 core. For example:

```
2024-01-30T13:42:04,048  INFO [1bd2c26d-2da1-4bea-aa8d-53e24b21fa9e HiveServer2-Handler-Pool: Thread-60] ql.Context: New scratch
dir is hdfs://adh/tmp/hive/hive/1bd2c26d-2da1-4bea-aa8d-53e24b21fa9e/hive_2024-01-30_13-42-02_298_4565019303774254377-2
2024-01-30T13:42:04,097 ERROR [1bd2c26d-2da1-4bea-aa8d-53e24b21fa9e HiveServer2-Handler-Pool: Thread-60] parse.CalcitePlanner:
CBO failed, skipping CBO.
org.apache.hadoop.hive.ql.parse.SemanticException: Line 1:12 Cannot insert into target table because column number/types are
different 'transactions': Table insclause-0 has 4 columns, but query has 1 columns.
        at org.apache.hadoop.hive.ql.parse.SemanticAnalyzer.genConversionSelectOperator(SemanticAnalyzer.java:8031) ~[hive-exec-3.
1.3.jar:3.1.3]
        at org.apache.hadoop.hive.ql.parse.SemanticAnalyzer.genFileSinkPlan(SemanticAnalyzer.java:7576) ~[hive-exec-3.1.3.jar:3.1.
3]
        at org.apache.hadoop.hive.ql.parse.SemanticAnalyzer.genPostGroupByBodyPlan(SemanticAnalyzer.java:10635) ~[hive-exec-3.1.3.
jar:3.1.3]
        at org.apache.hadoop.hive.ql.parse.SemanticAnalyzer.genBodyPlan(SemanticAnalyzer.java:10507) ~[hive-exec-3.1.3.jar:3.1.3]
        at org.apache.hadoop.hive.ql.parse.SemanticAnalyzer.genPlan(SemanticAnalyzer.java:11420) ~[hive-exec-3.1.3.jar:3.1.3]
...
```

The *hive-server2.out* file stores the redirected HiveServer2 output and gets cleaned up during the HiveServer2 startup.

Metastore logs

hive-metastore.log files store information related to Hive Metastore activity. For example:

```
2024-01-30T14:02:50,819  INFO [main] server.HiveServer2: Starting HiveServer2
2024-01-30T14:02:52,582  INFO [main] service.AbstractService: Service:OperationManager is inited.
2024-01-30T14:02:52,582  INFO [main] service.AbstractService: Service:SessionManager is inited.
2024-01-30T14:02:52,583  INFO [main] service.AbstractService: Service:CLIService is inited.
2024-01-30T14:02:52,583  INFO [main] service.AbstractService: Service:ThriftBinaryCLIService is inited.
...
```

Also, Metastore logs include audit information. For example:

```
$ grep 'audit' /var/log/hive/hive-metastore.log
```

The output:

```
2024-02-01T00:00:48,327  INFO [pool-6-thread-4] HiveMetaStore.audit: ugi=hive   ip=10.92.41.182 cmd=source:10.92.41.182
get_config_value: name=metastore.batch.retrieve.max defaultValue=50
...
```

Hive Beeline logs

Log files *hive-beeline.log* reflect information about the beeline shell operation.

Configure logging with ADCM

With the use of Log4j, configuring Hive logging processes assumes modification of key/value properties in **-log4j.properties* files. These files are available at different locations on hosts with corresponding Hive components and Hive reads these files during the startup. However, instead of editing these files manually, ADCM provides convenient settings to update Log4j configurations on all ADH cluster hosts at once. For this:

1. In ADCM, go to ***Clusters*** → ***<your_cluster_name>*** → ***Services*** → ***Hive*** → ***Primary Configuration*** and enable the ***Show advanced*** option.
2. In the ***Custom log4j.properties*** section, click the ***hive-log4j.properties*** item that corresponds to the Hive component you need and edit the Log4j properties. Log4j properties defined in ADCM UI will extend/overwrite the values in **-log4j.properties* files used by Hive. The detailed information about supported Log4j properties can be found in Log4j 2 documentation.
3. Save the Hive service configuration.
4. Restart the Hive service.

> **NOTE**
>
> In this article, the "update a Log4j property" action means updating the property through ADCM UI, including Hive restart.

Below is the default *hive-log4j.properties* file used by the HiveServer2 component. The major properties are highlighted to help you get started with Log4j configuration. For detailed reference on supported Log4j properties, see Log4j 2 documentation.

```
status = INFO
name = HiveLog4j2
packages = org.apache.hadoop.hive.ql.log

# list of properties
property.hive.log.level = INFO ①
property.hive.root.logger = DRFA ②
property.hive.log.dir = /var/log/hive ③
property.hive.log.file = hive-server2.log ④
property.hive.perflogger.log.level = INFO

# list of all appenders
appenders = console, DRFA ⑤

# console appender
appender.console.type = Console
appender.console.name = console
appender.console.target = SYSTEM_ERR
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = %d{ISO8601} %5p [%t] %c{2}: %m%n

# daily rolling file appender
appender.DRFA.type = RollingRandomAccessFile ⑥
appender.DRFA.name = DRFA
appender.DRFA.fileName = ${sys:hive.log.dir}/${sys:hive.log.file}
# Use %pid in the filePattern to append <process-id>@<host-name> to the filename if you want separate log files for different CLI
session
appender.DRFA.filePattern = ${sys:hive.log.dir}/${sys:hive.log.file}.%d{yyyy-MM-dd} ⑦
appender.DRFA.layout.type = PatternLayout
appender.DRFA.layout.pattern = %d{ISO8601} %5p [%t] %c{2}: %m%n ⑧
appender.DRFA.policies.type = Policies
appender.DRFA.policies.time.type = TimeBasedTriggeringPolicy ⑨
appender.DRFA.policies.time.interval = 1 ⑩
appender.DRFA.policies.time.modulate = true

# list of all loggers
loggers = NIOServerCnxn, ClientCnxnSocketNIO, DataNucleus, Datastore, JPOX, PerfLogger, AmazonAws, ApacheHttp ⑪

logger.NIOServerCnxn.name = org.apache.zookeeper.server.NIOServerCnxn
logger.NIOServerCnxn.level = WARN

logger.ClientCnxnSocketNIO.name = org.apache.zookeeper.ClientCnxnSocketNIO
logger.ClientCnxnSocketNIO.level = WARN

logger.DataNucleus.name = DataNucleus
logger.DataNucleus.level = ERROR

logger.Datastore.name = Datastore
logger.Datastore.level = ERROR

logger.JPOX.name = JPOX
logger.JPOX.level = ERROR

logger.AmazonAws.name=com.amazonaws
logger.AmazonAws.level = INFO

logger.ApacheHttp.name=org.apache.http
logger.ApacheHttp.level = INFO

logger.PerfLogger.name = org.apache.hadoop.hive.ql.log.PerfLogger
logger.PerfLogger.level = ${sys:hive.perflogger.log.level}

# root logger
rootLogger.level = ${sys:hive.log.level} ⑫
rootLogger.appenderRefs = root
rootLogger.appenderRef.root.ref = ${sys:hive.root.logger}
```

① Sets the root log level. The possible severity levels are: `ALL`, `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF`. Events with lower severity are not included in logs. For example, using `DEBUG` generates more verbose logs, whereas using `ERROR` will force Hive to log only errors and more critical events.

② Defines the root logger. The `DRFA` logger writes events to a file that gets rotated periodically.

③ Specifies the root directory to store log files. All log files generated using this Log4j configuration will be stored under the specified directory.

④ Specifies a name for the log file.

⑤ Defines a list of Log4j appenders which are responsible for delivering log events to various destinations like a file, a JMS queue, etc. In this configuration, the two default appenders route events to a file and to the console.

⑥ Uses the RollingRandomAccessFile appender to write log events to a file that gets rotated periodically.

⑦ Specifies a pattern used to rename log files when rotated.

⑧ Defines a pattern that converts a log event to a string.

⑨ Uses TimeBasedTriggeringPolicy that defines when the log file should be rotated.

⑩ Sets the interval (1 day) for rotating log files.

⑪ Defines a list of all active loggers. Each logger is responsible for generating log events related to certain functionality.

⑫ Defines the root logger properties. In Log4j, logger entities are hierarchical, so the root logger's properties get inherited by other non-root loggers (unless the inheriting logger overrides logger properties explicitly).

Common tasks

Below are common tasks that may arise while configuring Hive logging with Log4j.

Change default log level

To change the logging level (defaults to `INFO`), use the `property.hive.log.level` property. You can set a root log level which will be inherited by non-root Log4j loggers, or you can set log levels individually for a specific logger, for example `logger.PerfLogger.level = <OTHER_LEVEL>`.

Change default logs location

By default, Hive logs are stored under the */var/log/hive/* directory. To change this directory, modify the `property.hive.log.dir` property.

Enable/disable specific loggers

In Log4j, each logger entity is responsible for generating log events related to certain functionality. To exclude specific events from logs (for example, `PerfLogger` events), remove the corresponding logger from the `loggers = …` definition list, or set a higher severity for that logger.

Reflect query/session IDs in HiveServer2 logs

You can include Hive query/session IDs into HiveServer2 logs. For this, modify the `appender.DRFA.layout.pattern` property in the ***HiveServer2 hive-log4j.properties*** configuration as shown in the example below.

```
appender.DRFA.layout.pattern = %d{ISO8601} %5p [%t] %c{2}: QueryID: %X{queryId}; SessionID: %X{sessionId} %m%n
```

After this, Hive query/session IDs will be available in log events, for example:

```
2024-02-01T17:17:01,175  INFO [5818afd9-10b9-4d59-9efc-33fb4b5d2c2c HiveServer2-Handler-Pool: Thread-49] exec.SelectOperator:
QueryID: hive_20240201171700_bdc1be52-6070-41f9-bb5d-d9b2c33dd24b; SessionID: 5818afd9-10b9-4d59-9efc-33fb4b5d2c2c SELECT
struct<txn_id:int,acc_id:int,txn_amount:decimal(10,2),txn_date:date>
2024-02-01T17:17:01,175  INFO [5818afd9-10b9-4d59-9efc-33fb4b5d2c2c HiveServer2-Handler-Pool: Thread-49] exec.
ListSinkOperator: QueryID: hive_20240201171700_bdc1be52-6070-41f9-bb5d-d9b2c33dd24b; SessionID: 5818afd9-10b9-4d59-9efc-
33fb4b5d2c2c Initializing operator LIST_SINK[3]
...
```

Change logs rotation policy

By default, Hive logs get rotated on a daily basis. To change the rotation policy and make Hive additionally rotate logs based on size, use the SizeBasedTriggeringPolicy. The following properties instruct Log4j to rotate the log file upon reaching a specific size threshold (10 MB).

```
appender.DRFA.policies.size.type = SizeBasedTriggeringPolicy
appender.DRFA.policies.size.size = 10MB
# '%i' is mandatory if SizeBasedTriggeringPolicy is used with TimeBasedTriggeringPolicy:
appender.DRFA.filePattern = ${sys:hive.log.dir}/${sys:hive.log.file}.%d{yyyy-MM-dd}-%i
```

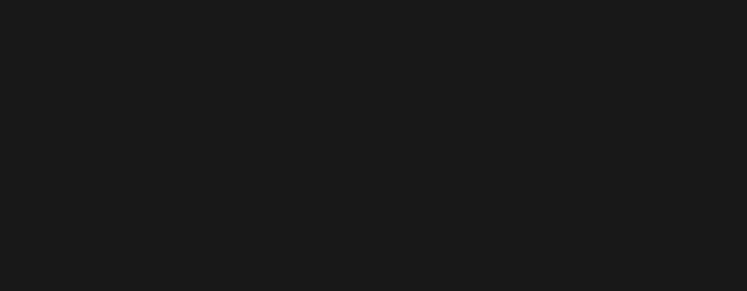
Auto-compress logs

When a log file gets rotated, you might want to compress the "old" log file. For this, specify the desired compression format within the `filePattern` property as shown in the following example. If `filePattern` ends with ".gz", ".zip", ".bz2", ".deflate", ".pack200", ".zst", or ".xz", the resulting archive will be compressed using the compression scheme that matches the suffix.

```
# '.gz' ensures that the files will be compressed using gzip
appender.DRFA.filePattern = ${sys:hive.log.dir}/${sys:hive.log.file}.%d{yyyy-MM-dd}-%i.log.gz
```

View Hive logs in web UI

HiveServer2 web UI allows you to view plain-text logs on the Local logs page. This page lists the contents of the root logs directory (defined via `hive.log.dir`) where you can navigate between log files and view them as plain text.



Directory: /logs/

Name ⇧	Last Modified	Size
beeline-logs/	Jan 30, 2024 12:51:27 PM	6 bytes
hive-metastore.log	Jan 30, 2024 8:12:52 PM	507,604 bytes
hive-metastore.log.2024-01-29	Jan 29, 2024 4:26:24 PM	69,737 bytes
hive-server2.log	Jan 30, 2024 8:12:52 PM	612,905 bytes
hive-server2.log.2024-01-29	Jan 29, 2024 4:26:24 PM	155,835 bytes
hive-server2.out	Jan 30, 2024 2:28:58 PM	856 bytes
metastore-logs/	Jan 30, 2024 12:50:42 PM	6 bytes
test/	Jan 30, 2024 10:36:43 AM	46 bytes
test1/	Jan 30, 2024 10:36:43 AM	47 bytes

Local logs page

Hive ACID transactions

Konstantin Alpashkin

Contents

To Table of Contents

- Delta files and compactions
- Enable ACID transactions
- Work with transactional tables
  - Create a table
  - Convert non-ACID tables to ACID
  - Insert and update data
- Special Hive transaction commands
  - COMPACT
  - SHOW TRANSACTIONS
  - SHOW LOCKS
- Limitations

Hive supports ACID transactions that provide the following features:

- **Atomicity**. Ensures that transactions are either executed fully or not executed at all. In other words, transactions are indivisible, and if one part of a transaction fails, the entire transaction is rolled back, leaving the system in its original state.
- **Consistency**. Ensures that the database remains healthy before and after committing a transaction. The transactions preserve the integrity of the data and the constraints defined on it.
- **Isolation**. Ensures that the parallel execution of multiple transactions does not interfere with each other.
- **Durability**. Once a transaction is committed, its effects are permanent and will not be lost, even in the case of a system failure.

Together, these properties ensure reliability, integrity, and robustness of Hive transactions. Hive supports ACID features at the row level, meaning that one application (for example, a streaming Kafka-powered app) can continuously add rows while other apps can still read from the same partition without interfering with each other.

Delta files and compactions

When you insert or update data in a transactional table, Hive creates a set of delta files each reflecting the changes made to the table (partition) in scope of one transaction. When Hive clients request data, Hive analyzes the delta files to include the updates from individual transactions into the result set, thus returning the most up-to-date table state — this is the basic idea of Hive transactions.

To avoid accumulation of too many delta files (which inevitably leads to performance degradation), Hive performs regular cleanups called compactions. The compaction process automatically runs in the background at regular intervals without affecting concurrent reads/writes.

There are two types of compactions:

- **Minor**. Merges several delta files to a single file.
- **Major**. Rewrites one or more delta files and the base file to a new base file. This process is far more resource-expensive than the minor compaction.

Besides automatic regular compaction runs, you can trigger a compaction manually using the COMPACT command.

Enable ACID transactions

By default, Hive ACID transactions are disabled. To work with transactional tables, enable the ***ACID Transactions*** option in ADCM (***Clusters → <clusterName> → Services → Hive → Primary configuration***) and then restart your Hive service. When you activate this option, ADCM automatically updates all the Hive configuration properties required to work with ACID transactions, including the transaction manager (`hive.txn.manager`).

Work with transactional tables

Create a table

To create a fully-ACID transactional table, the following is required:

- Hive ACID transactions are enabled in ADCM.
- The created table is bucketed.
- ORC is used as the storage format. Currently, this is the only format supported for transactional tables.
- The `TBLPROPERTIES ("transactional"="true")` clause is used.

For example:

```
CREATE TABLE `transactions`
    (`txn_id` int,
    `acc_id` int,
    `txn_amount` decimal(10,2),
    `txn_date` date)
    CLUSTERED BY (acc_id) INTO 3 BUCKETS ①
    STORED AS ORC ②
    TBLPROPERTIES ("transactional"="true"); ③
```

① Splits the new table into X buckets.

② Sets ORC as the storage format.

③ Sets the property to create a transactional table.

Using `DESCRIBE FORMATTED <table_name>`, you can ensure that the new table has been created as transactional. The following output row indicates this.

```
...
| transactional                                      | true                                               |
...
```

If you observe the following error while creating a transactional table, ensure that the ***ACID Transactions*** option is set to `true` in ADCM and your Hive service was restarted.

```
This command is not allowed on an ACID table <table_name> with a non-ACID transaction manager
```

Convert non-ACID tables to ACID

You can convert a non-ACID Hive table into a full-ACID one. The major requirement is that the source table should be a managed Hive table with ORC used as the storage format. For example:

```
ALTER TABLE <non-acid_table> SET TBLPROPERTIES ("transactional"="true");
```

To make an external table ACID, first, you have to make the table a managed one. For example:

```
ALTER TABLE <external_table_name> SET TBLPROPERTIES("EXTERNAL"="false")
```

> **NOTE**
>
> Once a Hive table is created as an ACID table, you cannot convert it back to a non-ACID table.

Insert and update data

Writing or updating data in a transactional table is a no fancy operation. The `INSERT`/`UPDATE`/`DELETE` syntax is standard, for example:

```
INSERT INTO transactions VALUES
(1, 1002, 10.00, '2023-01-01'),
(2, 1001, 100.50, '2023-01-02'),
(3, 1003, 50.00, '2023-01-03');
```

The main feature is when you update a transactional table, Hive stores the updates as delta files in the HDFS warehouse directory. To view the delta files created as a result of the `INSERT` query above, run the command:

```
$ hdfs dfs -ls /apps/hive/warehouse/transactions/delta_0000001_0000001_0000
```

> **TIP**
>
> The numbers in *delta_0000001_0000001_0000* indicate the transaction ID.

The output:

```
Found 3 items
-rw-r--r--   3 hive hadoop          1 2024-03-28 17:21
/apps/hive/warehouse/transactions/delta_0000001_0000001_0000/_orc_acid_version ①
-rw-r--r--   3 hive hadoop        906 2024-03-28 17:21 /apps/hive/warehouse/transactions/delta_0000001_0000001_0000/bucket_00000
②
-rw-r--r--   3 hive hadoop        959 2024-03-28 17:21 /apps/hive/warehouse/transactions/delta_0000001_0000001_0000/bucket_00002
②
```

① Stores the ORC format version.

② These files store the inserted/updated data split by buckets.

Special Hive transaction commands

Below are several commands specific to Hive transactional mechanism.

COMPACT

You can manually trigger a major/minor compaction using the following syntax:

```
ALTER TABLE <table_name> COMPACT 'major|minor';
```

Running this command enqueues a compaction operation and outputs the queued compaction ID.

```
...
Compaction enqueued with id {id}
```

To view the list of all compactions, use the command:

```
SHOW COMPACTIONS;
```

The sample output:

```
+---------------+-----------+----------------+-------------+--------+------------+-----------+-------------+---------------+------
--------+
| compactionid  |  dbname   |    tabname     |  partname   |  type  |   state    | workerid  |  starttime  |   duration    |
hadoopjobid  |
+---------------+-----------+----------------+-------------+--------+------------+-----------+-------------+---------------+------
--------+
| CompactionId  | Database  | Table          | Partition   | Type   | State      | Worker    | Start Time  | Duration(ms)  |
HadoopJobId  |
| 1             | default   | transactions   |  ---        | MINOR  | initiated  |  ---      |  ---        |  ---          | None
|
+---------------+-----------+----------------+-------------+--------+------------+-----------+-------------+---------------+------
--------+
```

SHOW TRANSACTIONS

This command returns a list of all currently open and aborted transactions in the system, for example:

```
+-----------------+--------------------+----------------+----------------------+-------+--------------------------------+
|      txnid      |       state        |  startedtime   |  lastheartbeattime   | user  |              host              |
+-----------------+--------------------+----------------+----------------------+-------+--------------------------------+
| Transaction ID  | Transaction State  | Started Time   | Last Heartbeat Time  | User  | Hostname                       |
| 10              | ABORTED            | 1711646536000  | 1711646536000        | hive  | ka-adh-2.ru-central1.internal  |
| 29              | OPEN               | 1711970603000  | 1711970603000        | hive  | ka-adh-2.ru-central1.internal  |
+-----------------+--------------------+----------------+----------------------+-------+--------------------------------+
```

SHOW LOCKS

This command displays the locks acquired on a table or a partition. A sample output below.

```
+----------+------------------+----------------+------------+-------------+---------------+--------------+-----------------+-----
------------+----------------+-------+--------------------------------+----------------------------------------------------------+
|  lockid  |     database     |     table      | partition  | lock_state  |  blocked_by   |  lock_type   | transaction_id  |
last_heartbeat  |  acquired_at   | user  |           hostname             |                      agent_info                          |
+----------+------------------+----------------+------------+-------------+---------------+--------------+-----------------+-----
------------+----------------+-------+--------------------------------+----------------------------------------------------------+
| Lock ID  | Database         | Table          | Partition  | State       | Blocked By    | Type         | Transaction ID  | Last
Heartbeat  | Acquired At    | User  | Hostname                       | Agent Info                                               |
| 41.1     | _dummy_database  | _dummy_table   | NULL       | ACQUIRED    |               | SHARED_READ  | 50              | 0
| 1711992560000  | hive  | ka-adh-2.ru-central1.internal  | hive_20240401172919_fbb1738c-5736-458f-956e-d89ea7d3a727 |
| 41.2     | default          | transactions2  | NULL       | ACQUIRED    |               | SHARED_READ  | 50              | 0
| 1711992560000  | hive  | ka-adh-2.ru-central1.internal  | hive_20240401172919_fbb1738c-5736-458f-956e-d89ea7d3a727 |
+----------+------------------+----------------+------------+-------------+---------------+--------------+-----------------+-----
------------+----------------+-------+--------------------------------+----------------------------------------------------------+
```

Limitations

There are several limitations for Hive ACID transactions:

- Unlike traditional SQL transactions, `BEGIN`, `COMMIT`, and `ROLLBACK` operations are not supported. All Hive transactions are auto-commit.
- Only ORC storage format is supported.
- A Hive table must be bucketed to make use of ACID transactions.
- External tables cannot be made ACID.
- Reading/writing to an ACID table from a non-ACID session is not allowed. In other words, the Hive transaction manager must be set to `org.apache.hadoop.hive.ql.lockmgr.DbTxnManager` in order to work with ACID tables.
- The existing ZooKeeper and in-memory lock managers are not compatible with transactions.
- Using Oracle as the Metastore DB and `datanucleus.connectionPoolingType=BONECP` may generate intermittent "No such lock ..." error and "No such transaction..." errors. Setting `datanucleus.connectionPoolingType=DBCP` is recommended in this case.
- `LOAD DATA` is not supported with transactional tables.

Work with Iceberg tables in Hive

Konstantin Alpashkin

Contents

To Table of Contents

DDL commands
  - Create an Iceberg table
  - CREATE TABLE AS SELECT
  - CREATE TABLE LIKE
  - Create a partitioned table
DML commands
  - SELECT
  - INSERT
Custom Hive catalogs
Limitations

Apache Iceberg is an open, high-performance format for large analytic tables. The ADH Hive service adopts this format to enhance the performance of Hive tables.

NOTE

Hive supports Iceberg tables starting with ADH 3.2.4.3.

In Hive, the Iceberg table format is available out-of-the-box. The support for Iceberg tables is based on Hive StorageHandlers. This means that to create an Iceberg table in Hive, you should specify the `org.apache.iceberg.mr.hive.HiveIcebergStorageHandler` storage handler in the `STORED BY` clause. The Iceberg tables support is enabled by default and can be toggled using the `iceberg.engine.hive.enabled` property (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Hive*** → ***Primary Configuration*** → ***hive-site.xml***). Also, the `engine.hive.enabled` property can be set programmatically as a table property during the creation of an Iceberg table. An example is below:

```
Catalog catalog = ... ;
    Map<String, String> tableProperties=Maps.newHashMap();
    tableProperties.put(TableProperties.ENGINE_HIVE_ENABLED,"true");
    catalog.createTable(tableId,schema,spec,tableProperties);
```

NOTE

The table property configuration overrides the global ADH configuration.

As of ADH 3.2.4.3, the support for Iceberg tables in Hive comprises the following capabilities:

- Create a table.
- Drop a table.
- Select data.
- Insert data.

DDL commands

Create an Iceberg table

To create an Iceberg table, append the `STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler'` clause to the standard Hive `CREATE TABLE` statement.

Example:

```
CREATE TABLE transactions (txn_id int, acc_id int, txn_amount decimal(10,2), txn_date date)
STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler';
```

Run the `DESCRIBE FORMATTED` command to retrieve information about the newly created table:

```
DESCRIBE FORMATTED transactions;
```

Sample output

```
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
|           col_name            |                      data_type                     |                        comment
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
| # col_name                    | data_type                                          | comment
|
| id                            | int                                                | from deserializer
|
| value                         | string                                             | from deserializer
|
|                               | NULL                                               | NULL
|
| # Detailed Table Information  | NULL                                               | NULL
|
| Database:                     | default                                            | NULL
|
| OwnerType:                    | USER                                               | NULL
|
| Owner:                        | hive                                               | NULL
|
| CreateTime:                   | Wed Jun 05 14:09:36 UTC 2024                       | NULL
|
| LastAccessTime:               | UNKNOWN                                            | NULL
|
| Retention:                    | 0                                                  | NULL
|
| Location:                     | hdfs://adh/apps/hive/warehouse/test_iceberg_table  | NULL
|
| Table Type:                   | MANAGED_TABLE                                      | NULL
|
| Table Parameters:             | NULL                                               | NULL
|
|                               | bucketing_version                                  | 2
|
|                               | current-schema                                     | {\"type\":\"struct\",\"schema-
id\":0,\"fields\":[{\"id\":1,\"name\":\"id\",\"required\":false,\"type\":\"int\"},
{\"id\":2,\"name\":\"value\",\"required\":false,\"type\":\"string\"}]} |
|                               | engine.hive.enabled                                | true
|
|                               | external.table.purge                               | TRUE
|
|                               | metadata_location                                  |
hdfs://adh/apps/hive/warehouse/test_iceberg_table/metadata/00000-fa65f392-3b34-4e51-915b-8e2aae261098.metadata.json |
|                               | numFiles                                           | 0
|
|                               | numRows                                            | 0
|
|                               | rawDataSize                                        | 0
|
|                               | snapshot-count                                     | 0
|
|                               | storage_handler                                    | org.apache.iceberg.mr.hive.
HiveIcebergStorageHandler |
|                               | table_type                                         | ICEBERG
|
|                               | totalSize                                          | 0
|
|                               | transient_lastDdlTime                              | 1717596576
|
|                               | uuid                                               | f9da6135-095e-491f-919e-f2194253b8f8
|
|                               | write.parquet.compression-codec                    | zstd
|
|                               | NULL                                               | NULL
|
| # Storage Information         | NULL                                               | NULL
|
| SerDe Library:                | org.apache.iceberg.mr.hive.HiveIcebergSerDe        | NULL
|
| InputFormat:                  | org.apache.iceberg.mr.hive.HiveIcebergInputFormat  | NULL
|
| OutputFormat:                 | org.apache.iceberg.mr.hive.HiveIcebergOutputFormat | NULL
|
| Compressed:                   | No                                                 | NULL
|
| Num Buckets:                  | 0                                                  | NULL
|
| Bucket Columns:               | []                                                 | NULL
|
| Sort Columns:                 | []                                                 | NULL
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
```

The `table_type` parameter in the output indicates that the table has been created as an Iceberg table.

CREATE TABLE AS SELECT

You can also use the Hive's native `CREATE TABLE AS <tbl_name>` (CTAS) syntax to create an Iceberg table based on an existing Hive table. In this case, the major distinction is that an Iceberg table is created at the very beginning of the query execution, whereas the data is inserted after the query is completed. Thus, for some period of time, the table may exist but contain no data.

```
CREATE TABLE transactions_ctas
STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler' AS
SELECT * FROM transactions;
```

Sample DESCRIBE FORMATTED output

```
DESCRIBE FORMATTED transactions_ctas;
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
|           col_name            |                      data_type                     |                        comment
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
| # col_name                    | data_type                                          | comment
|
| id                            | int                                                | from deserializer
|
|                               | NULL                                               | NULL
|
| # Detailed Table Information  | NULL                                               | NULL
|
| Database:                     | default                                            | NULL
|
| OwnerType:                    | USER                                               | NULL
|
| Owner:                        | hive                                               | NULL
|
| CreateTime:                   | Mon Jun 03 20:59:25 UTC 2024                       | NULL
|
| LastAccessTime:               | UNKNOWN                                            | NULL
|
| Retention:                    | 0                                                  | NULL
|
| Location:                     | hdfs://adh/apps/hive/warehouse/test_iceberg_table1 | NULL
|
| Table Type:                   | MANAGED_TABLE                                      | NULL
|
| Table Parameters:             | NULL                                               | NULL
|
|                               | current-schema                                     | {\"type\":\"struct\",\"schema-
id\":0,\"fields\":[{\"id\":1,\"name\":\"id\",\"required\":false,\"type\":\"int\"}]} |
|                               | engine.hive.enabled                                | true
|
|                               | external.table.purge                               | TRUE
|
|                               | metadata_location                                  |
hdfs://adh/apps/hive/warehouse/test_iceberg_table1/metadata/00000-df221cec-ab76-4de9-95cd-ef7605c8e79f.metadata.json |
|                               | numFiles                                           | 0
|
|                               | numRows                                            | 0
|
|                               | rawDataSize                                        | 0
|
|                               | snapshot-count                                     | 0
|
|                               | storage_handler                                    | org.apache.iceberg.mr.hive.
HiveIcebergStorageHandler |
|                               | table_type                                         | ICEBERG
|
|                               | totalSize                                          | 0
|
|                               | transient_lastDdlTime                              | 1717448365
|
|                               | uuid                                               | eb1933a0-43d9-4dd2-9426-cf7ace5063c8
|
|                               | write.parquet.compression-codec                    | zstd
|
|                               | NULL                                               | NULL
|
| # Storage Information         | NULL                                               | NULL
|
| SerDe Library:                | org.apache.iceberg.mr.hive.HiveIcebergSerDe        | NULL
|
| InputFormat:                  | org.apache.iceberg.mr.hive.HiveIcebergInputFormat  | NULL
|
| OutputFormat:                 | org.apache.iceberg.mr.hive.HiveIcebergOutputFormat | NULL
|
| Compressed:                   | No                                                 | NULL
|
| Num Buckets:                  | 0                                                  | NULL
|
| Bucket Columns:               | []                                                 | NULL
|
| Sort Columns:                 | []                                                 | NULL
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
```

CREATE TABLE LIKE

You can also use the `CREATE TABLE LIKE <tbl_name>` syntax to create an empty Iceberg table based on the definition of another Hive table.

Example:

```
CREATE TABLE transactions_ctlt LIKE transactions
STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler';
```

Sample DESCRIBE FORMATTED output

```
DESCRIBE FORMATTED transactions_ctlt;
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
|           col_name            |                      data_type                     |                        comment
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
| # col_name                    | data_type                                          | comment
|
| txn_id                        | int                                                | from deserializer
|
| acc_id                        | int                                                | from deserializer
|
| txn_amount                    | decimal(10,2)                                      | from deserializer
|
| txn_date                      | date                                               | from deserializer
|
|                               | NULL                                               | NULL
|
| # Detailed Table Information  | NULL                                               | NULL
|
| Database:                     | default                                            | NULL
|
| OwnerType:                    | USER                                               | NULL
|
| Owner:                        | hive                                               | NULL
|
| CreateTime:                   | Fri Jun 07 10:08:08 UTC 2024                       | NULL
|
| LastAccessTime:               | UNKNOWN                                            | NULL
|
| Retention:                    | 0                                                  | NULL
|
| Location:                     | hdfs://adh/apps/hive/warehouse/transactions_ctlt   | NULL
|
| Table Type:                   | MANAGED_TABLE                                      | NULL
|
| Table Parameters:             | NULL                                               | NULL
|
|                               | current-schema                                     | {\"type\":\"struct\",\"schema-
id\":0,\"fields\":[{\"id\":1,\"name\":\"txn_id\",\"required\":false,\"type\":\"int\"},
{\"id\":2,\"name\":\"acc_id\",\"required\":false,\"type\":\"int\"},
{\"id\":3,\"name\":\"txn_amount\",\"required\":false,\"type\":\"decimal(10, 2)\"},
{\"id\":4,\"name\":\"txn_date\",\"required\":false,\"type\":\"date\"}]} |
|                               | engine.hive.enabled                                | true
|
|                               | external.table.purge                               | TRUE
|
|                               | metadata_location                                  |
hdfs://adh/apps/hive/warehouse/transactions_ctlt/metadata/00000-72d6ad62-e7c4-4e53-97e9-be7f82d5ced9.metadata.json |
|                               | numFiles                                           | 0
|
|                               | numRows                                            | 0
|
|                               | rawDataSize                                        | 0
|
|                               | snapshot-count                                     | 0
|
|                               | storage_handler                                    | org.apache.iceberg.mr.hive.
HiveIcebergStorageHandler |
|                               | table_type                                         | ICEBERG
|
|                               | totalSize                                          | 0
|
|                               | transient_lastDdlTime                              | 1717754888
|
|                               | uuid                                               | d4e3811a-7a36-47b5-bb12-0948f251f6dc
|
|                               | write.parquet.compression-codec                    | zstd
|
|                               | NULL                                               | NULL
|
| # Storage Information         | NULL                                               | NULL
|
| SerDe Library:                | org.apache.iceberg.mr.hive.HiveIcebergSerDe        | NULL
|
| InputFormat:                  | org.apache.iceberg.mr.hive.HiveIcebergInputFormat  | NULL
|
| OutputFormat:                 | org.apache.iceberg.mr.hive.HiveIcebergOutputFormat | NULL
|
| Compressed:                   | No                                                 | NULL
|
| Num Buckets:                  | 0                                                  | NULL
|
| Bucket Columns:               | []                                                 | NULL
|
| Sort Columns:                 | []                                                 | NULL
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
```

Create a partitioned table

To create a partitioned table, use the following syntax:

```
CREATE TABLE transactions_partitioned (
    txn_id int, txn_amount decimal(10,2), txn_date date)
PARTITIONED BY (acc_id int)
STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler';
```

Run the `DESCRIBE FORMATTED` command for the newly created table and in the command's output, search for the `default-partition-spec` property. Its value stores the partition specification in the Iceberg format.

Sample DESCRIBE FORMATTED output

```
DESCRIBE FORMATTED transactions_partitioned;
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
|           col_name            |                      data_type                     |                        comment
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------+
| # col_name                    | data_type                                          | comment
|
| txn_id                        | int                                                | from deserializer
|
| txn_amount                    | decimal(10,2)                                      | from deserializer
|
| txn_date                      | date                                               | from deserializer
|
| acc_id                        | int                                                | from deserializer
|
|                               | NULL                                               | NULL
|
| # Detailed Table Information  | NULL                                               | NULL
|
| Database:                     | default                                            | NULL
|
| OwnerType:                    | USER                                               | NULL
|
| Owner:                        | hive                                               | NULL
|
| CreateTime:                   | Thu Jun 06 13:57:31 UTC 2024                       | NULL
|
| LastAccessTime:               | UNKNOWN                                            | NULL
|
| Retention:                    | 0                                                  | NULL
|
| Location:                     | hdfs://adh/apps/hive/warehouse/transactions        | NULL
|
| Table Type:                   | MANAGED_TABLE                                      | NULL
|
| Table Parameters:             | NULL                                               | NULL
|
|                               | bucketing_version                                  | 2
|
|                               | current-schema                                     | {\"type\":\"struct\",\"schema-
id\":0,\"fields\":[{\"id\":1,\"name\":\"txn_id\",\"required\":false,\"type\":\"int\"},
{\"id\":2,\"name\":\"txn_amount\",\"required\":false,\"type\":\"decimal(10, 2)\"},
{\"id\":3,\"name\":\"txn_date\",\"required\":false,\"type\":\"date\"},
{\"id\":4,\"name\":\"acc_id\",\"required\":false,\"type\":\"int\"}]} |
|                               | default-partition-spec                             | {\"spec-id\":0,\"fields\":
[{\"name\":\"acc_id\",\"transform\":\"identity\",\"source-id\":4,\"field-id\":1000}]} |
|                               | engine.hive.enabled                                | true
|
|                               | external.table.purge                               | TRUE
|
|                               | metadata_location                                  |
hdfs://adh/apps/hive/warehouse/transactions/metadata/00000-b6dde9b3-d1cc-4f92-828e-5dab83b1e5e8.metadata.json |
|                               | snapshot-count                                     | 0
|
|                               | storage_handler                                    | org.apache.iceberg.mr.hive.
HiveIcebergStorageHandler |
|                               | table_type                                         | ICEBERG
|
|                               | transient_lastDdlTime                              | 1717682251
|
|                               | uuid                                               | c6205d6d-ddca-4b04-a780-8b8e1048db0b
|
|                               | write.parquet.compression-codec                    | zstd
|
|                               | NULL                                               | NULL
|
| # Storage Information         | NULL                                               | NULL
|
| SerDe Library:                | org.apache.iceberg.mr.hive.HiveIcebergSerDe        | NULL
|
| InputFormat:                  | org.apache.iceberg.mr.hive.HiveIcebergInputFormat  | NULL
|
| OutputFormat:                 | org.apache.iceberg.mr.hive.HiveIcebergOutputFormat | NULL
|
| Compressed:                   | No                                                 | NULL
|
| Num Buckets:                  | 0                                                  | NULL
|
| Bucket Columns:               | []                                                 | NULL
|
| Sort Columns:                 | []                                                 | NULL
|
+-------------------------------+----------------------------------------------------+----------------------------------------
------------
```

DML commands

As of ADH 3.2.4.3, Hive supports the `SELECT` and `INSERT` DML operations for Iceberg tables.

NOTE

To run DML commands on Iceberg tables, Hive must use the MapReduce execution engine.

To change the Hive execution engine, use the `hive.execution.engine` property in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Hive*** → ***Primary Configuration*** → ***hive-site.xml***). After changing the property, a Hive restart is required.

SELECT

The syntax of the `SELECT` statement to run on an Iceberg table is identical to a regular Hive `SELECT` command. However, under the hood Hive runs a different set of actions to benefit from the Iceberg format. The distinctions are present in both query compilation and execution phases.

Here's a brief description of Iceberg-specific events taking place under the hood when Hive selects data from an Iceberg table:

1. Hive requests the Iceberg catalog to get the table's **metadata file** that stores the table definition. The Iceberg catalog returns a pointer to the metadata file.
2. Hive reads the metadata file and gets the **manifest list** location. This entry carries information about **manifest files** that make up the snapshot, define partitions and partition boundaries, etc. Also, if the requested table is partitioned, Hive reads the partition specification, thus, being able to apply certain partition optimizations at this step.
3. Hive analyzes each individual manifest file from the manifest files list. Each manifest file carries information about the data files that store actual physical data.
4. Finally, Hive scans the data files. If the table is partitioned, the engine deals only with data files that match the `WHERE` clause or other query filters. The major Iceberg format advantage lies in this phase — by this time, the engine has enough information to scan only those data files that are relevant to the given `SELECT` query. This is made possible due to the data being tracked at the file level (rather than the directory level as implemented in the traditional Hive model).

INSERT

For Iceberg tables, Hive supports the traditional `INSERT INTO` operation.

Example:

```
INSERT INTO transactions VALUES
(1, 1002, 10.00, '2023-01-01'),
(2, 1002, 20.00, '2023-01-03'),
(3, 1002, 30.00, '2023-01-02'),
(4, 1001, 100.50, '2023-01-02'),
(5, 1001, 150.50, '2023-01-04'),
(6, 1001, 200.50, '2023-01-03'),
(7, 1003, 50.00, '2023-01-03'),
(8, 1003, 50.00, '2023-01-01'),
(9, 1003, 75.00, '2023-01-04');
```

Multi-table inserts are also supported, but they are not atomic. The commits are made on a per-table basis. During the commit process, clients may observe partial changes to the table contents and failures can leave partial changes committed. Changes within a single table remain atomic.

The following example shows how to populate two Iceberg tables with data from an existing Hive table using one multi-insert operation.

```
FROM transactions
   INSERT INTO transactions11 SELECT txn_id, txn_date
   INSERT INTO transactions22 SELECT txn_id, acc_id;
```

Custom Hive catalogs

From the Hive operational perspective, there is only one global data catalog — `HiveCatalog` that stores data in HDFS under the */apps/hive/warehouse/* location. However, when using Iceberg tables, you can have several different data catalog types, such as `Hive`, `Hadoop`, or a custom catalog implementation.

Iceberg also allows loading a table directly based on its path in HDFS. In this case, a table does not belong to any catalog. Such cross-catalog, path-based tables can be used as temporary or intermediate tables, for example, for JOIN cases.

The following example creates another Hive catalog and specifies a custom warehouse location for the catalog.

```
SET iceberg.catalog.my_test_hive.type=hive;
SET iceberg.catalog.my_test_hive.warehouse=hdfs:/adh/apps/hive/custom_catalog/warehouse;
```

Limitations

There are several limitations for Iceberg tables support in Hive. These restrictions are related to the 3.x Hive version being used in ADH. In the upcoming ADH releases, the shift to Hive 4.x will extend Iceberg tables support and eliminate the limitations listed below.

- As of ADH 3.2.4.3, Hive supports the following DML/DDL operations:
  - `CREATE TABLE`
  - `DROP TABLE`
  - `SELECT`
  - `INSERT`
- As of ADH 3.2.4.3, Iceberg tables in Hive are created using the `STORED BY 'org.apache.iceberg.mr.hive.HiveIcebergStorageHandler'` clause rather than `STORED BY ICEBERG`.
- Starting with Apache Iceberg 0.11.0, when using Hive with Tez you also have to disable vectorization (`hive.vectorized.execution.enabled=false`).

Hive service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Hive service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Hive service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Hive components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM
Manage Ranger plugin	Enables or disables Ranger plugin for Hive

Hive components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

HUE UI overview

Vladimir Adamenkov

Contents

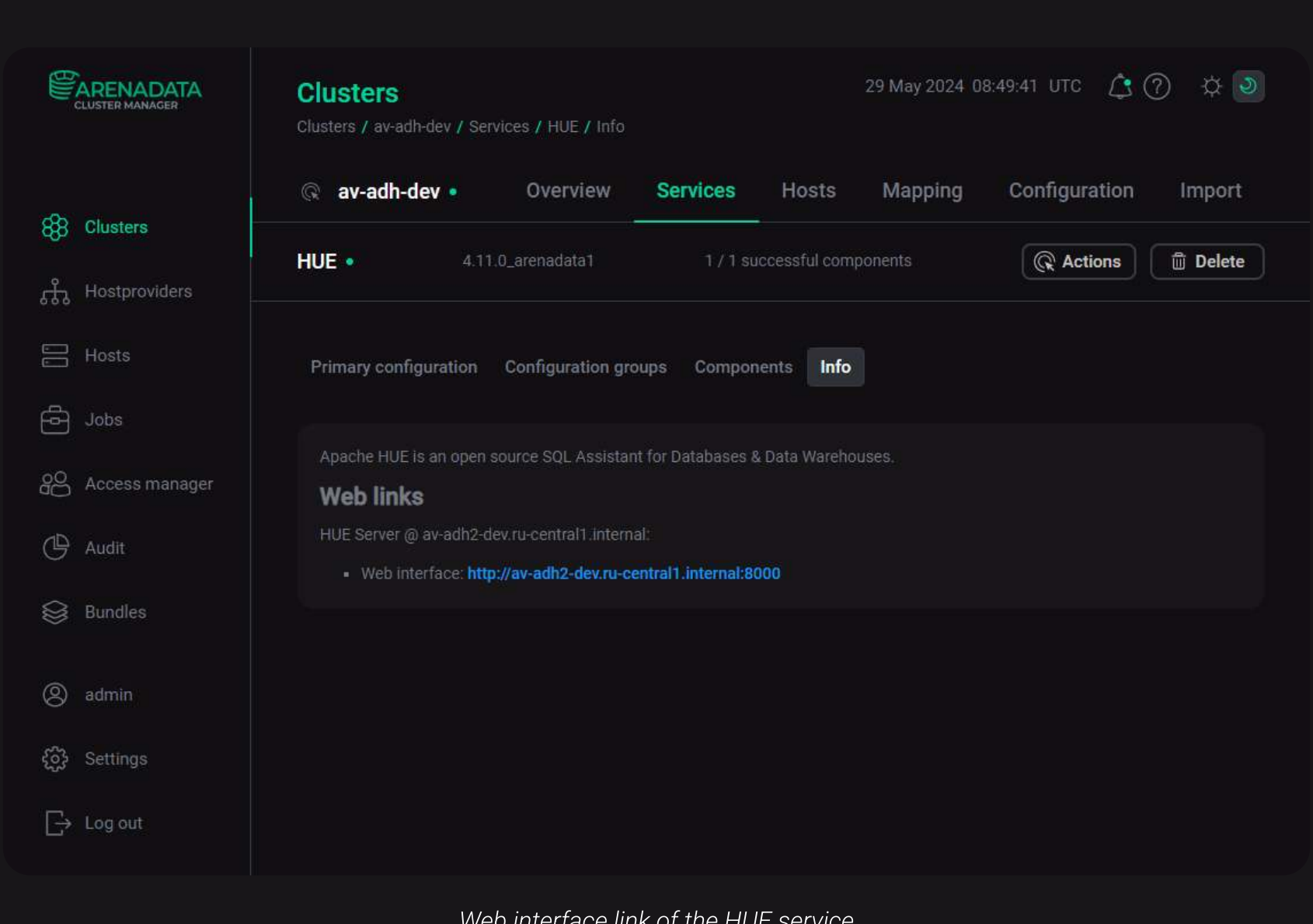
To Table of Contents

- Editor
- Documents
- Files
- Ozone
- Tables
  - Overview
  - Sample
  - Details
- Jobs
- Importer
  - Remote file import
  - External database import
  - Manual data import

The HUE service provides a web interface that allows you to negotiate with various databases and data warehouses in a convenient manner. In ADH distribution, it contains pre-set SQL interpreters for the following solutions:

- Hive
- Impala
- PySpark
- Trino
- Kyuubi (with pre-set Spark3)
- Flink 2

You can find the link to the HUE web interface on the ***Info*** tab of the HUE service in the ADCM UI.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / av-adh-dev / Services / HUE / Info
29 May 2024 08:49:41 UTC
av-adh-dev
Overview
Services
Hosts
Mapping
Configuration
Import
HUE
4.11.0_arenadata1
1 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Components
Info

Apache HUE is an open source SQL Assistant for Databases & Data Warehouses.

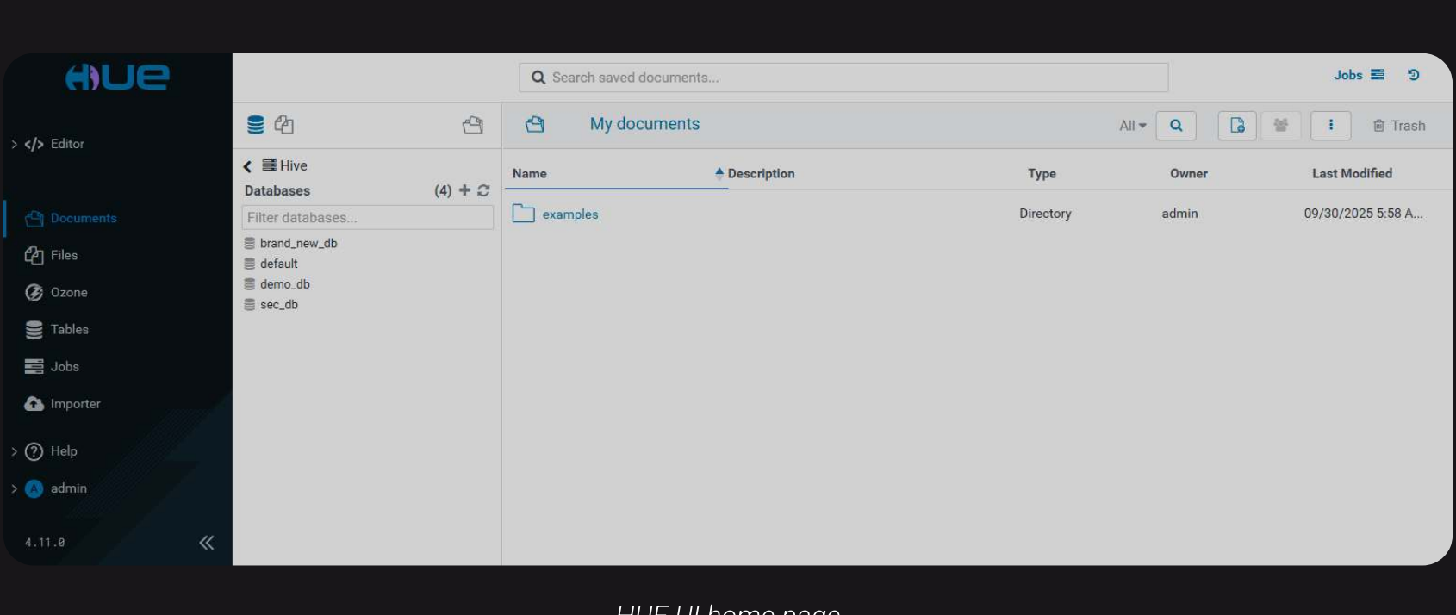
Web links

HUE Server @ av-adh2-dev.ru-central1.internal:

- Web interface: http://av-adh2-dev.ru-central1.internal:8000

Web interface link of the HUE service

HUE Server UI works on the `8000` port.


HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
Hive
Databases
Filter databases...
brand_new_db
default
demo_db
sec_db
My documents
Trash

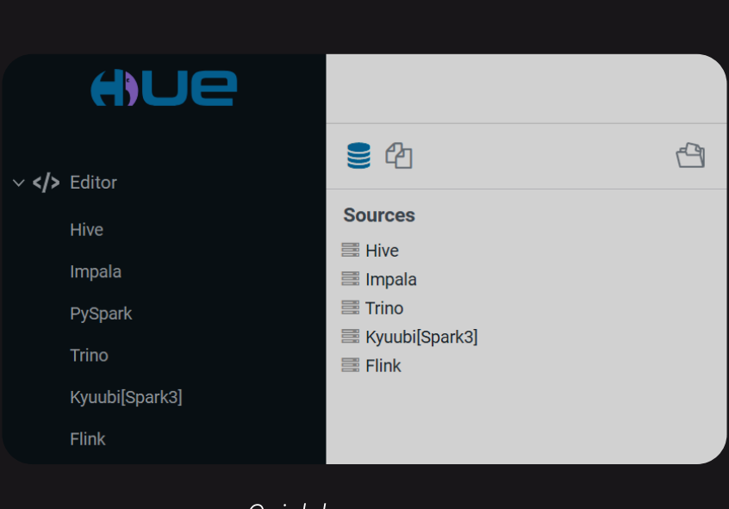
Name	Description	Type	Owner	Last Modified
examples		Directory	admin	09/30/2025 5:58 A...

HUE UI home page

The top bar of the home page contains the documents search field, jobs browser and preview buttons, and task history button.

Central part of the UI is occupied by the application area. Most of the work in HUE happens here.

Collapsible menu between the application area and the leftmost part of the UI is the quick browse menu. On the top of it, there is a switch between the database, files, and documents browsers. The main part contains links to various data, such as database elements, files, and documents, depending on the selected browser. You can open these data in the application area by clicking the links. You can also drag and drop the links to the application area to quickly paste the names of the database elements, file paths, etc.


HUE
Editor
Hive
Impala
PySpark
Trino
Kyuubi[Spark3]
Flink
Sources
Hive
Impala
Trino
Kyuubi[Spark3]
Flink

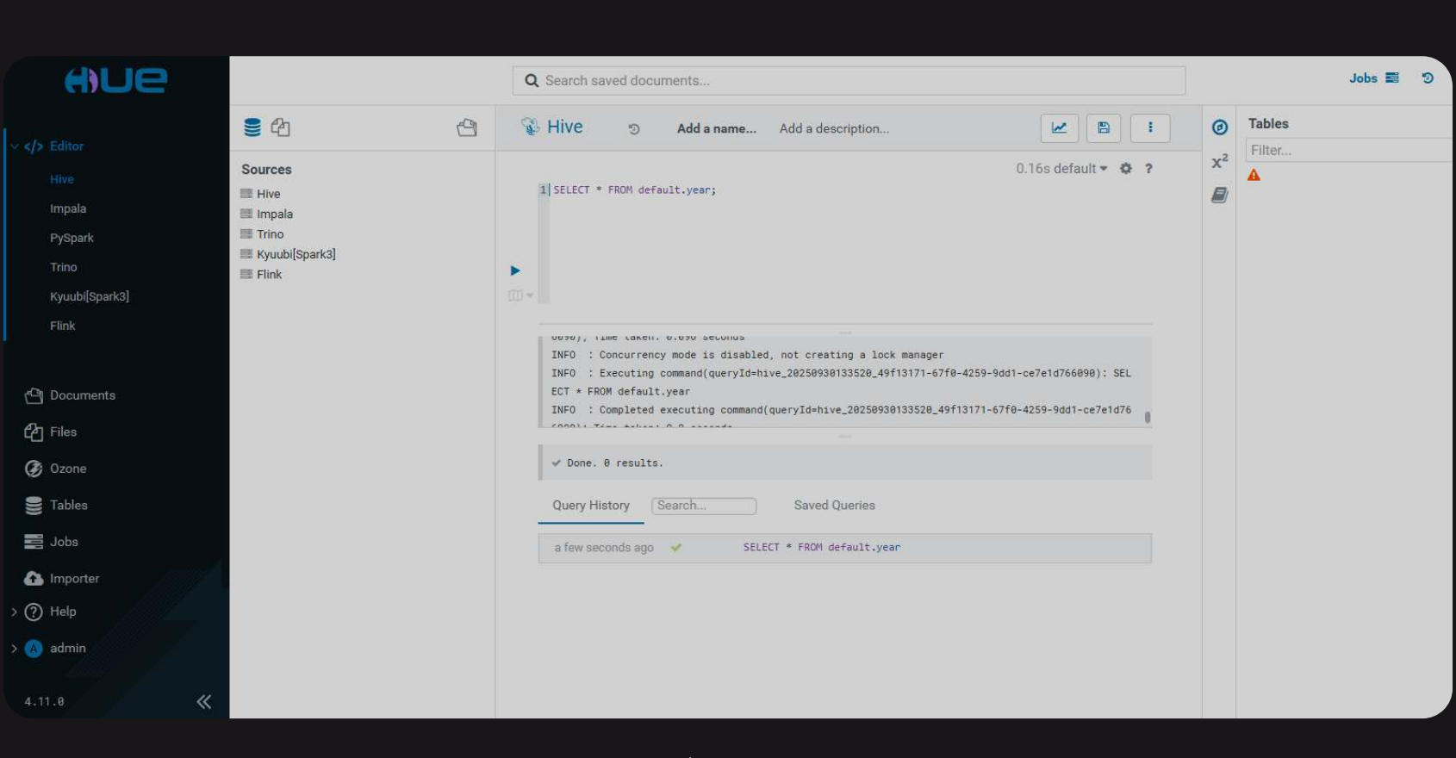
Quick browse menu

The leftmost collapsible menu contains the following pages:

- Editor
- Documents
- Files
- Ozone
- Tables
- Jobs
- Importer

Editor

The ***Editor*** page has the following view.


HUE
Search saved documents...
Jobs
Editor
Hive
Impala
PySpark
Trino
Kyuubi[Spark3]
Flink
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
Sources
Hive
Add a name...
Add a description...
Tables
Query History
Saved Queries

```
SELECT * FROM default.year;
```

```
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20250930133528_49f13171-67f0-4259-9dd1-ce7e1d766890): SELECT * FROM default.year
INFO  : Completed executing command(queryId=hive_20250930133528_49f13171-67f0-4259-9dd1-ce7e1d76
```

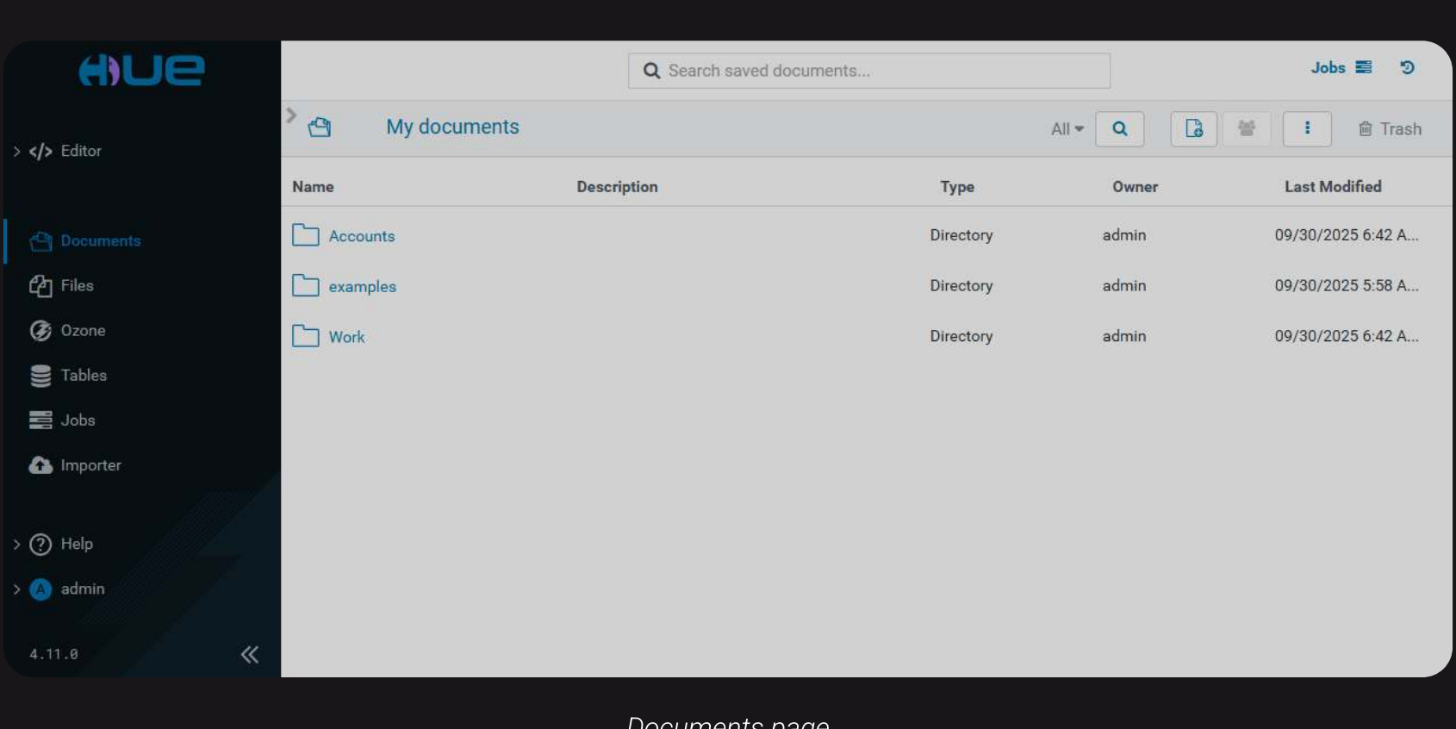
✓ Done. 0 results.

Editor page

The ***Editor*** page provides the application area which allows you to compose, execute, and explain SQL queries. Data sources include ***impala***, ***hive***, and ***kyuubi***. You can also use the notebook mode to create and store SQL snippets.

Documents

The ***Documents*** page has the following view.


HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
My documents
Trash

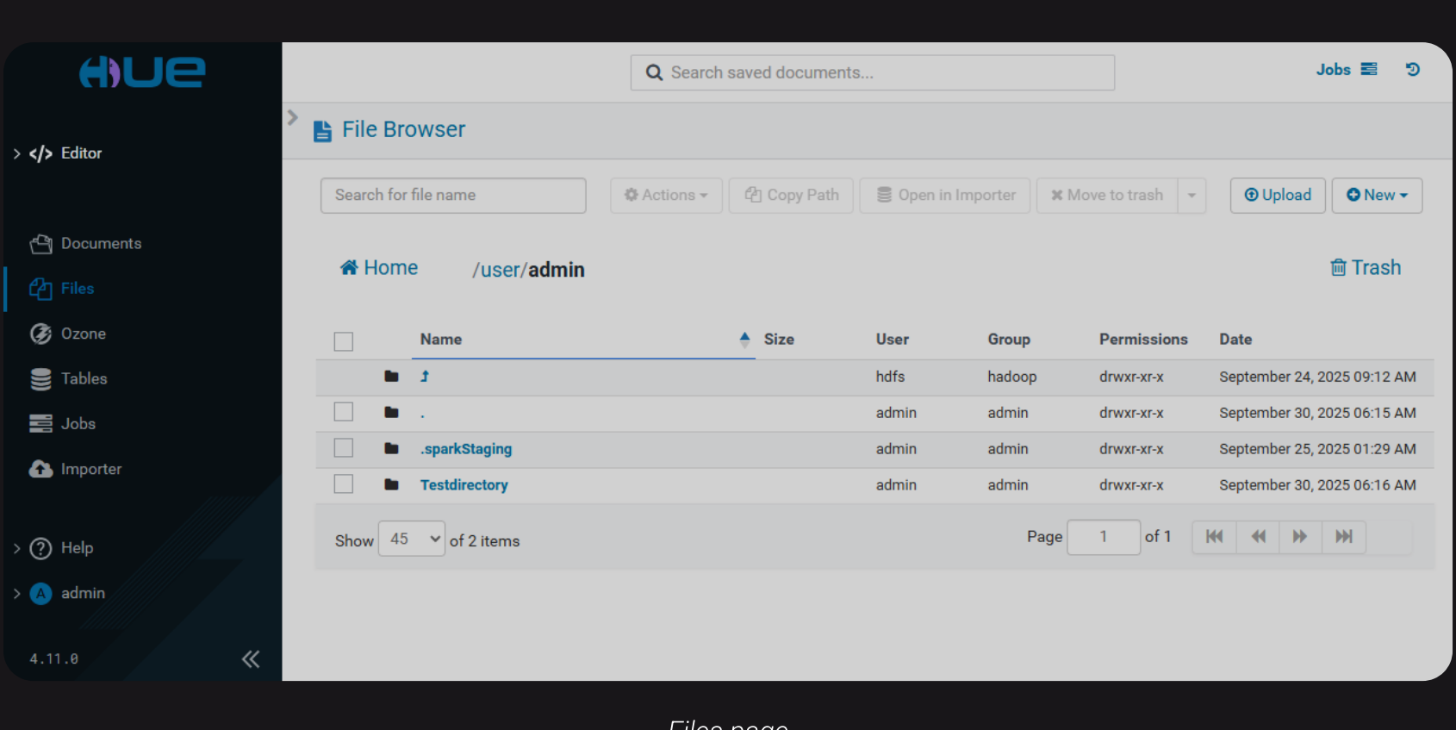
Name	Description	Type	Owner	Last Modified
Accounts		Directory	admin	09/30/2025 6:42 A...
examples		Directory	admin	09/30/2025 5:58 A...
Work		Directory	admin	09/30/2025 6:42 A...

Documents page

The ***Documents*** page contains your documents and folders. You can create, share, and manage your documents, and browse the trash folder using the corresponding buttons.

Files

The ***Files*** page has the following view.


HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
File Browser
Search for file name
Actions
Copy Path
Open in importer
Move to trash
Upload
New
Home
/user/admin
Trash

Name	Size	User	Group	Permissions	Date
⬆		hdfs	hadoop	drwxr-xr-x	September 24, 2025 09:12 AM
.		admin	admin	drwxr-xr-x	September 30, 2025 06:15 AM
.sparkStaging		admin	admin	drwxr-xr-x	September 25, 2025 01:29 AM
TestDirectory		admin	admin	drwxr-xr-x	September 30, 2025 06:16 AM

Show 45 of 2 items Page 1 of 1

Files page

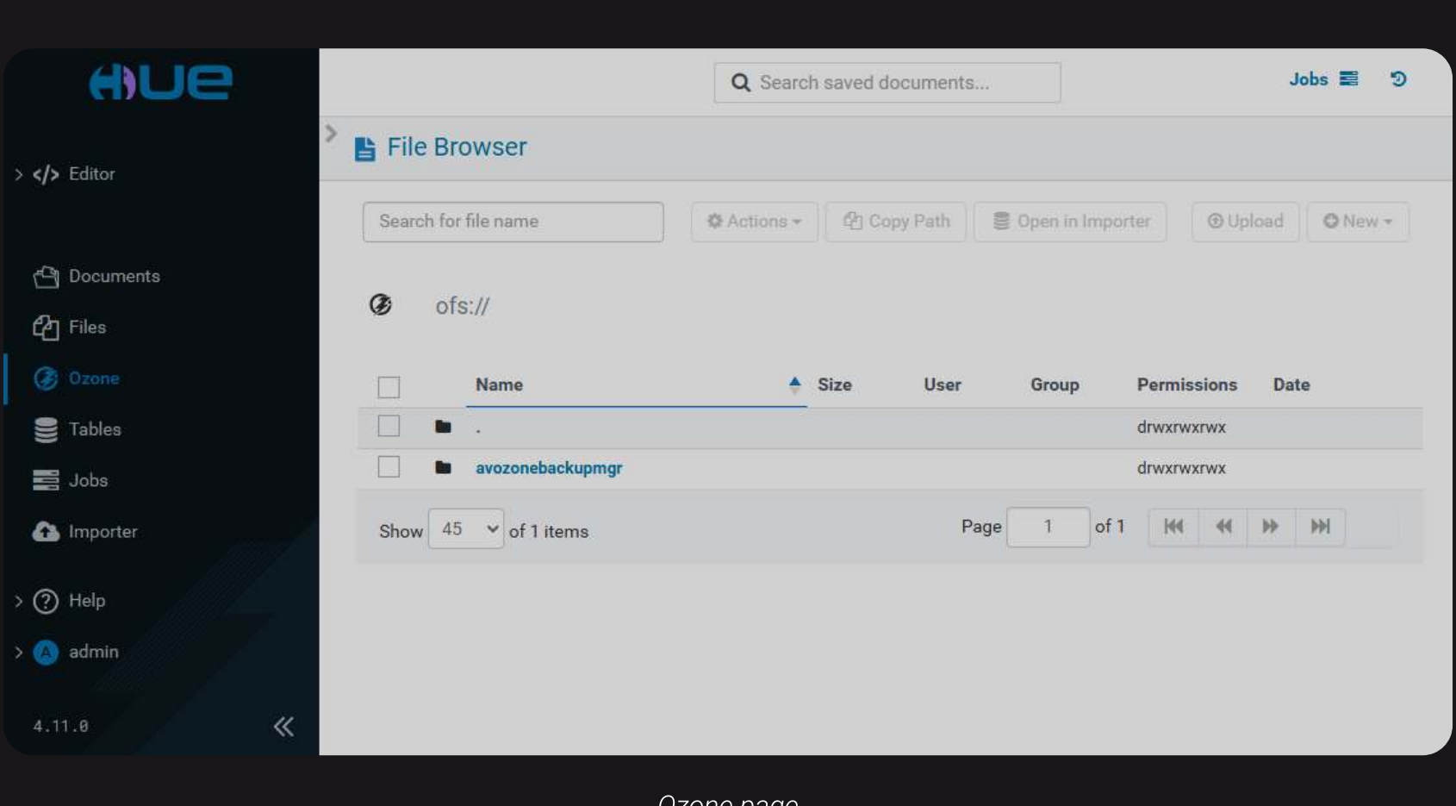
The ***Files*** page provides a browser for the Hadoop file system. You can search for files and folders or manage the existing ones. Possible actions include:

- rename;
- move;
- copy;
- download;
- change permissions;
- view summary;
- change the replication factor;
- move to the trash folder;
- delete permanently.

You can also create or upload new files on this page, copy a file path or open it on the ***Importer*** page.

Ozone

The ***Ozone*** page has the following view.


HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
File Browser
Search for file name
Actions
Copy Path
Open in importer
Upload
New
ofs://

Name	Size	User	Group	Permissions	Date
.				drwxrwxrwx	
avozonebackupmgr				drwxrwxrwx	

Show 45 of 1 items Page 1 of 1

Ozone page

The ***Ozone*** page provides a browser for the Ozone file system (OFS). You can search for files and folders or manage the existing ones. Possible actions include:

- rename;
- move;
- copy;
- download;
- change permissions;
- view summary;
- delete permanently.

You can also create or upload new files on this page, copy a file path or open it on the ***Importer*** page.

Tables

The ***Tables*** page has the following view.

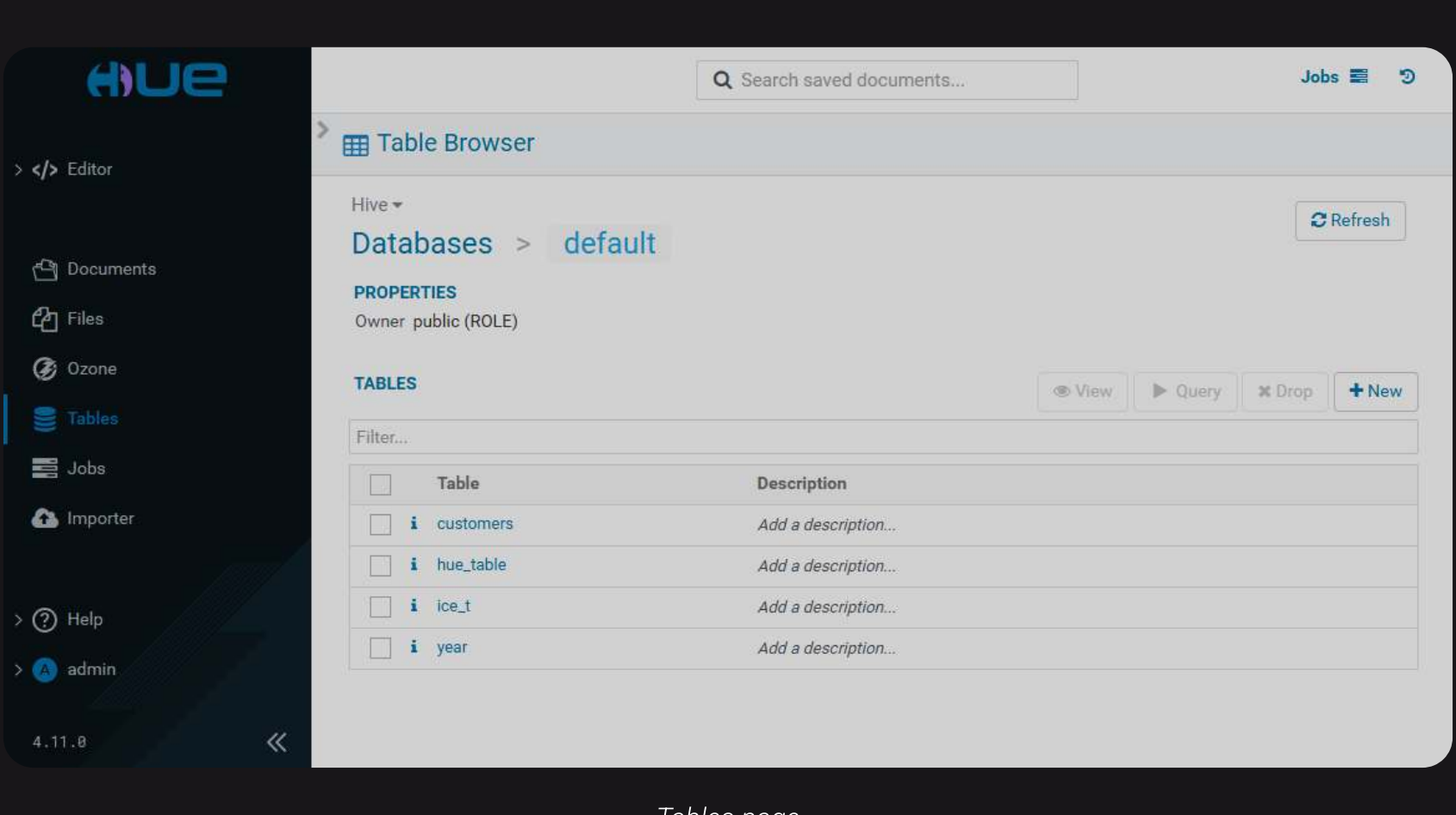

HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
Table Browser
Hive
Databases > default
Refresh
PROPERTIES
Owner public (ROLE)
TABLES
View
Query
Drop
New
Filter...

Table	Description
customers	Add a description...
hue_table	Add a description...
ice_t	Add a description...
year	Add a description...

Tables page

The ***Tables*** page allows you to choose the database source and browse through the tables of available databases. The following actions become available when a table is selected:

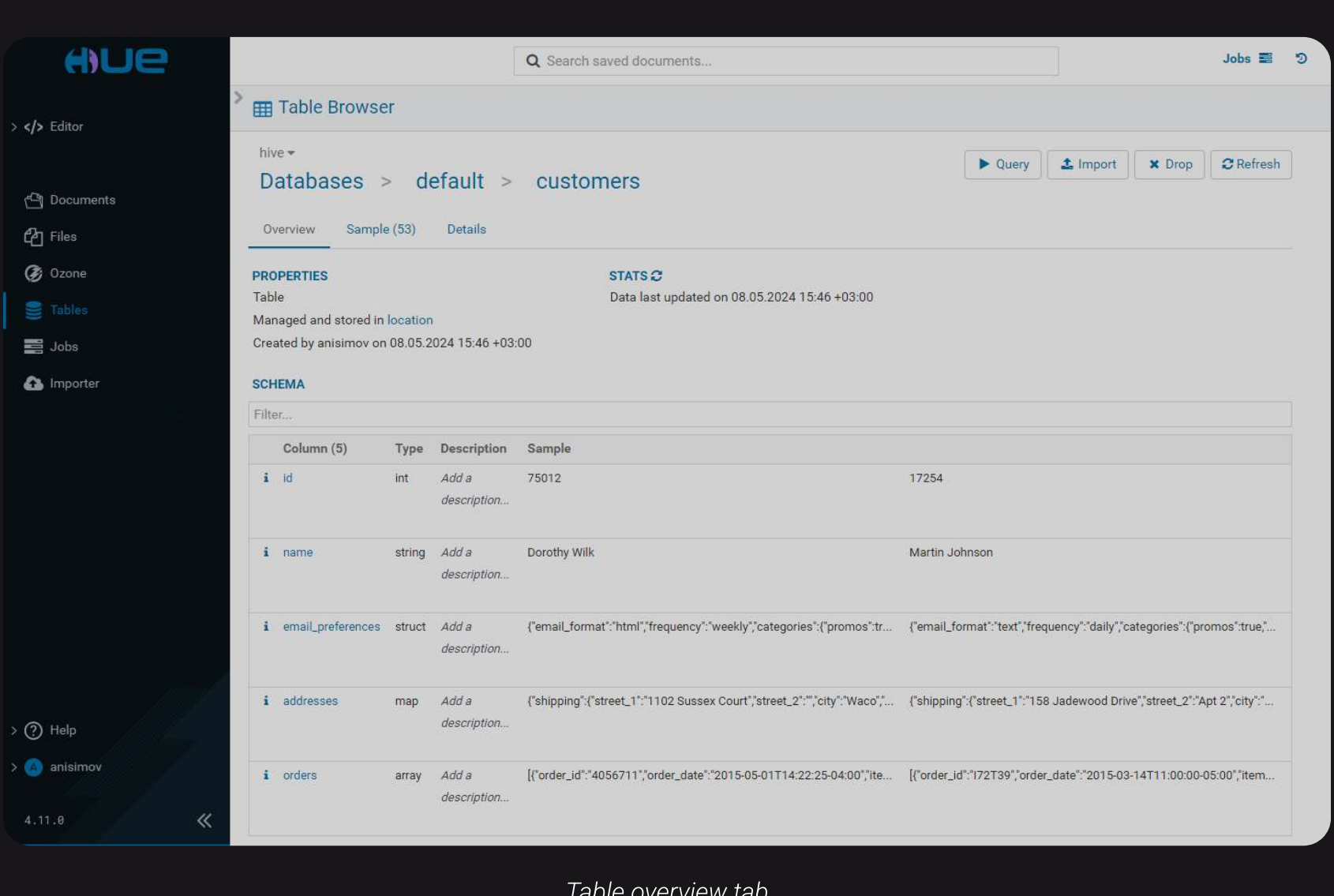
- ***Query*** — run the following query against the selected table:

```
SELECT * FROM <selected_db>.<selected_table> LIMIT 100;
```

- ***Import*** — import data from Hadoop filesystem. This action moves the data from its location into the selected table storage location. You can also choose to overwrite existing data during this action.
- ***Drop*** — drop the selected table.
- ***Refresh*** — refresh the displayed data in the selected table.

Overview

When you select a particular table, you can view the list of its columns along with their descriptions and sample values on the ***Overview*** tab.


HUE
Table Browser
hive
Databases > default > customers
Query
Import
Drop
Refresh
Overview
Sample (53)
Details
PROPERTIES
STATS
SCHEMA

Table

Managed and stored in location

Created by anisimov on 08.05.2024 15:46 +03:00

Data last updated on 08.05.2024 15:46 +03:00

Table overview tab

Sample

The ***Sample*** tab displays a collection of rows of the selected table, like shown below.

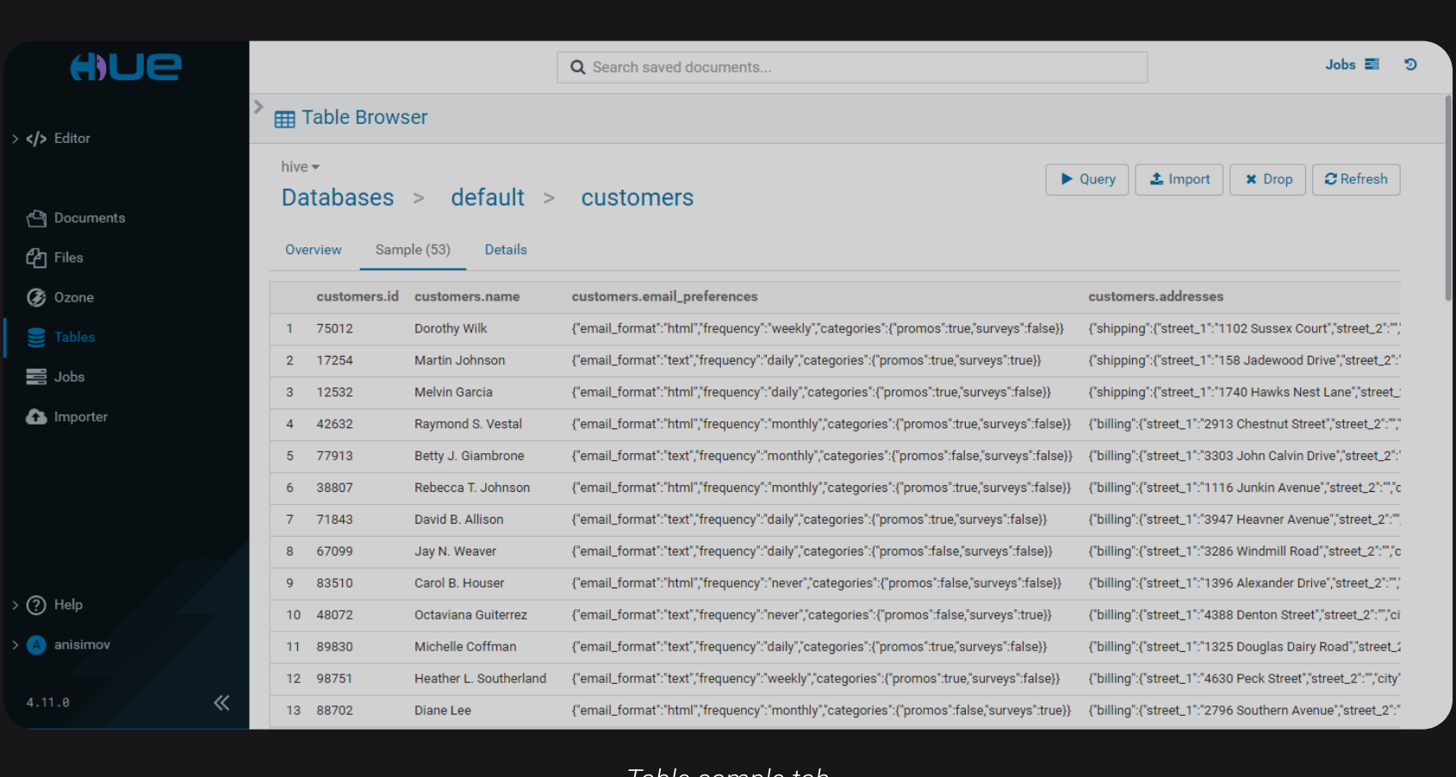
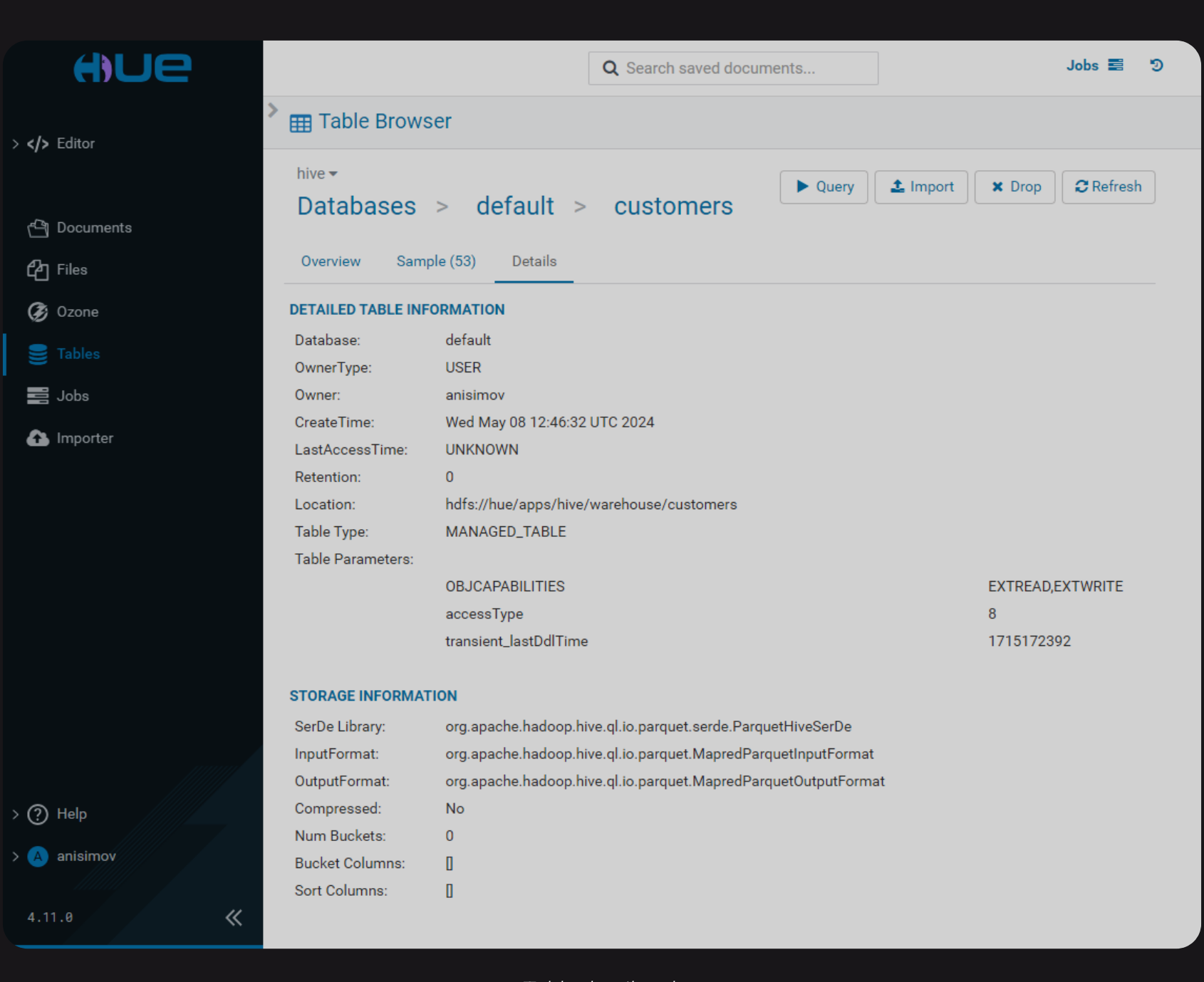

HUE
Table Browser
Databases > default > customers
Overview
Sample (53)
Details
customers.id
customers.name
customers.email_preferences
customers.addresses

Table sample tab

Details

The ***Details*** tab displays various details of the selected table, like shown below.


HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
anisimov
4.11.0
Table Browser
hive
Databases > default > customers
Query
Import
Drop
Refresh
Overview
Sample (53)
Details

DETAILED TABLE INFORMATION

Database:	default	
OwnerType:	USER	
Owner:	anisimov	
CreateTime:	Wed May 08 12:46:32 UTC 2024	
LastAccessTime:	UNKNOWN	
Retention:	0	
Location:	hdfs://hue/apps/hive/warehouse/customers	
Table Type:	MANAGED_TABLE	
Table Parameters:		
	OBJCAPABILITIES	EXTREAD,EXTWRITE
	accessType	8
	transient_lastDdlTime	1715172392

STORAGE INFORMATION

SerDe Library:	org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe
InputFormat:	org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat
OutputFormat:	org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat
Compressed:	No
Num Buckets:	0
Bucket Columns:	[]
Sort Columns:	[]

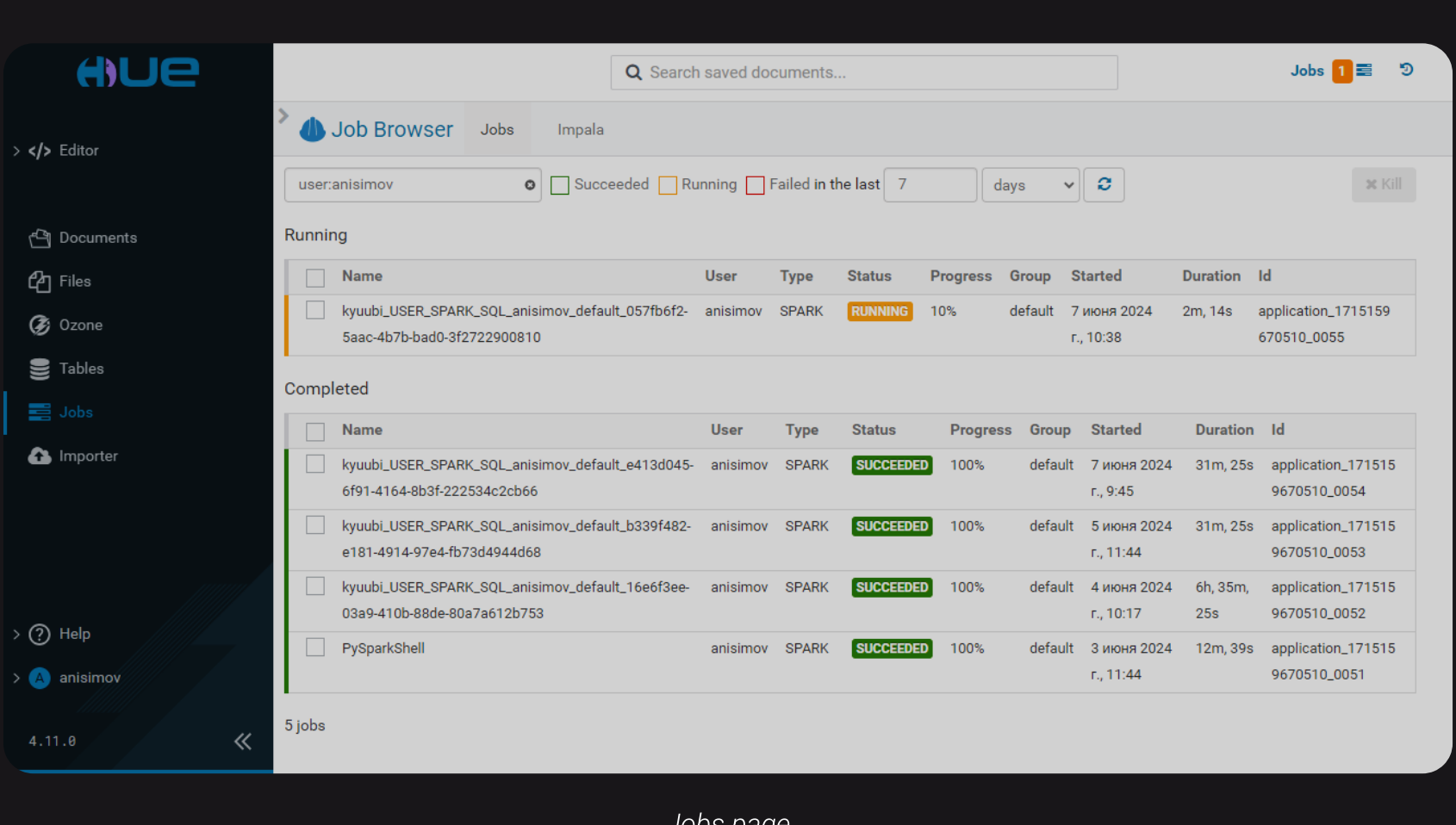
Table details tab

Jobs

The ***Jobs*** page displays the list of all running and completed jobs. You can view the YARN jobs and Impala queries history on the corresponding tabs.

> **NOTE**
>
> The tabs only appear if all of the corresponding service components are present in the ADH cluster.

You can filter the jobs list by checking the boxes ***Succeeded***, ***Running***, or ***Failed***. You can also filter the list by the time period since the task start.

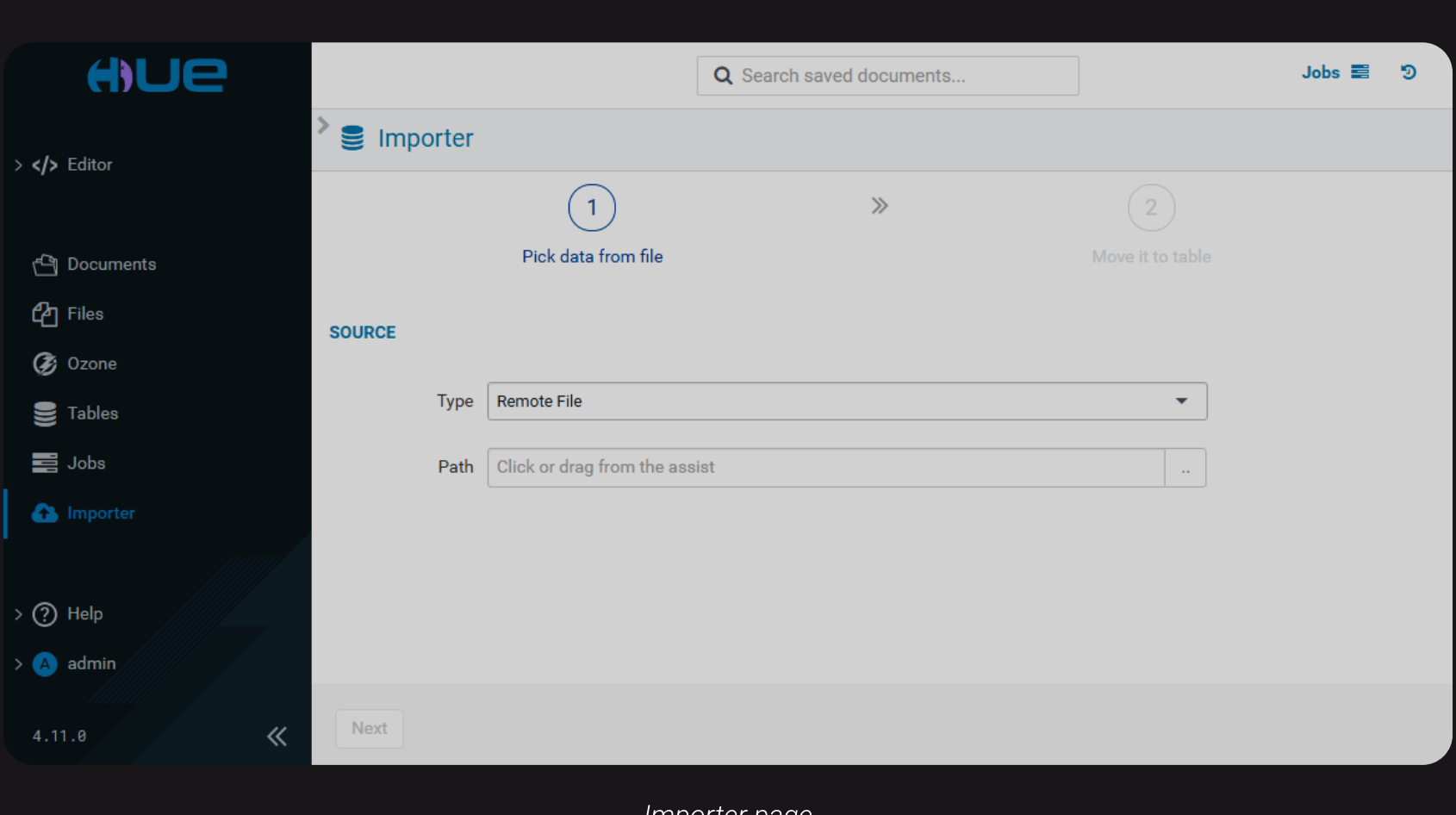

HUE
Job Browser
Jobs
Impala
user:anisimov
Succeeded
Running
Failed in the last
7
days
Running
Completed
5 jobs

Jobs page

To suspend, resume, or stop a job, select its box and click ***Suspend***, ***Resume***, or ***Kill***, respectively. You can perform these actions on any number of jobs in this manner.

Importer

The ***Importer*** page allows you to create manually or import existing folders, tables, and databases.


HUE
Search saved documents...
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
admin
4.11.0
Importer
1
Pick data from file
2
Move it to table
SOURCE
Next

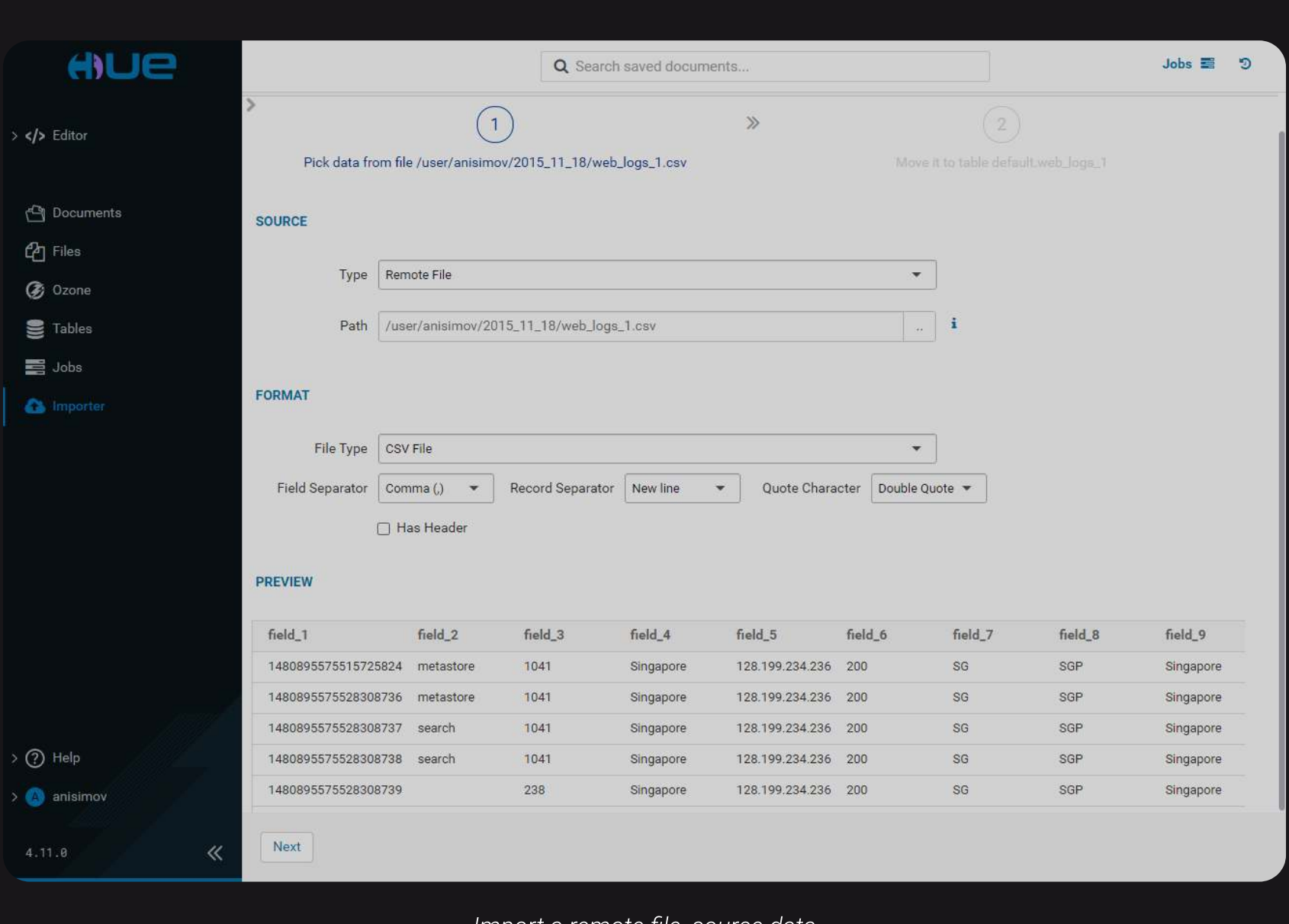
Type: Remote File

Path: Click or drag from the assist

Importer page

Remote file import

If you want to upload a file-stored table, select the `Remote File` type and specify the path to that file. Correct the values in the ***Format*** section if necessary. Click ***Next***.


HUE
Search saved documents...
Jobs
1
Pick data from file /user/anisimov/2015_11_18/web_logs_1.csv
2
Move it to table default.web_logs_1
SOURCE
FORMAT
PREVIEW
Next

Type: Remote File

Path: /user/anisimov/2015_11_18/web_logs_1.csv

File Type: CSV File

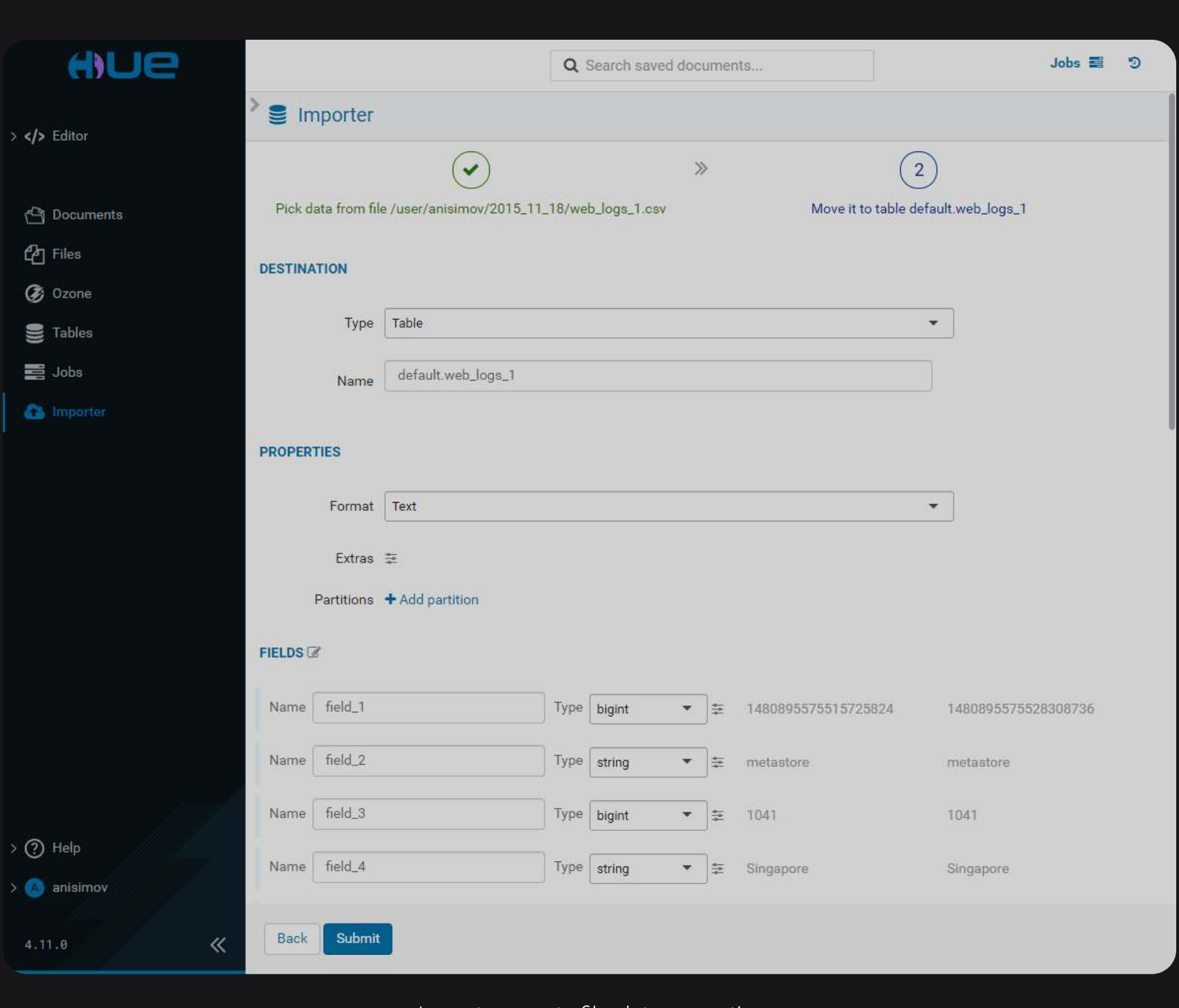
Field Separator: Comma (,) Record Separator: New line Quote Character: Double Quote

☐ Has Header

field_1	field_2	field_3	field_4	field_5	field_6	field_7	field_8	field_9
1480895575515725824	metastore	1041	Singapore	128.199.234.236	200	SG	SGP	Singapore
1480895575528308736	metastore	1041	Singapore	128.199.234.236	200	SG	SGP	Singapore
1480895575528308737	search	1041	Singapore	128.199.234.236	200	SG	SGP	Singapore
1480895575528308738	search	1041	Singapore	128.199.234.236	200	SG	SGP	Singapore
1480895575528308739		238	Singapore	128.199.234.236	200	SG	SGP	Singapore

Import a remote file, source data

On the next page, specify the data type (table or search index), data properties and field details, and click ***Submit***.


HUE
Search saved documents...
Jobs
Importer
Pick data from file /user/anisimov/2015_11_18/web_logs_1.csv
2
Move it to table default.web_logs_1
DESTINATION
PROPERTIES
FIELDS
Back
Submit

Type: Table

Name: default.web_logs_1

Format: Text

Extras

Partitions + Add partition

Name	Type		
field_1	bigint	1480895575515725824	1480895575528308736
field_2	string	metastore	metastore
field_3	bigint	1041	1041
field_4	string	Singapore	Singapore

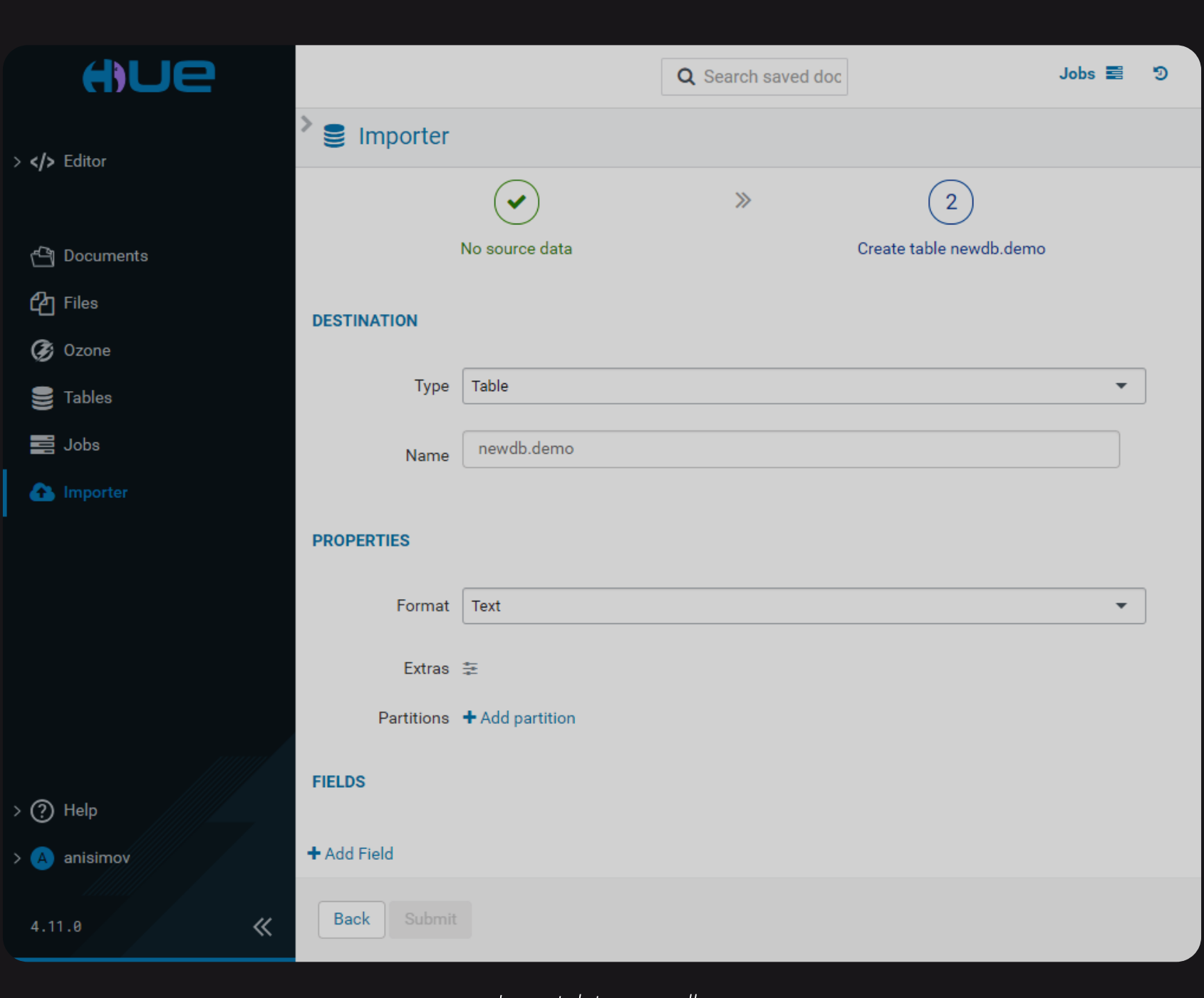
Import a remote file, data properties

External database import

If you want to upload an external database, select the `External Databse` type and specify the connection details. Click ***Next***. On the following page, specify the import details and click ***Submit***.

Manual data import

If you want to upload data manually, select the `Manually` type, and click ***Next***. On the following screen, specify the data type (`table`, `database`, or `folder`), data properties and field details if necessary, and click ***Submit***.


HUE
Search saved doc
Jobs
Editor
Documents
Files
Ozone
Tables
Jobs
Importer
Help
anisimov
4.11.0
Importer
No source data
2
Create table newdb.demo
DESTINATION
PROPERTIES
FIELDS
+ Add Field
Back
Submit

Type: Table

Name: newdb.demo

Format: Text

Extras

Partitions + Add partition

Import data manually

Add and use an interpreter in HUE

Vladimir Adamenkov

Contents

To Table of Contents

Connectors used by HUE

Connect to ADB (Greenplum)

Connectors used by HUE

HUE in ADH has pre-set connectors for the Hive, Impala, and Kyuubi[Spark3] data sources. Many more can be added via the ***Custom hue.ini*** section available at the HUE service configuration page; the full list can be found in the HUE documentation.

NOTE

Pre-set connectors only appear in the HUE UI if corresponding service components are present in the ADH cluster:

- Hive — Hive service with all components;
- Impala — Impala service with all components;
- PySpark — Spark3 Livy Server component on the same host;
- Trino — Trino service with all components, also requires SSL if Kerberos authentication is enabled;
- Kyuubi[Spark3] — Kyuubi Server and Spark3 Client components on the same host;
- Flink — Flink2 SQL-Gateway component on any host.

After adding necessary services and/or components, you need to restart the HUE service for the changes to take effect.

IMPORTANT

The Trino interpreter requires the usage of SSL if Kerberos authentication is enabled in the cluster.

To configure a connector, go to ADCM UI, select your ADH cluster, and navigate to ***Services*** → ***HUE*** → ***Components*** → ***HUE Server***. Select the required interpreter section and adjust the parameters as needed. Refer to HUE configuration parameters reference for assistance.

Connect to ADB (Greenplum)

This is an example of how to add and configure a connector for the ADB database, which is based on Greenplum.

To be able to connect HUE to ADB database instance make sure that it expects connection on a certain network interface with certain credentials. Go to the ADCM UI and select your ADB cluster, then navigate to ***Services*** → ***ADB*** → ***Primary configuration***. Open the ***Custom pg_hba section*** parameter and add a string of the following kind to it:

```
host <dbname> <user> <IP-address>/<mask> trust
```

where:

- `<dbname>` — name of the database being connected to.
- `<user>` — username of that database.
- `<IP-address>/<mask>` — address of the ADH host where HUE Server is installed.

Example:

```
host adb admin 10.10.10.21/0 trust
```

Save the configuration, then click ***Actions*** and select ***Reconfigure***.

Next, you need to add connection parameters to the HUE configuration. Select your ADH cluster in ADCM UI, and navigate to ***Services*** → ***HUE*** → ***Components*** → ***HUE Server***. In the ***Primary configuration*** tab that appears, click ***Show advanced***. Open the ***Custom hue.ini*** section and add three parameters by clicking ***Add property***:

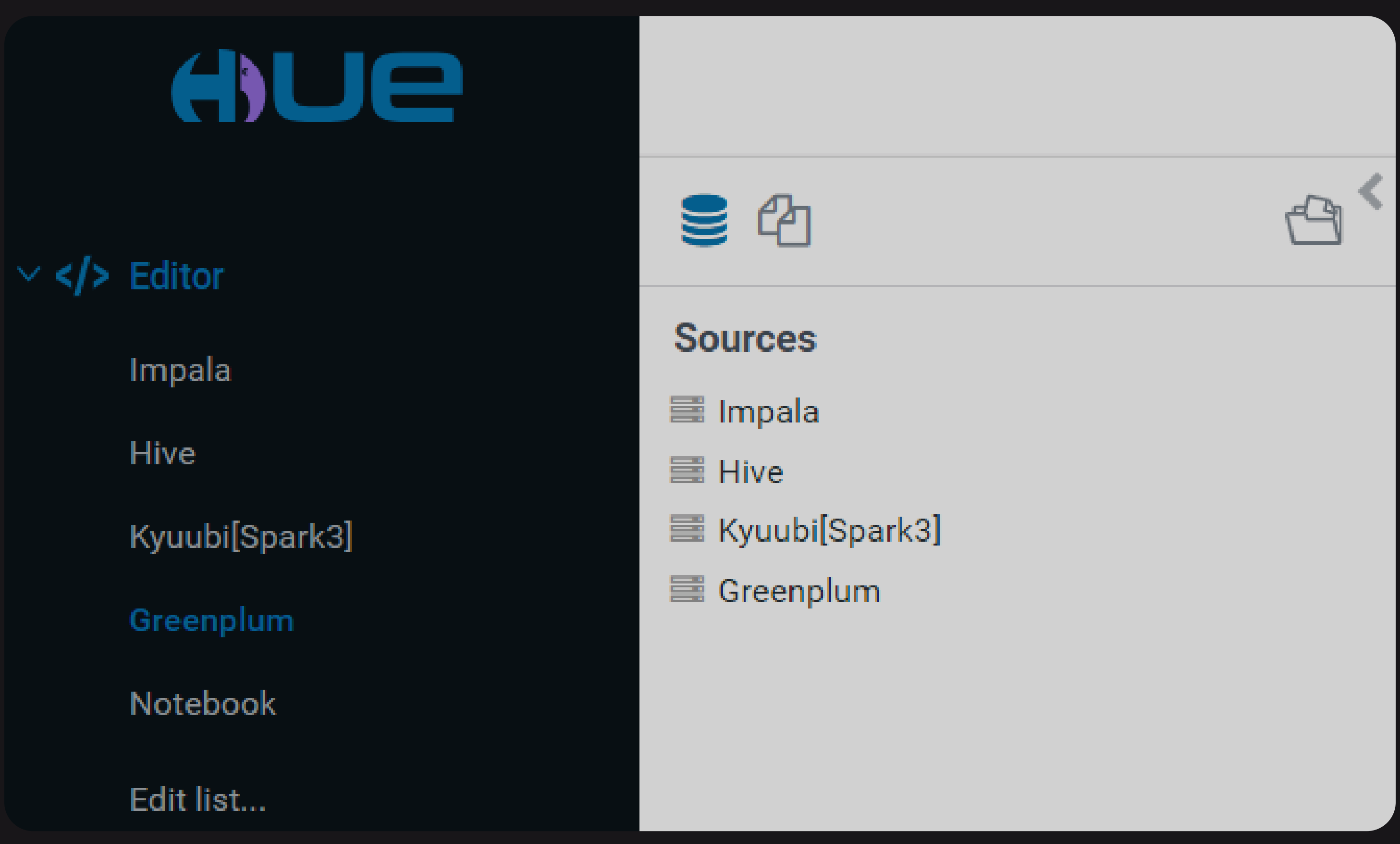
- `notebook.interpreters.greenplum.name` = `Greenplum`
- `notebook.interpreters.greenplum.interface` = `sqlalchemy`
- `notebook.interpreters.greenplum.options` = `'{"url": "postgresql+psycopg2://<user>:<password>@<dbaddress>:<port>/<dbname>"}'`

where:

- `<password>` — database user password.
- `<dbaddress>` — address of the database host awaiting connection (ADB master).
- `<port>` — database listen port.

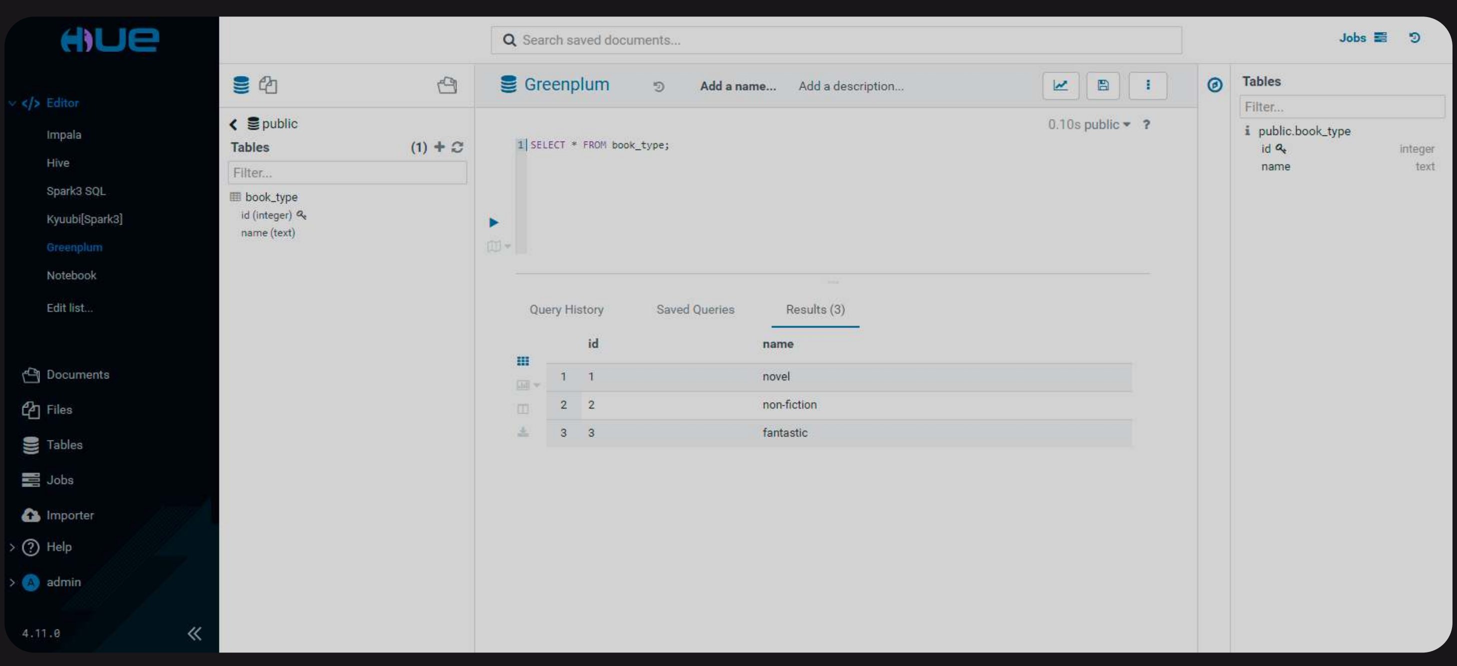
Standard Greenplum port in ADB is 5432. Example for the `options` parameter: `'{"url": "postgresql+psycopg2://admin:Secure_pass35@10.10.10.10:5432/adb"}'`.

Save the configuration and click ***Restart HUE Server*** in the ***Actions*** menu. Wait for the restart to complete and go to the HUE UI. The ***Greenplum*** data source should appear in the quick browse menu and in the ***Editor*** list.



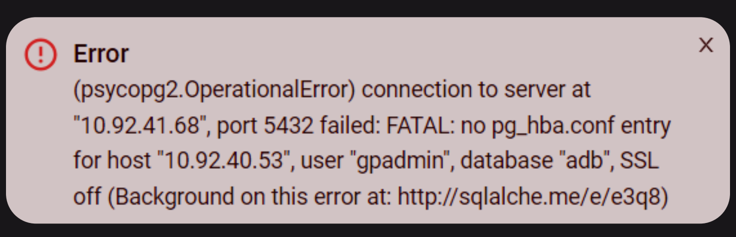
Greenplum data source

You can perform queries against the selected table. Type in the query in the application area and click ***Execute*** (or press `Ctrl+Enter`).



Query example

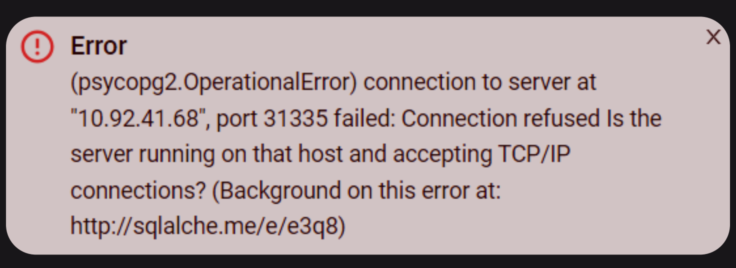
If ADB parameter ***Custom pg_hba section*** wasn't configured correctly, or invalid credentials were specified on the HUE Server configuration side, the connection will fail and an error message will appear.



Error
(psycopg2.OperationalError) connection to server at "10.92.41.68", port 5432 failed: FATAL: no pg_hba.conf entry for host "10.92.40.53", user "gpadmin", database "adb", SSL off (Background on this error at: http://sqlalche.me/e/e3q8)

Invalid pg_hba.conf configuration error

If a wrong port was specified on the HUE Server configuration side, the connection will fail and an error message will appear.



Error
(psycopg2.OperationalError) connection to server at "10.92.41.68", port 31335 failed: Connection refused Is the server running on that host and accepting TCP/IP connections? (Background on this error at: http://sqlalche.me/e/e3q8)

Connection refused error

Work with Iceberg tables in HUE

Vladimir Adamenkov

Contents

To Table of Contents

Overview

Add an Iceberg table

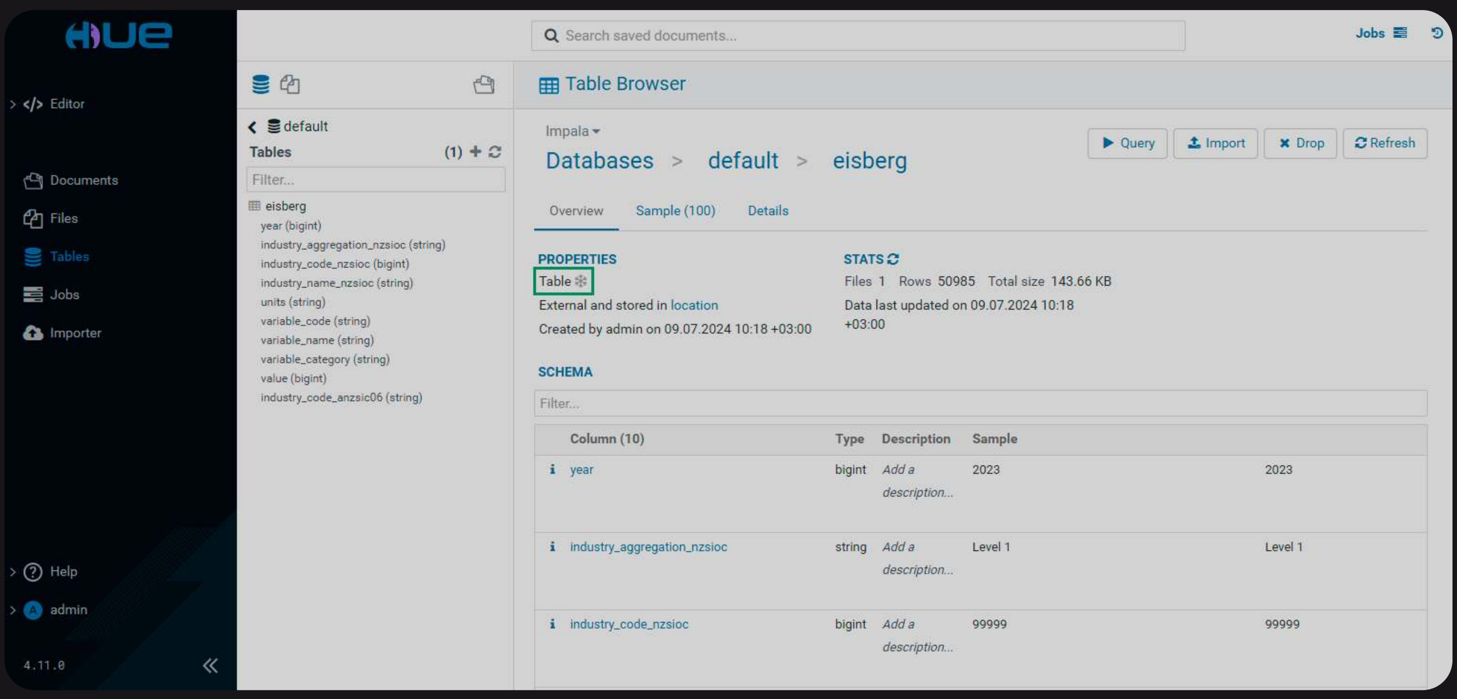
Overview

Apache Iceberg is an open, high-performance format for large analytic tables. HUE can recognize Iceberg tables and convert remote file tables to Iceberg format.

NOTE

Iceberg tables are supported in ADH version 3.2.4.3 and later.

When opened in HUE table browser, Iceberg tables are marked with ❄ icon on the ***Overview*** tab under the ***Properties*** header.



Editor
Documents
Files
Tables
Jobs
Importer
Help
admin
4.11.0
Search saved documents...
Jobs
Table Browser
Impala
Query
Import
Drop
Refresh
Overview
Sample (100)
Details

default

Tables (1)

eisberg
- year (bigint)
- industry_aggregation_nzsioc (string)
- industry_code_nzsioc (bigint)
- industry_name_nzsioc (string)
- units (string)
- variable_code (string)
- variable_name (string)
- variable_category (string)
- value (bigint)
- industry_code_anzsic06 (string)

Databases > default > eisberg

PROPERTIES

Table ❄

External and stored in location

Created by admin on 09.07.2024 10:18 +03:00

STATS

Files 1 Rows 50985 Total size 143.66 KB

Data last updated on 09.07.2024 10:18 +03:00

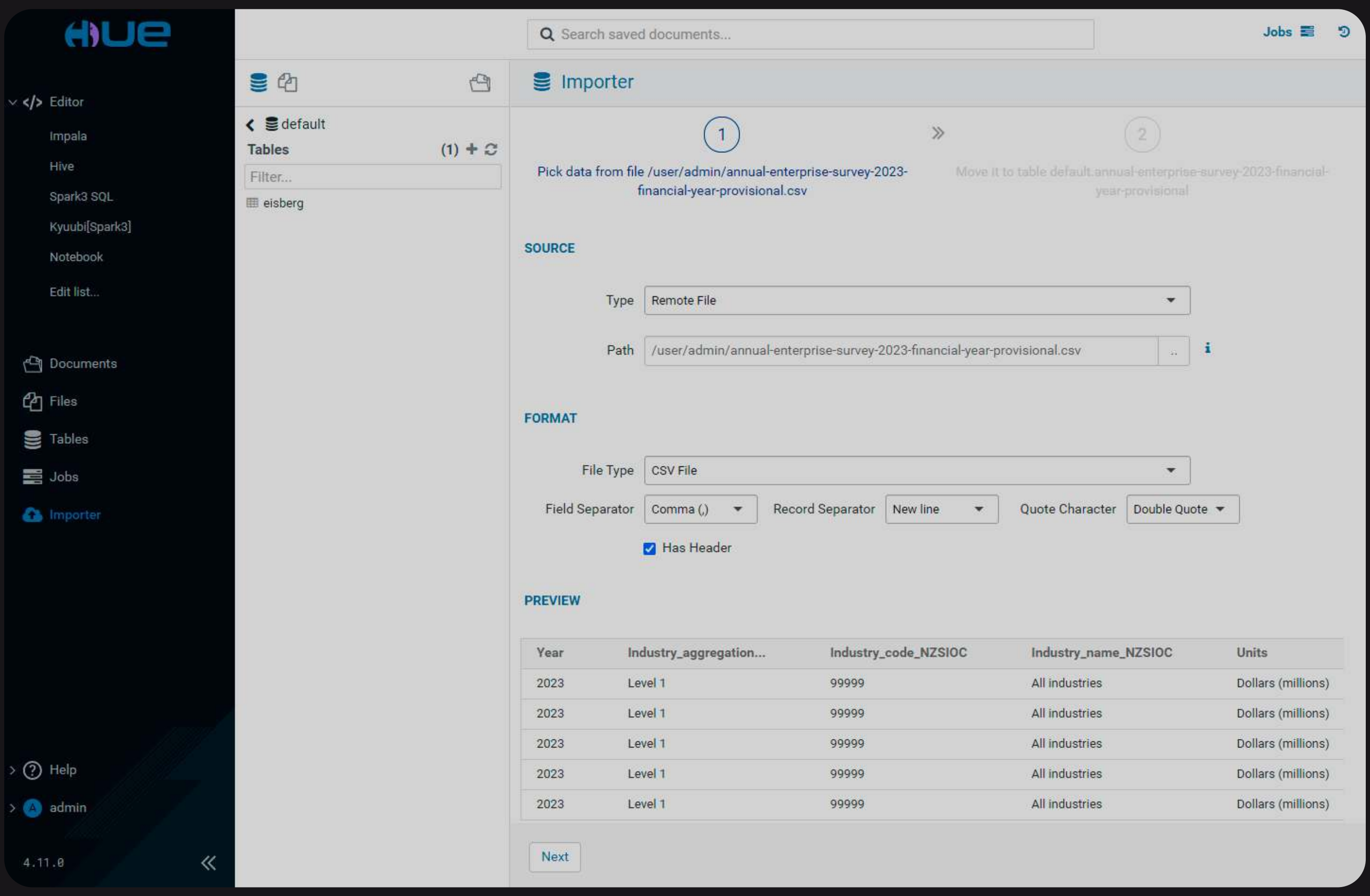
SCHEMA

Column (10)	Type	Description	Sample	
year	bigint	Add a description...	2023	2023
industry_aggregation_nzsioc	string	Add a description...	Level 1	Level 1
industry_code_nzsioc	bigint	Add a description...	99999	99999

Iceberg table label

Add an Iceberg table

You can add Iceberg tables by importing remote file tables and converting them. This can be done on the ***Importer*** page or by clicking ➕ icon in the quick browse menu (provided a database is selected). Select the `Remote File` type and specify the path to the table file. Adjust the values that appear under the ***Format*** header if necessary, and click ***Next***.



Editor
Impala
Hive
Spark3 SQL
Kyuubi[Spark3]
Notebook
Edit list...
Documents
Files
Tables
Jobs
Importer
Help
admin
4.11.0
default
Tables (1)
Filter...
eisberg
Search saved documents...
Jobs
Importer
1
2
Next

Pick data from file /user/admin/annual-enterprise-survey-2023-financial-year-provisional.csv

Move it to table default.annual-enterprise-survey-2023-financial-year-provisional

SOURCE

Type: Remote File

Path: /user/admin/annual-enterprise-survey-2023-financial-year-provisional.csv

FORMAT

File Type: CSV File

Field Separator: Comma (,) Record Separator: New line Quote Character: Double Quote

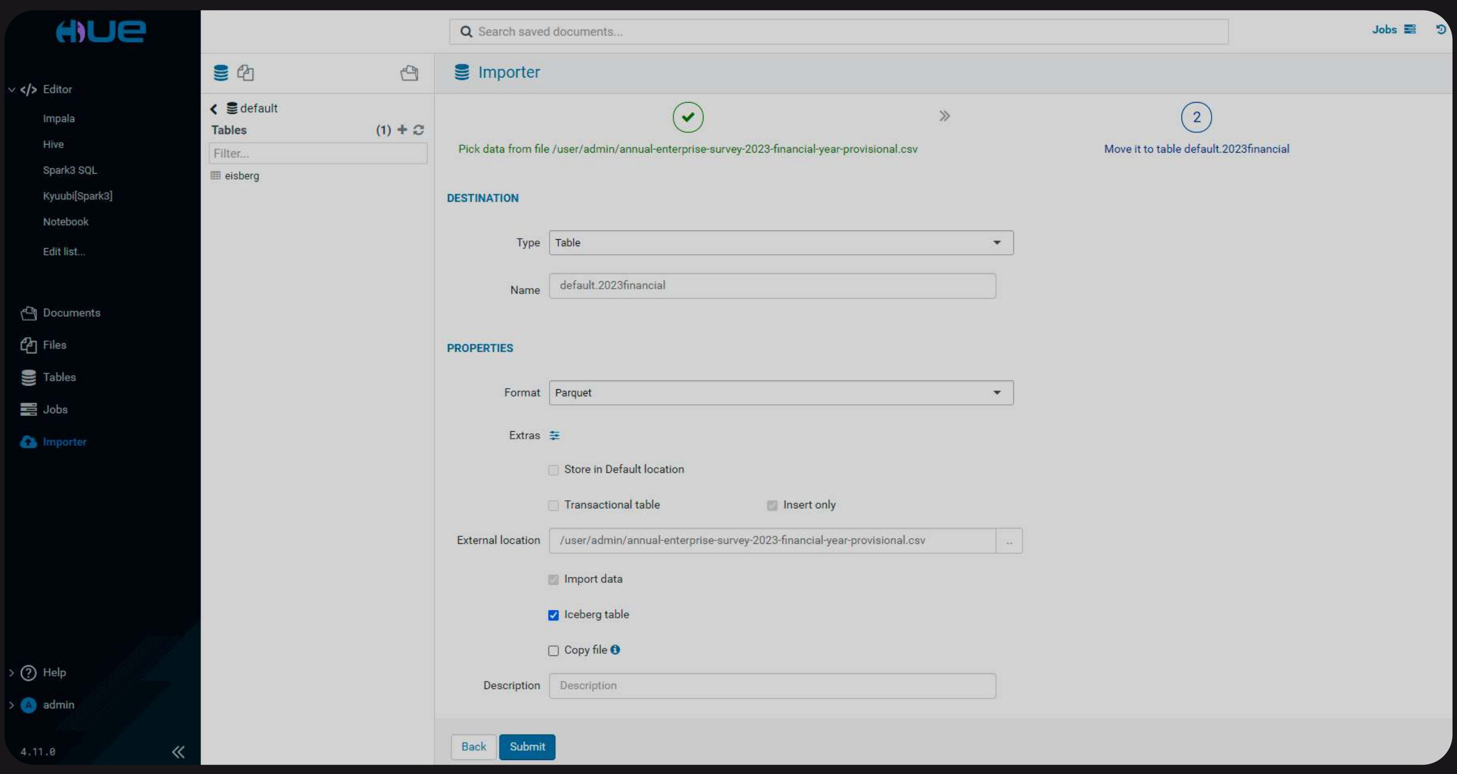
☑ Has Header

PREVIEW

Year	Industry_aggregation...	Industry_code_NZSIOC	Industry_name_NZSIOC	Units
2023	Level 1	99999	All industries	Dollars (millions)
2023	Level 1	99999	All industries	Dollars (millions)
2023	Level 1	99999	All industries	Dollars (millions)
2023	Level 1	99999	All industries	Dollars (millions)
2023	Level 1	99999	All industries	Dollars (millions)

Remote table file import

On the next screen, click ***Extras***, and check the ***Iceberg table*** box. Review and adjust, if necessary, other parameters, fields names and types. Click ***Submit***.



Editor
Impala
Hive
Spark3 SQL
Kyuubi[Spark3]
Notebook
Edit list...
Documents
Files
Tables
Jobs
Importer
Help
admin
4.11.0
default
Tables (1)
Filter...
eisberg
Search saved documents...
Jobs
Importer
2
Back
Submit

Pick data from file /user/admin/annual-enterprise-survey-2023-financial-year-provisional.csv

Move it to table default.2023financial

DESTINATION

Type: Table

Name: default.2023financial

PROPERTIES

Format: Parquet

Extras

☐ Store in Default location

☐ Transactional table ☑ Insert only

External location: /user/admin/annual-enterprise-survey-2023-financial-year-provisional.csv

☑ Import data

☑ Iceberg table

☐ Copy file

Description: Description

Conversion to Iceberg format details

After successful import, the new Iceberg table appears in the quick browse menu under the corresponding database title.

SPNEGO authentication in HUE

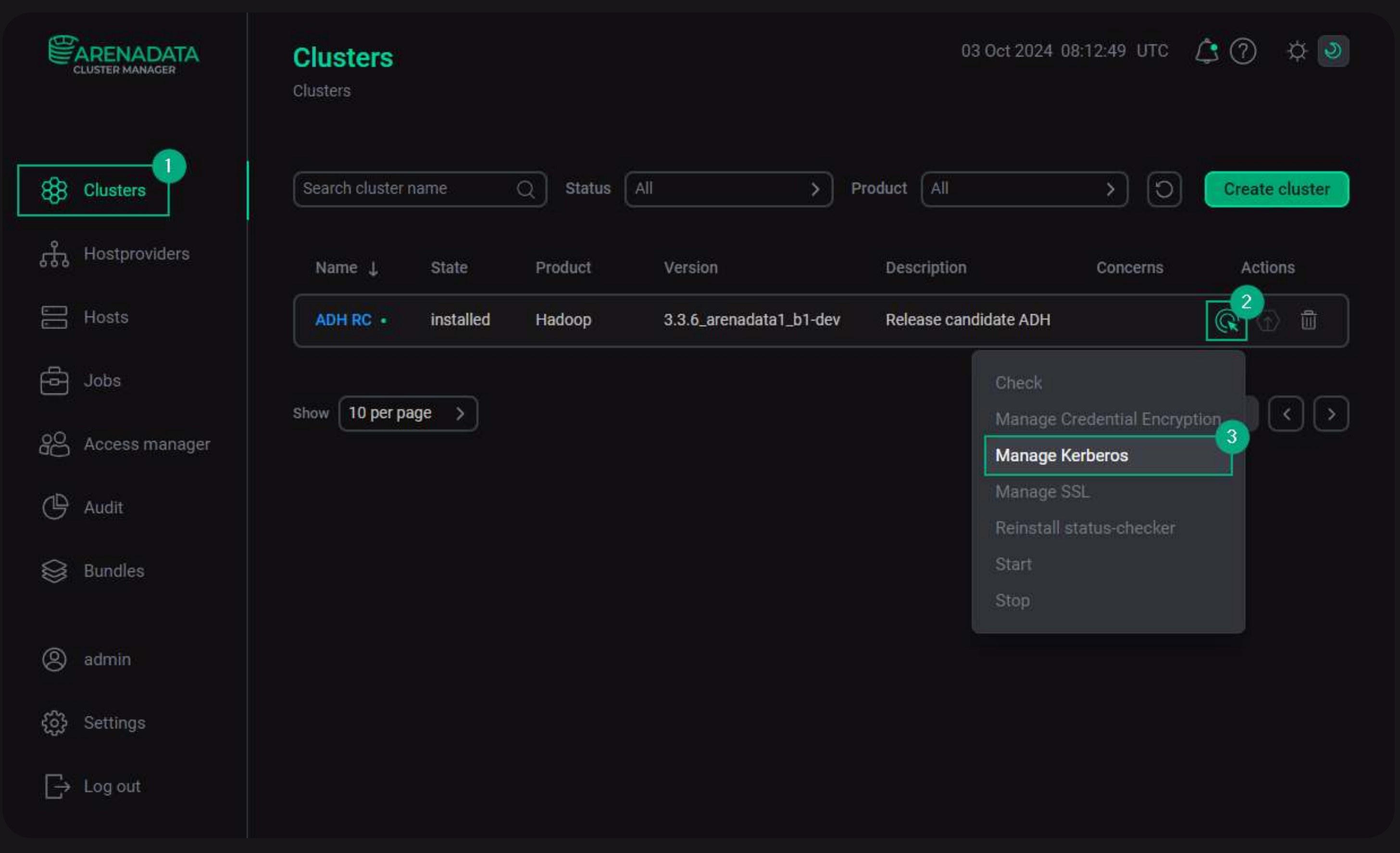
Vladimir Adamenkov

To Table of Contents

You can set up the SPNEGO authentication for the HUE UI, which allows accessing it securely and remotely. This authentication type is compatible with the LDAP authentication.

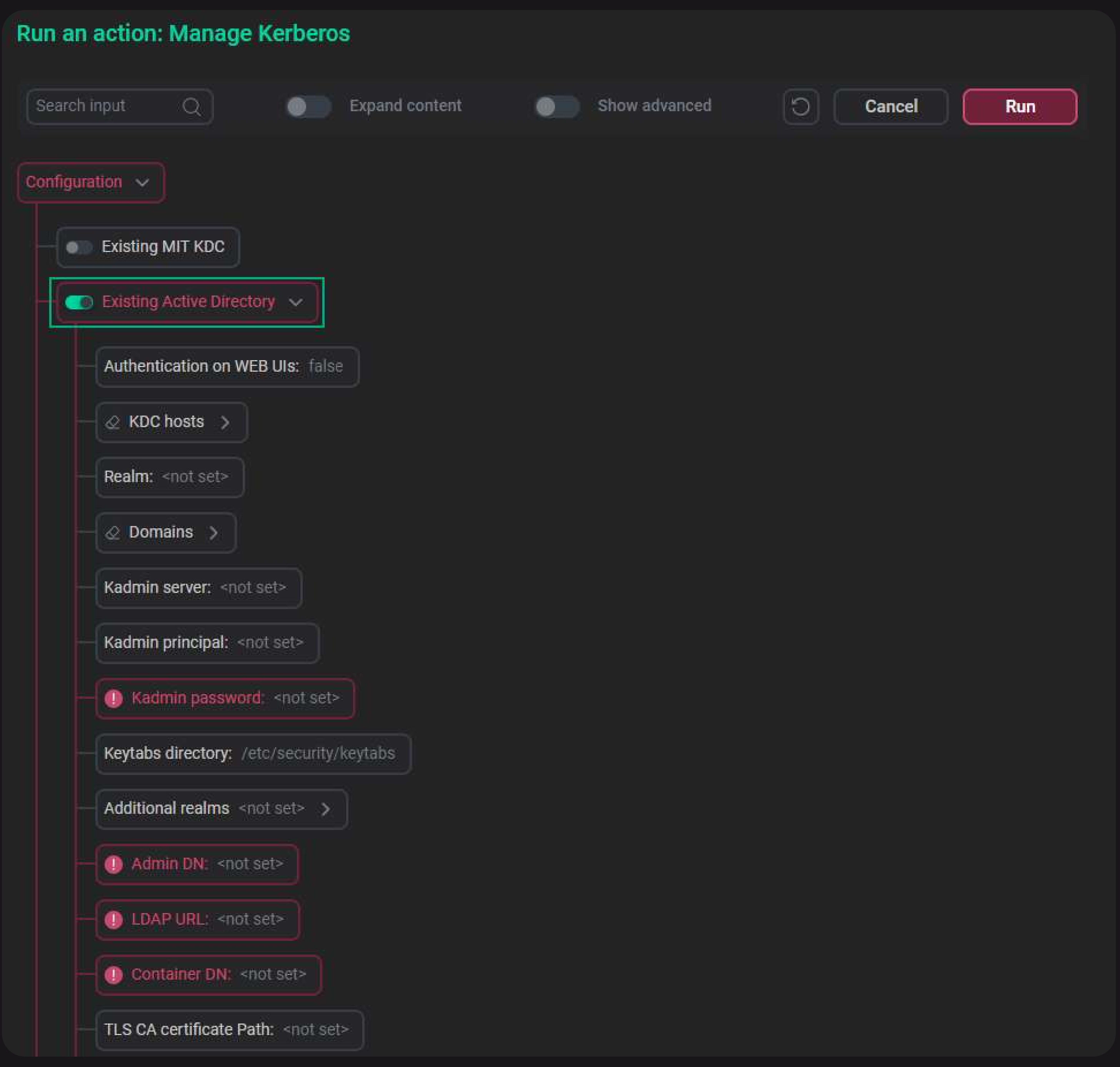
To set up the SPNEGO authentication, complete the following steps:

1. Go to the ADCM UI and select the ***Clusters*** page.
2. Click the ***Action*** icon in your cluster entry and select ***Manage Kerberos***.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters
03 Oct 2024 08:12:49 UTC
1
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
ADH RC
installed
Hadoop
3.3.6_arenadata1_b1-dev
Release candidate ADH
2
Show
10 per page
Check
Manage Credential Encryption
3
Manage Kerberos
Manage SSL
Reinstall status-checker
Start
Stop

Manage Kerberos action

3. In the window that appears, turn on the ***Existing Active Directory*** switch.


Run an action: Manage Kerberos
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
Existing Active Directory
KDC hosts
Domains

Authentication on WEB UIs: false

Realm: <not set>

Kadmin server: <not set>

Kadmin principal: <not set>

Kadmin password: <not set>

Keytabs directory: /etc/security/keytabs

Additional realms <not set>

Admin DN: <not set>

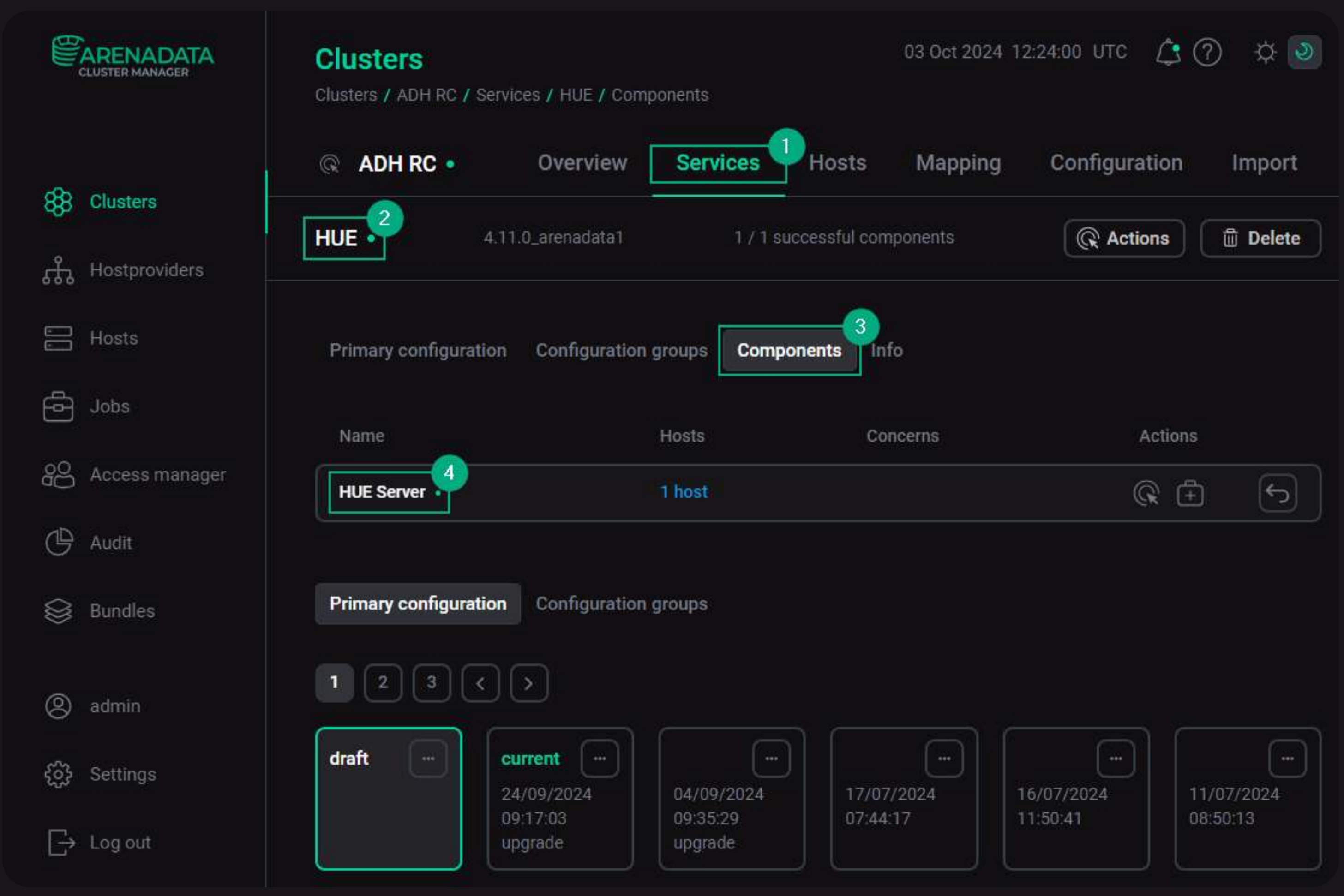
LDAP URL: <not set>

Container DN: <not set>

TLS CA certificate Path: <not set>

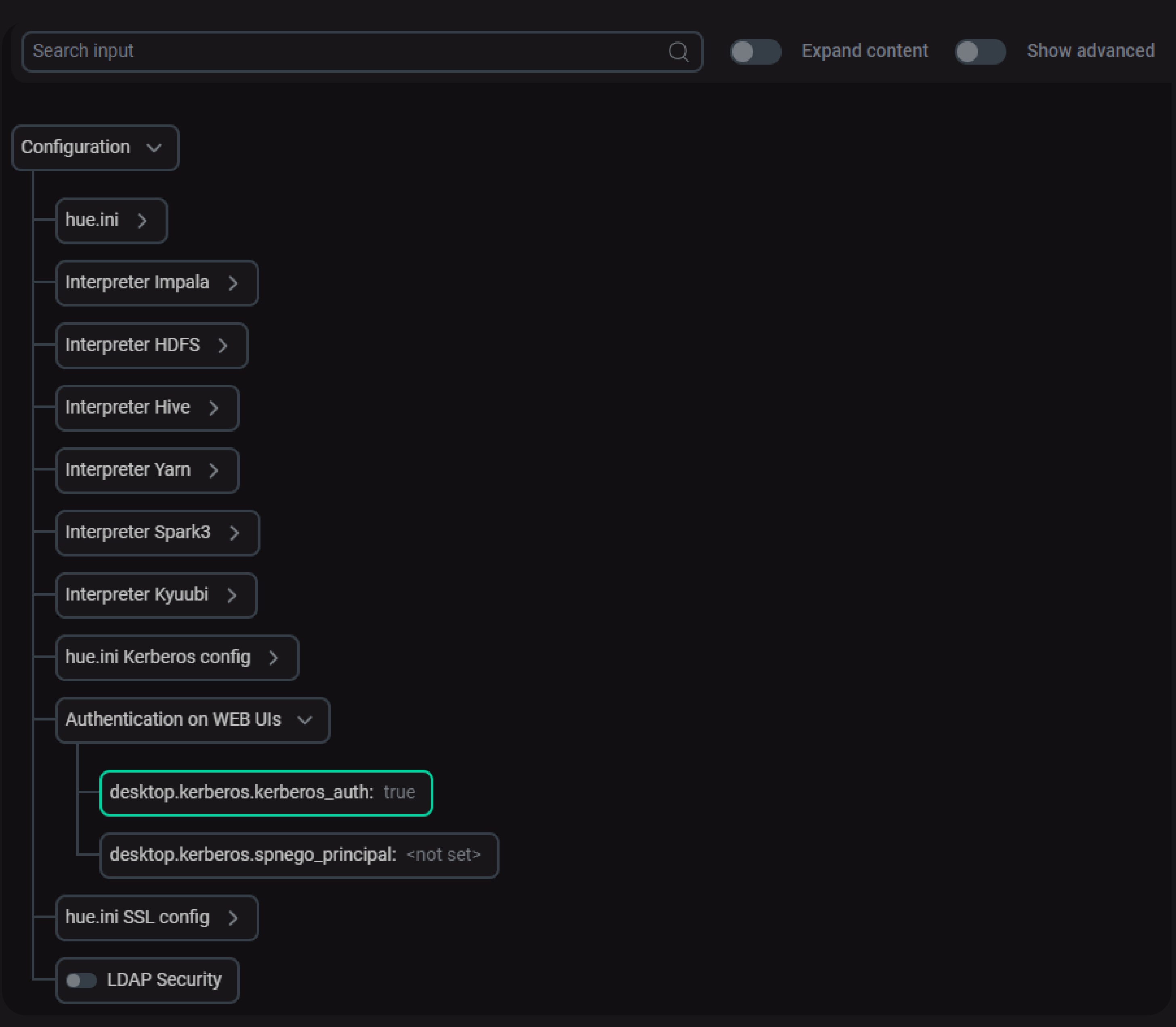
Existing Active Directory parameter group switch

4. Select the ***Authentication on WEB UIs*** parameter and set it to `true`.
5. Specify values for the other parameters in this group according to the Configure Kerberos authentication based on Active Directory via ADCM article.
6. Click ***Run*** and wait for the action to complete.
7. Select your cluster, open the ***HUE*** service page, and on the ***Components*** tab select the ***HUE Server*** component.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / ADH RC / Services / HUE / Components
03 Oct 2024 12:24:00 UTC
ADH RC
Overview
Services
1
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
HUE
2
4.11.0_arenadata1
1 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Components
3
Info
Name
Hosts
Concerns
Actions
HUE Server
4
1 host
Primary configuration
Configuration groups
1
2
3
draft
current
24/09/2024 09:17:03 upgrade
04/09/2024 09:35:29 upgrade
17/07/2024 07:44:17
16/07/2024 11:50:41
11/07/2024 08:50:13

HUE Server component

8. On the ***Primary configuration*** tab, open the ***Authentication on WEB UIs*** parameter group and set the ***desktop.kerberos.kerberos_auth*** parameter to `true`.


Search input
Expand content
Show advanced
Configuration
hue.ini
Interpreter Impala
Interpreter HDFS
Interpreter Hive
Interpreter Yarn
Interpreter Spark3
Interpreter Kyuubi
hue.ini Kerberos config
Authentication on WEB UIs
hue.ini SSL config
LDAP Security

desktop.kerberos.kerberos_auth: true

desktop.kerberos.spnego_principal: <not set>

HUE Server primary configuration

9. If necessary, specify the ***desktop.kerberos.spnego_principal*** parameter. It defines the default Kerberos principal. If it is left empty, then the principal is obtained from the ticket.
10. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Restart HUE Server***.

Before you use the SPNEGO authentication for the HUE UI, make sure that the Kerberos service is installed and set up on your local machine. Also, your browser must be configured to support Kerberos — refer to the SPNEGO authentication article for instructions for Mozilla Firefox and Google Chrome. Run the `kinit` command before establishing connection.

HUE service management via ADCM

Vladimir Adamenkov

To Table of Contents

The ADCM UI provides actions to manage the HUE service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the HUE service are listed in the table below.

HUE service actions

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute HUE components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

HUE components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Use impala-shell to connect to Impala

Eugenia Kuzina

Contents

To Table of Contents

- Run impala-shell
- Configuration options
- The impala-shell configuration file
- Connect to impalad on a specific host
  - Use command-line options
  - Utilize a configuration file
  - Use the CONNECT command
- Connect to a specific database
- Execute SQL statements in non-interactive mode
- Run commands and SQL statements
  - Substitute variables

You can connect to Impala through the shell tool — `impala-shell`. Impala shell is available on hosts where you install the Impala Client component. The tool can connect remotely to any instance of the Impala Daemon.

Impala shell allows you to create and manage databases and tables, insert data, and issue queries. You can also submit SQL statements in an interactive session. It is possible to specify command-line options to process a single statement or a script file that contains a large number of statements.

In addition, you can store a set of default settings for impala-shell in a configuration file.

Run impala-shell

To run the impala-shell, use the command below:

```
$ impala-shell
```

To execute queries, you need to connect to an impalad daemon. The `impala-shell` command connects to an impalad running on the same machine. For information on how to connect to an impalad daemon on another host, see Connect to impalad on a specific host.

NOTE

If the following error occurs: `ImportError: no module named pkg_resources`, install the `python-setuptools` package. To do this, use the command: `yum install python-setuptools`.

Configuration options

You can use configuration options to tune impala-shell for your tasks. The table below lists options that can be passed as a command line parameter or defined in a configuration file.

Impala shell configuration options

Command-line option	Configuration file setting	Description
-B or --delimited	write_delimited=true	Prints query results in plain format as a delimited text file. It is useful for creating data files to be used with other Hadoop components. It also helps to avoid the performance overhead of the pretty-printing output, when running benchmark tests that use queries returning large result sets. To specify the delimiter character, use the `--output_delimiter` option. To define a file name, utilize the `-o` or `--output_file` option
-b or --kerberos_host_fqdn	kerberos_host_fqdn=load-balancer-hostname	Overrides the hostname of the Impala daemon's Kerberos service principal. impala-shell checks whether the server's principal matches this hostname. It can be used when impalad is accessed via a load-balancer, but impala-shell should connect to a specific impalad directly
--print_header	print_header=true	Determines whether the header row should be printed
-o filename or --output_file filename	output_file=filename	Specifies an output file name. It is typically used to store the results of a single query executed from the command line with the `-q` option. It also works for interactive sessions — the messages such as number of rows fetched are displayed but not the actual result set. To suppress these messages (when the `-q` and `-o` options are combined), redirect stderr to `/dev/null`
--output_delimiter=character	output_delimiter=character	Specifies the character to use as a delimiter between fields when query results are printed in plain format (the `-B` option). The default delimiter is tab (`\t`). If an output value contains the delimiter character, that field can be quoted, or escaped by doubling quotation marks, or both
-E or --vertical	vertical=true	Prints the output of a query (rows) vertically. This option has no effects if `-B` is used
-p or --show_profiles	show_profiles=true	Displays the query execution plan (the same output as the `EXPLAIN` statement) and more detailed execution steps
-h or --help	—	Displays help information
—	history_max=1000	Sets the maximum number of queries executed in impala-shell to be stored to the history file
-i hostname or --impalad=hostname[:port_num]	impalad=hostname[:port_num]	Connects to the impalad daemon installed on the specified host. If the `port_number` parameter is not defined, the default value `21050` is used. This option allows you to connect to any host in the cluster. If you access an impalad instance that is running on a different port (the `fe_port` parameter is changed), add this port to the option parameter
-q query or --query=query	query=query	Allows you to issue a single query from the command line, without starting the interactive interpreter. You can use this option to run impala-shell from a shell script or when the command is invoked from a Python, Perl, or other scripts. The query should contain a single statement, for example, `SELECT`, `CREATE TABLE`, `SHOW TABLES`, or any other statement recognized by impala-shell. Since you cannot pass the `USE` statement and another query, fully qualify the names for any table that does not belong to the default database. Alternatively, use the `-f` option to pass a file with the `USE` statement followed by other queries
-f query_file or --query_file=query_file	query_file=path_to_query_file	Allows you to process a file containing multiple SQL statements. For example, a set of DDL statements to create a group of tables and views. Multiple statements must be delimited by the semicolon (`;`). You can specify a filename as `-` to represent the standard input. This feature allows you to use impala-shell as part of a Unix pipeline where SQL statements are generated dynamically by other tools
--query_option="option=value" or -Q "option=value"	The header line — `[impala.query_options]` followed by subsequent lines `option=value`, one option per line	Sets default query options for impala-shell commands. To set multiple query options at once, use multiple instances of this command-line option. The query option names are not case-sensitive
-k or --kerberos	use_kerberos=true	Determines whether the Kerberos authentication should be used when the shell connects to impalad. If Kerberos is not enabled on the instance of impalad to which impala-shell is connecting, an error is displayed. See Enabling Kerberos Authentication for Impala
-s kerberos_service_name or --kerberos_service_name=name	kerberos_service_name=name	Specifies the impala-shell name to authenticate to a particular impalad service principal. If `kerberos_service_name` is not specified, `impala` is used by default. If this option is used for a connection in which Kerberos is not supported, an error is returned
-V or --verbose	verbose=true	Enables the verbose output
--quiet	verbose=false	Disables the verbose output
-v or --version	version=true	Displays the version information of impala-shell
-c	ignore_query_failure=true	Determines whether to ignore the query failure
-d default_db or --database=default_db	default_db=default_db	Specifies the initial database. The same as running the USE statement after the connection. If not specified, the `DEFAULT` database is used
-ssl	ssl=true	Enables TLS/SSL for impala-shell
--ca_cert=path_to_certificate	ca_cert=path_to_certificate	Specifies the local path to the third-party CA certificate, or to a copy of the server certificate for self-signed server certificates. If `--ca_cert` is not set, impala-shell enables TLS/SSL, but does not validate the server certificate. This is useful for connecting to a trusted Impala that is only running over TLS/SSL, when a copy of the certificate is not available, for example, when debugging customer installations
-l	use_ldap=true	Enables LDAP authentication
-u	user=user_name	Specifies the username, when LDAP authentication is enabled. Utilize the short username, not the full LDAP distinguished name. After using this option, the shell prompts for a password
--ldap_password_cmd=command	ldap_password_cmd	Specifies a command to retrieve the LDAP password, when the LDAP authentication is enabled. If the command includes space-separated arguments, enclose the command and its arguments in quotation marks. For example, `ldap_password_cmd=echo -n myPassword` if you store the password in *impalarc* or `ldap_password_cmd=cat /etc/impala/conf/user-pass.pwd` if you store the password in a file
--config_file=path_to_config_file	—	Specifies the path of the file containing impala-shell configuration settings. The default is `/etc/impalarc`. This setting can only be specified on the command line
--live_progress	live_progress=true	Displays a progress bar showing the approximate percentage of completion for each query. The information is updated interactively as the query is processed. See LIVE_PROGRESS Query Option
--disable_live_progress	live_progress=false	Allows you to disable `live_progress` in the interactive mode
--live_summary	—	Shows a detailed report, similar to the `SUMMARY command`, that shows progress details for each phase of the query execution. The information is updated interactively as the query is processed. See LIVE_PROGRESS Query Option
-var=variable_name=value	—	Defines a substitution variable that can be used within the impala-shell session. The variable can be substituted into statements processed by the `-q` or `-f` options, or in an interactive shell session. Within a SQL statement, use the following syntax: `${var:variable_name}`
--auth_creds_ok_in_clear	auth_creds_ok_in_clear	Allows you to use LDAP authentication for an insecure connection to the shell. Note that authentication credentials are sent unencrypted, and may be vulnerable to attack
--protocol=protocol	—	The protocol to use for the connection to Impala. Valid protocol values are: - `hs2` — impala-shell connects to impalad via HiveServer2 over TCP binary. This is the default setting. - `hs2-http` — impala-shell connects to impalad via HiveServer2 over HTTP. - `beeswax` — impala-shell connects to impalad via Beeswax over TCP binary.
--hs2_fp_format=HS2_FP_FORMAT	hs2_fp_format=HS2_FP_FORMAT	Sets the printing format specification for floating point values when using HS2 protocol. The default behavior makes the values handled by Python's `str()` built-in method. Use `16G` to match Beeswax protocol's floating-point output format

The impala-shell configuration file

As mentioned above, you can store a set of impala-shell default settings in a configuration file.

The default path to the global impala-shell configuration file is */etc/impalarc*. The default path to the user-level configuration file is *~/.impalarc*. Note that the global-level file name does not include a dot (`.`). To change the default path to the global configuration file, set the `$IMPALA_SHELL_GLOBAL_CONFIG_FILE` environment variable.

Since ADH Impala configuration files are located in the */etc/impala/conf* directory, you can create an impala-shell configuration file in this directory and set the `$IMPALA_SHELL_GLOBAL_CONFIG_FILE` variable:

```
$ export IMPALA_SHELL_GLOBAL_CONFIG_FILE=/etc/impala/conf
```

It is also possible to start impala-shell with the `--config_file` option set to the custom path of the configuration file.

The options specified in the user configuration file have higher priority than options from the global configuration file. In turn, any options you specify on the impala-shell command line override any corresponding options within the configuration file.

An impala-shell configuration file (global or user) must contain the header — `[impala]`, followed by the impala-shell options.

The names of the options in the configuration file are similar to the long-form command-line arguments to the impala-shell command. They are listed in the table above.

You can specify key/value pair options using `keyval`, similar to the `--var` command-line option. For example, `keyval=variable1=value1`.

The configuration file can also contain query options in the `[impala.query_options]` section. Options specified in the `[impala]` section override the options from the `[impala.query_options]` section.

The example below shows a configuration file where the verbose mode is enabled, so each SQL query output is followed by timing information. impala-shell operates with the `test` database by default. The query output is formatted as delimited text and stored in a file. The configuration file also enables the `show_profiles` option. This option prints detailed performance information after each query.

```
[impala]
verbose=true
default_db=test
write_delimited=true
output_delimiter=,
output_file=/home/test_result.csv
show_profiles=true
keyval=var1=value1,keyval=var2=value2
```

The following example shows a configuration file that connects to a specific Impala node, runs a single query within the `test` database, then exits. Any query options predefined under the `[impala.query_options]` section are applied during the session.

```
[impala]
impalad=ees-adh1.ru-central1.internal
default_db=test
# Run the query and exit
query=select count(*) from customers

[impala.query_options]
mem_limit=32g
```

You can use this type of configuration file with the command-line option `--config_file`, to easily select between many predefined queries that could be run against different databases, hosts, or even different clusters. To run a sequence of statements instead of a single query, create a file with statements and specify the configuration option `query_file=path_to_query_file` of the query text.

Connect to impalad on a specific host

To connect to an impalad daemon on another host, you need to specify the connection parameters in one of the ways described below.

Use command-line options

You can use command-line options when you run the impala-shell command:

```
$ impala-shell -i ees-1adh3.ru-central1.internal:28000 --protocol='hs2-http'
```

This example connects to the impalad on the host `ees-1adh3.ru-central1.internal`, on the port `28000`, via HiveServer2 over HTTP

Utilize a configuration file

It is possible to specify connection settings in a file. Create a file with connection settings as described in The impala-shell configuration file section. For example:

```
$ cat > ~/custom_conf1
[impala]
impalad=ees-1adh2.ru-central1.internal
query=select version()
```

When using the `cat` command, press `CTRL+D` to exit.

Run impala-shell:

```
$ impala-shell --config_file=~/custom_conf1
```

The result:

```
Starting Impala Shell with no authentication using Python 2.7.5
Warning: live_progress only applies to interactive shell sessions, and is being skipped for now.
Opened TCP connection to ees-1adh2.ru-central1.internal:21050
Connected to ees-1adh2.ru-central1.internal:21050
Server version: impalad version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6)
Query: select version()
Query submitted at: 2023-07-25 14:36:57 (Coordinator: http://ees-1adh2.ru-central1.internal:25000)
Query progress can be monitored at: http://ees-1adh2.ru-central1.internal:25000/query_plan?
query_id=084b59535cac1baf:5f5e61f500000000
+-------------------------------------------------------------------------------------------+
| version()                                                                                 |
+-------------------------------------------------------------------------------------------+
| impalad version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6) |
| Built on Mon Jun  5 16:33:03 UTC 2023                                                     |
+-------------------------------------------------------------------------------------------+
Fetched 1 row(s) in 0.44s
```

Use the CONNECT command

Execute the `CONNECT` command during an impala-shell session:

```
$ impala-shell
```

```
connect ees-1adh3.ru-central1.internal;
```

The result:

```
Opened TCP connection to ees-1adh3.ru-central1.internal:21050
Connected to ees-1adh3.ru-central1.internal:21050
Server version: impalad version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6)
[ees-1adh3.ru-central1.internal:21050] default>
```

Connect to a specific database

You can use all the same connection options as in previous examples, and add the `-d` option with the database name:

```
$ impala-shell -i localhost -d my_db
```

`localhost` refers to the same host where impala-shell is running.

Execute SQL statements in non-interactive mode

Invoke impala-shell with the `-q` option to run a single statement, or the `-f` option to run a sequence of statements from a file. The impala-shell command returns a result immediately, without going into the interactive mode:

```
$ impala-shell -i localhost -d my_db -q 'show tables'

$ impala-shell -i localhost -d my_db -f recreate_tables.sql
```

Run commands and SQL statements

The impala-shell interpreter accepts the SQL statements described in the Impala SQL Statements article and shell commands that you can use for tuning performance and diagnosing problems.

In non-interactive mode, query output is printed to stdout or to the file specified by the `-o` option. Errors are printed to stderr. So, you can process the query output as a part of a Unix pipeline.

In interactive mode, impala-shell uses the Readline library to recall and edit previous commands.

You can use the following keyboard shortcuts and syntax rules for running commands and SQL statements in impala-shell:

- To see the full list of available commands, press `TAB` twice.
- To display previous commands, click the up-arrow and down-arrow keys.
- Use the standard set of keyboard shortcuts in GNU Readline library for editing and cursor movement, such as `Ctrl+A` for the beginning of line and `Ctrl+E` for the end of line.
- Commands and SQL statements can span multiple lines and must be terminated by a semicolon (`;`).
- Use `--` to comment out a single line comment and `/* <content> */` to comment multi lines. A comment is considered as a part of the statement it precedes. When you type a comment, you get a continuation prompt until you enter a semicolon. For example:

```
default> -- Show tables starting with t
> SHOW TABLES LIKE 't*';
```

If a comment contains the `${variable_name}` and it is not for a variable substitution, the `$` character must be escaped, for example — `\${hello}`.

The example below uses impala-shell commands and the `test` database and the `table1` table:

```
create database test;
use test;
create table table1 (field1 int, field2 boolean, field3 string);
insert into table1 values ((1,true,"row1"), (2,true,"row2"), (3,false,"row3"));
select * from table1;
```

The result:

```
+--------+--------+--------+
| field1 | field2 | field3 |
+--------+--------+--------+
| 1      | true   | row1   |
| 2      | true   | row2   |
| 3      | false  | row3   |
+--------+--------+--------+
```

Substitute variables

You can define variables to be used within SQL statements processed by impala-shell. To do this on the command line, specify the `--var=variable_name=value` option. Within an interactive session or a script file processed by the `-f` option, use the `SET VAR:variable_name=value` command. To pass a variable in an SQL statement, use the notation: `${VAR:variable_name}`.

The examples below contain multiple commands that define variables and use them in SQL statements executed through the `-q` option. The `-q` argument strings containing variable references (`${var:value}`) are single-quoted.

Example 1:

```
$ impala-shell --var=tname=table1 --var=colname=col1 --var=coltype=string -q 'CREATE TABLE ${var:tname} (${var:colname}
${var:coltype}) STORED AS PARQUET'
```

The result:

```
Query: CREATE TABLE table1 (col1 string) STORED AS PARQUET
+-------------------------+
| summary                 |
+-------------------------+
| Table has been created. |
+-------------------------+
```

Example 2:

```
$ impala-shell --quiet --var=tname=table1
```

```
use test;
select count(*) from ${var:tname};
```

The result:

```
+----------+
| count(*) |
+----------+
| 3        |
+----------+
```

Example 3:

```
$ impala-shell
```

```
set var:tname=table1;
select count(*) from ${var:tname};
```

The result:

```
+----------+
| count(*) |
+----------+
| 3        |
+----------+
```

Use JDBC to connect to Impala

Eugenia Kuzina

Contents

To Table of Contents

Configure the JDBC Port

Enable Impala JDBC support on client systems

Connection strings for the Hive JDBC Driver

JDBC limitations

Impala supports the standard JDBC interface that allows access from commercial Business Intelligence tools and other software.

To set up a JDBC connection to Impala, follow the steps below:

1. Verify the port where the Impala daemons in your cluster are listening for incoming JDBC requests.
2. Install the JDBC driver on every host that runs the JDBC-enabled application.
3. Specify a connection string for the JDBC application to access one of the servers running the impalad daemon, with the appropriate security settings.

Configure the JDBC Port

The default ports on which Impala Daemons accepts JDBC connections are listed in the table below.

Protocol	Default Port	Parameter
HTTP	28000	hs2_http_port
Binary TCP	21050	hs2_port

To change these ports, go to the ***Components*** page of the ***Impala*** service in ADCM UI, select ***Impala Daemon***, open its ***Primary Configuration*** tab, and set the required parameter(***hs2_http_port*** or ***hs2_port***).

Enable Impala JDBC support on client systems

The Hive JDBC driver is installed on ADH hosts with Impala. This is the *hive-jdbc.noarch* package.

If you are using JDBC-enabled applications on hosts outside the cluster, install the JDBC driver (the *hive-jdbc* package) on at least one cluster host. For this, use the Linux package manager. Then download the following JAR files to each client machine that should use JDBC with Impala:

- commons-logging-X.X.X.jar
- hadoop-common.jar
- hive-common-X.XX.X.jar
- hive-jdbc-X.XX.X.jar
- hive-metastore-X.XX.X.jar
- hive-service-X.XX.X.jar
- httpclient-X.X.X.jar
- httpcore-X.X.X.jar
- libfb303-X.X.X.jar
- libthrift-X.X.X.jar
- log4j-X.X.XX.jar
- slf4j-api-X.X.X.jar
- slf4j-logXjXX-X.X.X.jar

To enable the JDBC support for Impala on the system where you run the JDBC application, perform the following steps:

1. Download the JAR files listed above to each client machine. For Maven users, see the sample github page for an example of the dependencies you could add to a *pom* file instead of downloading the individual JARs.
2. Store the JAR files. You can do it in a directory that already is referenced in your CLASSPATH setting. On Linux, you can use the location */opt/jars/*. On Windows, you can use a subdirectory in the *C:\Program Files* folder.
3. Set CLASSPATH to include the JAR files. To successfully load the Impala JDBC driver, client programs must find the associated JAR files. For this, in most cases, you need to set the CLASSPATH variable. You can find some examples below. For more information about installing JDBC drivers, see the documentation for your JDBC client.
   On Linux, if you extracted the JARs to */opt/jars/*, you can execute the following command:

```
$ export CLASSPATH=/opt/jars/*.jar:$CLASSPATH
```

   On Windows, use the ***System Properties*** item of ***Control Panel*** to modify the environment variables for your system. Change the environment variables to include the path to the extracted JAR files.

The class name of the Hive JDBC driver is `org.apache.hive.jdbc.HiveDriver`.

Connection strings for the Hive JDBC Driver

After you configured Impala to work with JDBC, you can establish a connection.

IMPORTANT

When using the HTTP protocol as the transport mode, make sure to execute the `kinit` command before connecting to the database.

Without Kerberos

For a cluster that does not use Kerberos authentication, utilize the following connection string syntax:

```
jdbc:hive2://<hostname>:<port>/;auth=noSasl
```

Example (binary TCP protocol):

```
jdbc:hive2://host1.example.com:21050/;auth=noSasl
```

Example (HTTP protocol):

```
jdbc:hive2://host1.example.com:28000/;transportMode=http;auth=noSasl
```

SSL

To connect to an Impala instance with SSL enabled:

```
jdbc:hive2://<hostname>:<port>/;ssl=true;sslTrustStore=<trust_store_path>;trustStorePassword=<trust_store_password>;principal=
<principal_name>
```

Example (binary TCP protocol):

```
jdbc:hive2://host1.example.com:21050/;ssl=true;sslTrustStore=/tmp/truststore.
jks;trustStorePassword=bigdata;principal=hive/_HOST@EXAMPLE.COM
```

Example (HTTP protocol):

```
jdbc:hive2://host1.example.com:28000/;transportMode=http;ssl=true;sslTrustStore=/tmp/truststore.
jks;trustStorePassword=bigdata;principal=hive/_HOST@EXAMPLE.COM
```

SSL+Kerberos

To connect to an Impala instance that requires Kerberos authentication with SSL enabled:

```
jdbc:hive2://<hostname>:<port>/;principal=<principal_name>/hostname@EXAMPLE.COM;ssl=true;sslTrustStore=
<truststore_path>;trustStorePassword=<truststore_pass>
```

The principal must be the same user principal that you use when starting Impala.

Example (binary TCP protocol):

```
jdbc:hive2://host1.example.com:21050/;principal=impala/host1.example.com@EXAMPLE.COM;ssl=true;sslTrustStore=/tmp/truststore.
jks;trustStorePassword=bigdata
```

Example (HTTP protocol):

```
jdbc:hive2://host1.example.com:28000/;transportMode=http;principal=impala/host1.example.com@EXAMPLE.
COM;ssl=true;sslTrustStore=/tmp/truststore.jks;trustStorePassword=bigdata
```

LDAP

To connect to an Impala instance that requires LDAP authentication:

```
jdbc:hive2://<hostname>:<port>/<db_name>;user=<ldap_userid>;password=<ldap_password>
```

Example (binary TCP protocol):

```
jdbc:hive2://host1.example.com:21050/test_db;user=user1;password=password1
```

Example (HTTP protocol):

```
jdbc:hive2://host1.example.com:28000/test_db;transportMode=http;user=user1;password=password1
```

JDBC limitations

Most Impala SQL features work equivalently through the impala-shell interpreter and the JDBC API, but there are some exceptions:

- Queries involving the complex types (`ARRAY`, `STRUCT`, and `MAP`) require notation that might not be available on all levels of the JDBC driver. You can create a view that exposes a "flattened" version of the complex columns and point the application at the view. For more information, see Complex Types.
- The Impala complex types are supported by the JDBC `getColumns()` method. Both `MAP` and `ARRAY` are reported as the JDBC SQL type `ARRAY`. This behavior is consistent with Hive. The `STRUCT` type is reported as the JDBC SQL type `STRUCT`. To be consistent with Hive, the `TYPE_NAME` field is populated with the primitive type name for scalar types. For complex types, the `toSQL()` method is executed. The resulting type names are not fully consistent. The following table shows how `toSQL()` converts types.

Initial type	Resulting type
DECIMAL(10,10)	DECIMAL
CHAR(10)	CHAR
VARCHAR(10)	VARCHAR
ARRAY<DECIMAL(10,10)>	ARRAY<DECIMAL(10,10)>
ARRAY<CHAR(10)>	ARRAY<CHAR(10)>
ARRAY<VARCHAR(10)>	ARRAY<VARCHAR(10)>

- Currently, Impala `INSERT`, `UPDATE`, or other DML statements executed through the JDBC interface against a Kudu table do not return JDBC errors for conditions such as duplicate primary key columns. Therefore, in applications that execute a large number of DML statements, it is better to use the Kudu Java API directly rather than JDBC.

Impala web UIs overview

Eugenia Kuzina

Contents

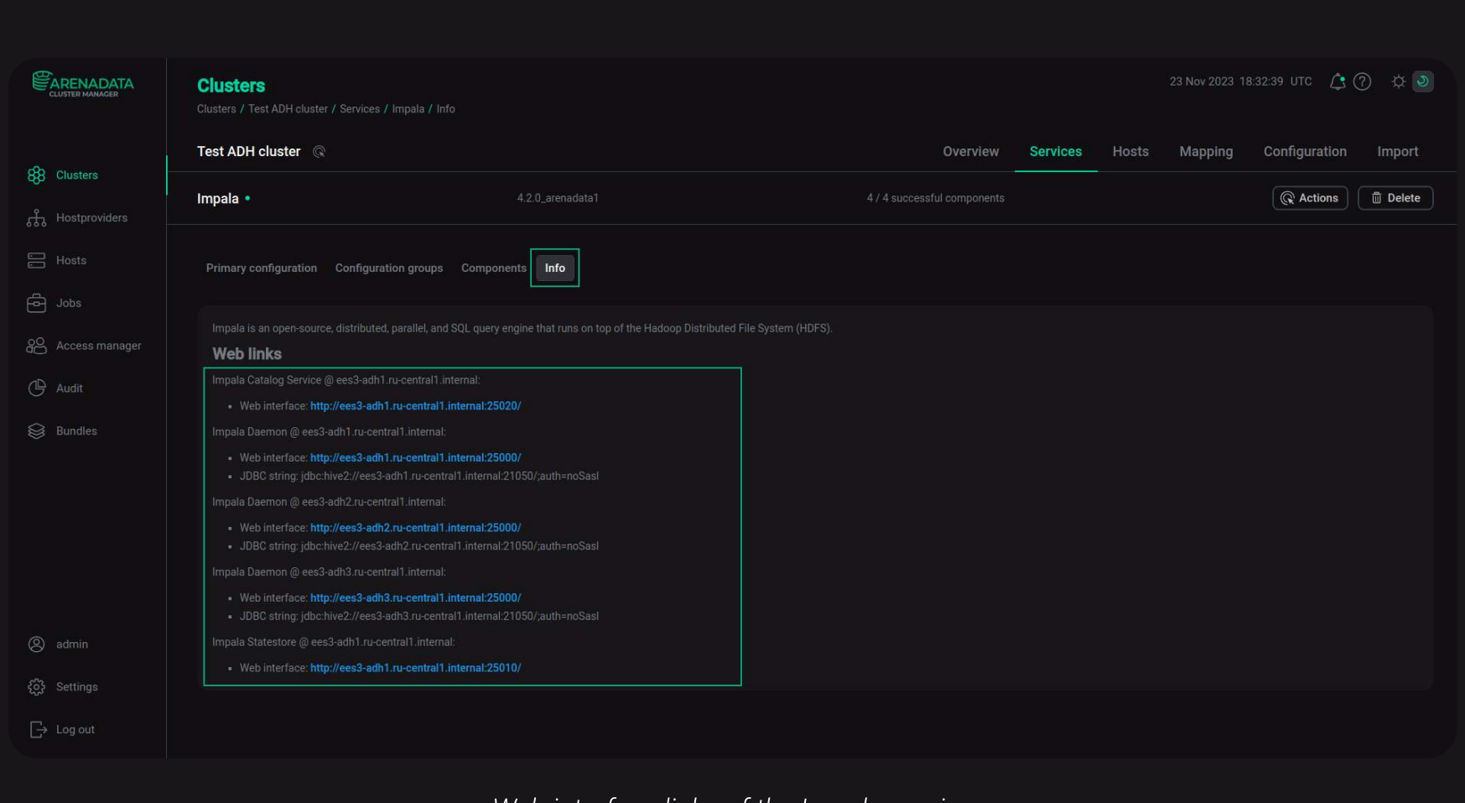
To Table of Contents

Impala Daemon (impalad) web UI

Impala Statestore (statestored) web UI

Impala Catalog Service (catalogd) web UI

Each of the Impala daemons (*impalad*, *statestored*, and *catalogd*) provides a built-in web server that displays diagnostic and status information. You can find links to these web interfaces on the ***Info*** tab of the Impala service in the ADCM UI.



Impala is an open-source, distributed, parallel, and SQL query engine that runs on top of the Hadoop Distributed File System (HDFS).

Web links

Impala Catalog Service @ ees3-adh1.ru-central1.internal:
- Web interface: http://ees3-adh1.ru-central1.internal:25020/

Impala Daemon @ ees3-adh1.ru-central1.internal:
- Web interface: http://ees3-adh1.ru-central1.internal:25000/
- JDBC string: jdbc:hive2://ees3-adh1.ru-central1.internal:21050/;auth=noSasl

Impala Daemon @ ees3-adh2.ru-central1.internal:
- Web interface: http://ees3-adh2.ru-central1.internal:25000/
- JDBC string: jdbc:hive2://ees3-adh2.ru-central1.internal:21050/;auth=noSasl

Impala Daemon @ ees3-adh3.ru-central1.internal:
- Web interface: http://ees3-adh3.ru-central1.internal:25000/
- JDBC string: jdbc:hive2://ees3-adh3.ru-central1.internal:21050/;auth=noSasl

Impala Statestore @ ees3-adh1.ru-central1.internal:
- Web interface: http://ees3-adh1.ru-central1.internal:25010/

Web interface links of the Impala service

A URL has the following form: `http://<server-hostname>:<port>/`.

Example: `http://adh-host1.ru-central1.internal:25000/`.

For secure clusters, use `https://` instead of `http://`.

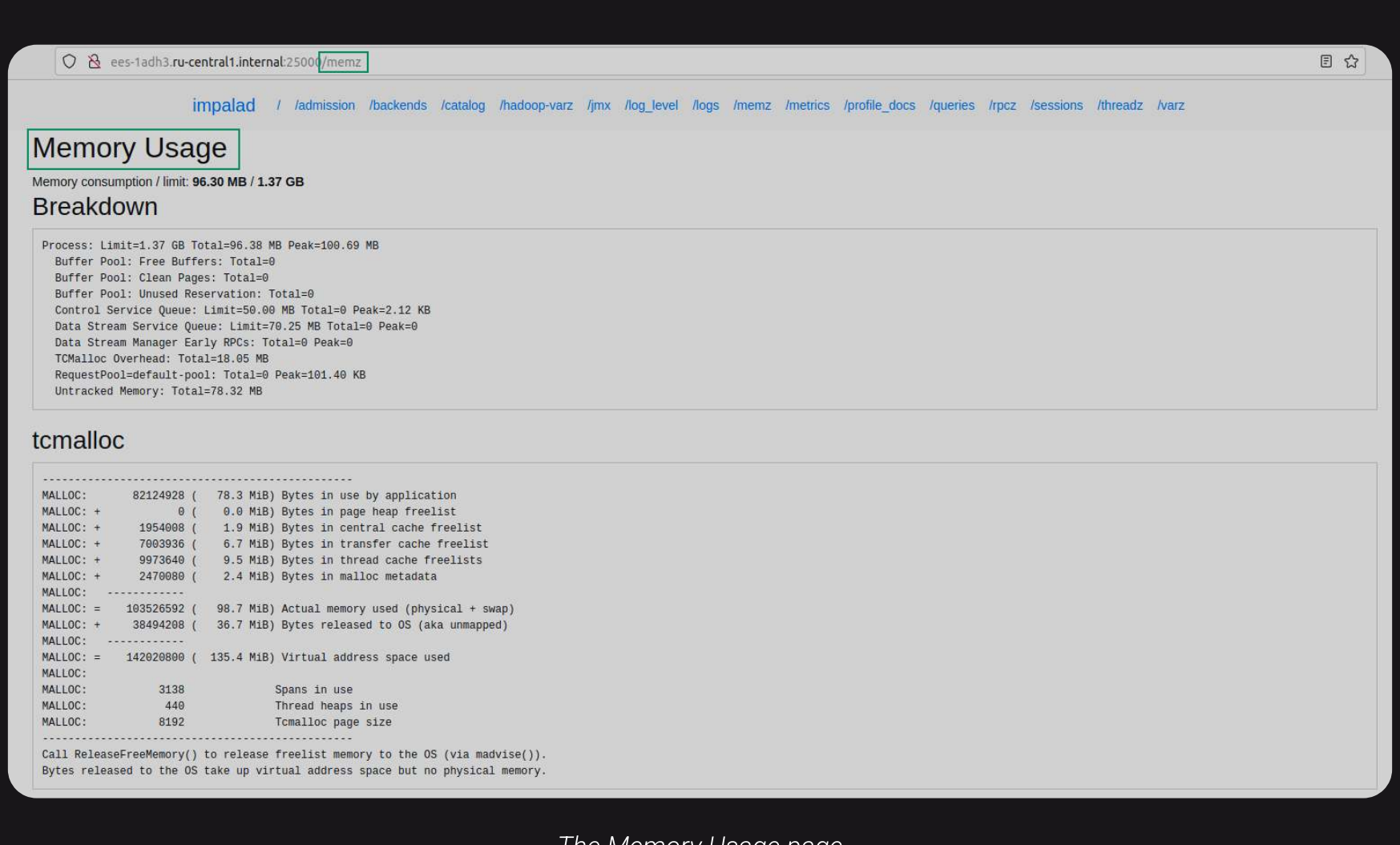
The default web server ports are listed in the table below.

Component	Daemon	Port
Impala Daemon	impalad	25000
Impala Statestore	statestored	25010
Impala Catalog Service	catalogd	25020

To change the default web server port, open the ***Primary Configuration*** tab of a corresponding Impala component and set the ***webserver_port*** parameter. To disable Web UI, disable the ***enable_webserver*** checkbox and restart the required component using the ***Restart Impala Daemon***, or ***Restart Impala Statestore***, or ***Restart Impala Catalog*** component action.

Impala Web UIs support SPNEGO to provide security through Kerberos. This protocol passes negotiation tokens through HTTP headers to secure HTTP requests with Kerberos. To enable authorization using SPNEGO, activate the ***webserver_require_spnego*** checkbox (on the component ***Primary Configuration*** tab in the ADCM UI) for each component that should utilize SPNEGO.

You can use the menu at the top of the main web UIs pages to navigate through the pages described below. Alternatively, add a slash (`/`) and the page name (for example, `/memz`) to the HTTP address in the browser to open this page.



Memory Usage

Memory consumption / limit: **96.30 MB / 1.37 GB**

Breakdown

```
Process: Limit=1.37 GB Total=96.38 MB Peak=100.69 MB
  Buffer Pool: Free Buffers: Total=0
  Buffer Pool: Clean Pages: Total=0
  Buffer Pool: Unused Reservation: Total=0
  Control Service Queue: Limit=50.00 MB Total=0 Peak=2.12 KB
  Data Stream Service Queue: Limit=70.25 MB Total=0 Peak=0
  Data Stream Manager Early RPCs: Total=0 Peak=0
  TCMalloc Overhead: Total=18.05 MB
  RequestPool=default-pool: Total=0 Peak=101.40 KB
  Untracked Memory: Total=78.32 MB
```

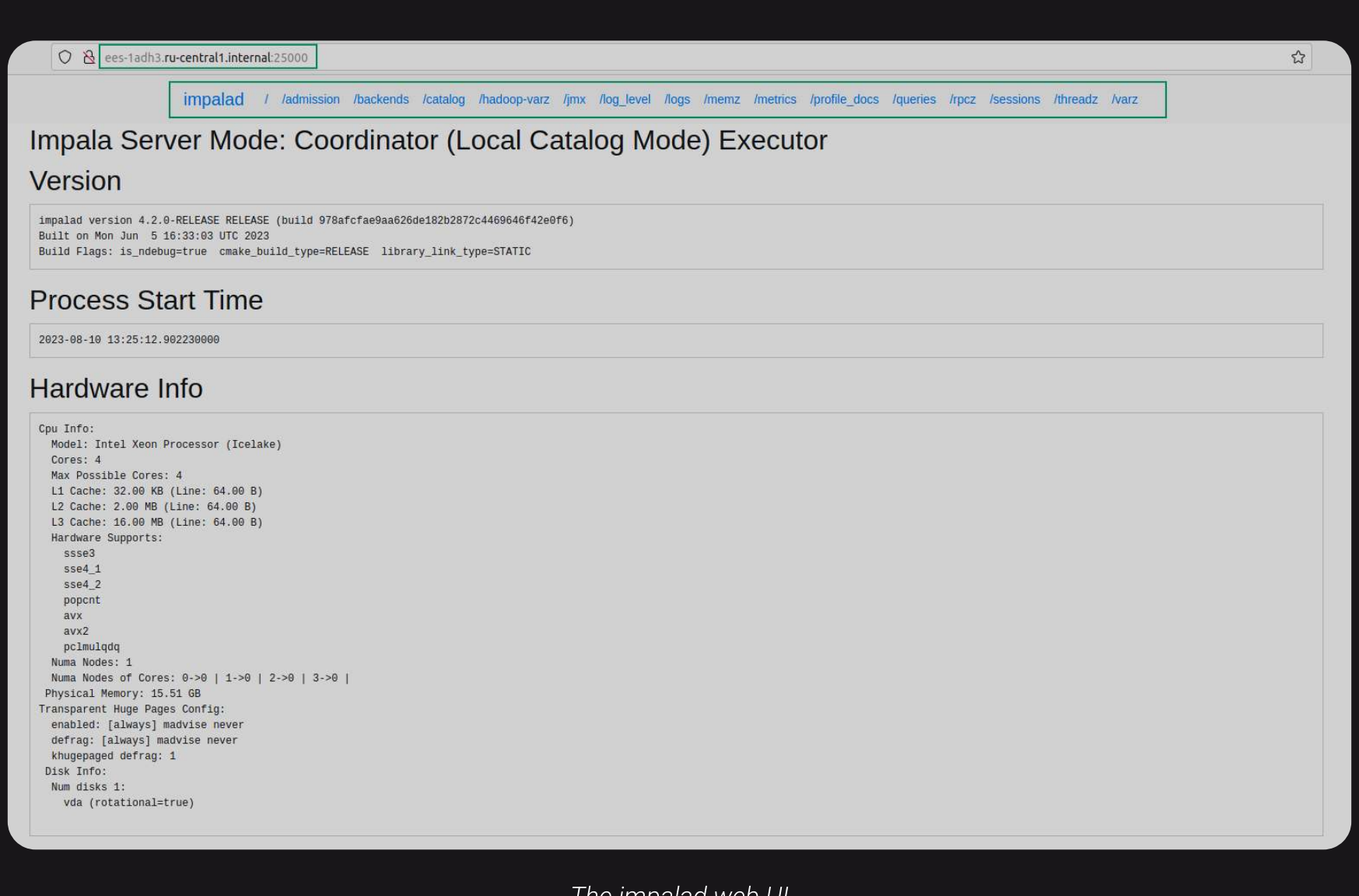
tcmalloc

```
------------------------------------------------
MALLOC:       82124928 (   78.3 MiB) Bytes in use by application
MALLOC: +            0 (    0.0 MiB) Bytes in page heap freelist
MALLOC: +      1954008 (    1.9 MiB) Bytes in central cache freelist
MALLOC: +      7003936 (    6.7 MiB) Bytes in transfer cache freelist
MALLOC: +      9973640 (    9.5 MiB) Bytes in thread cache freelists
MALLOC: +      2470080 (    2.4 MiB) Bytes in malloc metadata
MALLOC:   ------------
MALLOC: =    103526592 (   98.7 MiB) Actual memory used (physical + swap)
MALLOC: +     38494208 (   36.7 MiB) Bytes released to OS (aka unmapped)
MALLOC:   ------------
MALLOC: =    142020800 (  135.4 MiB) Virtual address space used
MALLOC:
MALLOC:           3138              Spans in use
MALLOC:            440              Thread heaps in use
MALLOC:           8192              Tcmalloc page size
------------------------------------------------
Call ReleaseFreeMemory() to release freelist memory to the OS (via madvise()).
Bytes released to the OS take up virtual address space but no physical memory.
```

The Memory Usage page

Impala Daemon (impalad) web UI

The impalad web UI includes information about configuration settings, running and completed queries, and associated performance and resource usage for queries.



Impala Server Mode: Coordinator (Local Catalog Mode) Executor

Version

```
impalad version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6)
Built on Mon Jun  5 16:33:03 UTC 2023
Build Flags: is_ndebug=true  cmake_build_type=RELEASE  library_link_type=STATIC
```

Process Start Time

```
2023-08-10 13:25:12.902230000
```

Hardware Info

```
Cpu Info:
  Model: Intel Xeon Processor (Icelake)
  Cores: 4
  Max Possible Cores: 4
  L1 Cache: 32.00 KB (Line: 64.00 B)
  L2 Cache: 2.00 MB (Line: 64.00 B)
  L3 Cache: 16.00 MB (Line: 64.00 B)
  Hardware Supports:
    ssse3
    sse4_1
    sse4_2
    popcnt
    avx
    avx2
    pclmulqdq
  Numa Nodes: 1
  Numa Nodes of Cores: 0->0 | 1->0 | 2->0 | 3->0 |
 Physical Memory: 15.51 GB
Transparent Huge Pages Config:
  enabled: [always] madvise never
  defrag: [always] madvise never
  khugepaged defrag: 1
 Disk Info:
  Num disks 1:
    vda (rotational=true)
```

The impalad web UI

Each host that runs the impalad daemon has its own instance of the web UI, with details about queries for which this host was a coordinator. To investigate a problem with queries, select the node that was the coordinator for them.

The main impalad web UI page contains the following information about impalad:

- ***Version*** — the version of the daemon and information about when Impala was built and what build flags were used.
- ***Process start time*** — the time when the daemon was started.
- ***Hardware info*** — information about the host hardware.
- ***OS information*** — information about the operating system that is installed on the host.
- ***Process information*** — information about the daemon process.
- ***CGroup information*** — information about CGroup settings.

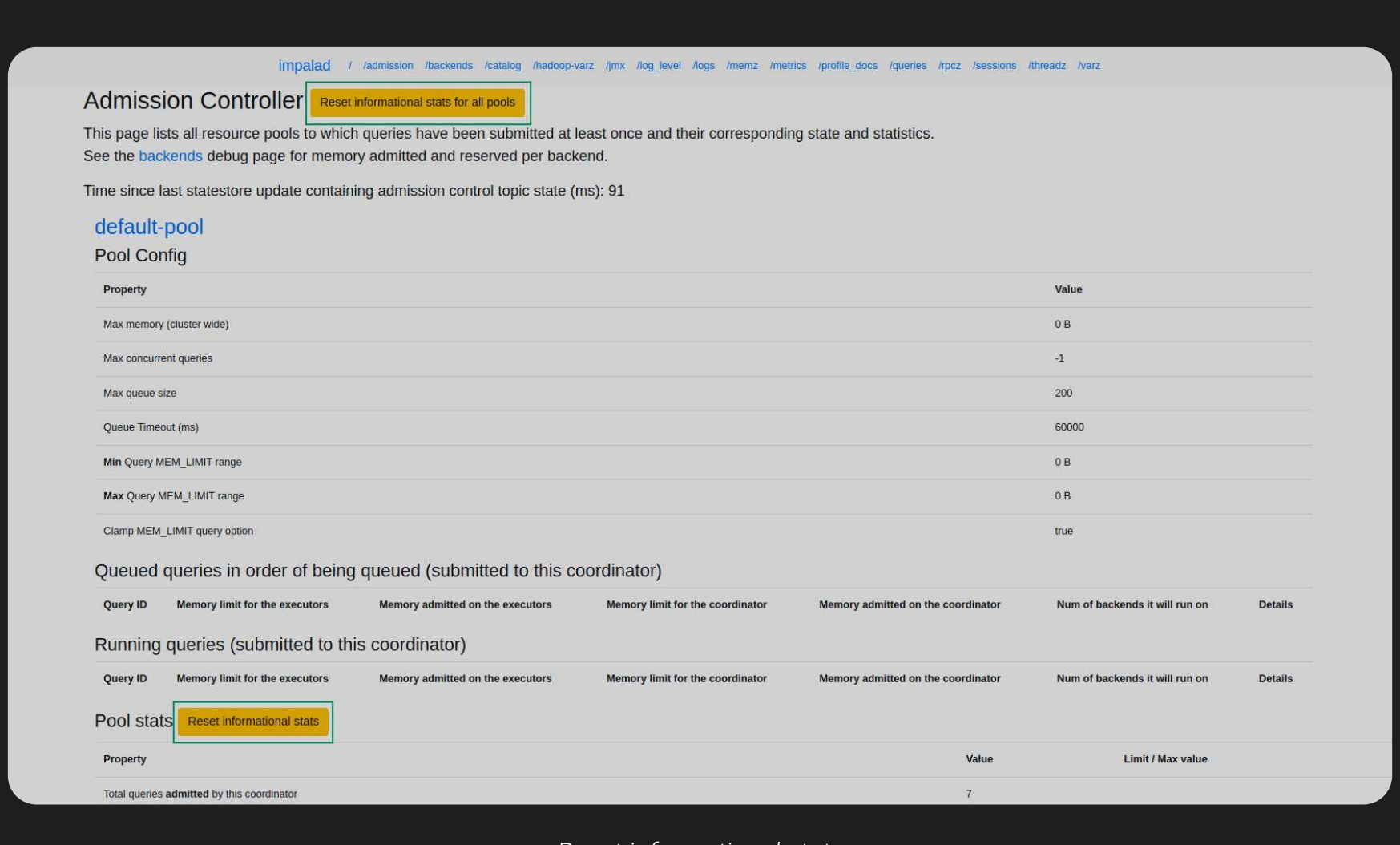
admission

The ***Admission Controller*** page lists all resource pools to which queries have been submitted at least once and their state and statistics.

The page provides the following information about each resource pool:

- The time since the statestored received the last update.
- A warning if this impalad is disconnected from the statestored and the information on this page can be out of date.
- Pool configuration.
- Queued queries submitted to this coordinator in the order of submission.
- Running queries submitted to this coordinator.
- Pool statistics. The time in queue parameter is calculated as an exponential moving average which represents the average time in queue over the last 10 to 12 queries. If a query is accepted immediately, this average wait time is calculated using the wait time `0`.
- Histogram of the peak memory distribution. It takes into account the memory used by queries admitted to the pool. The histogram displays data for all queries admitted to the pool, including completed, canceled, and failed queries.

You can click ***Reset informational stats*** to reset the statistics of historical data, such as total stats, time in queue, and the histogram for a current pull or click ***Reset informational stats for all pulls*** to clear statistics for all pools.



Admission Controller

This page lists all resource pools to which queries have been submitted at least once and their corresponding state and statistics.
See the backends debug page for memory admitted and reserved per backend.

Time since last statestore update containing admission control topic state (ms): 91

default-pool

Pool Config

Property	Value
Max memory (cluster wide)	0 B
Max concurrent queries	-1
Max queue size	200
Queue Timeout (ms)	60000
Min Query MEM_LIMIT range	0 B
Max Query MEM_LIMIT range	0 B
Clamp MEM_LIMIT query option	true

Queued queries in order of being queued (submitted to this coordinator)

Query ID	Memory limit for the executors	Memory admitted on the executors	Memory limit for the coordinator	Memory admitted on the coordinator	Num of backends it will run on	Details

Running queries (submitted to this coordinator)

Query ID	Memory limit for the executors	Memory admitted on the executors	Memory limit for the coordinator	Memory admitted on the coordinator	Num of backends it will run on	Details

Pool stats

Property	Value	Limit / Max value
Total queries **admitted** by this coordinator	7	

Reset informational stats

The Admission Controller statistics are also available as a JSON object from the following HTTP endpoint: `http://<server-hostname>:<port>/admission?json`. Example: `/http://adh-host1.ru-central1.internal:25000/admission?json`. For more information, see Admission Control and Query Queuing.

backends

This page lists the following info for each of the impalad nodes in the cluster:

- ***Address*** — address of the node (a host name and port).
- ***KRPC Address*** — a KRPC address of the node. It can be used in the `SHUTDOWN` command.
- ***Backend Id***.
- ***Coordinator*** — a boolean value that shows whether the node is a coordinator.
- ***Executor*** — a boolean value that shows whether the node is an executor.
- ***Memory Limit for Admission*** — the amount of memory that can be admitted to this backend by the admission controller.
- ***Memory Reserved*** — the amount of memory reserved by queries on this backend.
- ***Memory Admitted by Queries Submitted to this Coordinator*** — the memory required for queries processed by this coordinator.
- ***Admission Control Slots In Use*** — the number of used Admission control slots.
- ***Num. Queries Admitted by this Coordinator*** — the number of queries processed by the current coordinator.
- ***Executor Groups***.

If there is any stopped impalad daemon, nodes are divided into two tables: ***Active backend(s)*** and ***Quiescing backend(s)***.

Since each impalad daemon knows about other impalad daemons through Statestore, all impalad web servers should display the same information.

catalog

The ***Catalog*** page displays a list of databases and associated tables. You can use this page to determine which database includes a particular table, check the spelling of a database or table name, search for the same table names in multiple databases. The main use case in debugging would be to check if the impalad instance knows about a table from a particular database.

hadoop-varz

This page displays the Hadoop common configuration settings with which Impala is running.

jmx

This page displays monitoring information about various JVM subsystems in the JSON format, for example, memory pools, thread management, runtime, and others.

log_level

This page displays the current Java and backend log levels and allows you to change the log levels dynamically without restarting the daemon.

logs

This page shows the latest entries of the *impalad.INFO* log file, including the `info`, `warning`, and `error` logs. You can see the details of the most recent operations to determine whether the operations succeeded or encountered errors.

This page allows you to avoid searching log files in the file system.

memz

This page displays the summary and detailed information about memory usage by the impalad daemon.

metrics

The page displays the current set of metrics, counters, and flags representing various aspects of the daemon internal operations.

profile_docs

The page lists counters of the `PROFILE` statement.

queries

The page contains the following tables:

- ***Queries in flight*** — running queries.
- ***Waiting to be closed*** — queries that have completed their execution, but have not been closed yet.
- ***Last 100 completed queries*** — the latest completed queries. You can control the number of queries and therefore the amount of memory allocated to store information about completed queries, by specifying the `--query_log_size` startup option for impalad.
- ***Query Locations*** — shows how running queries are distributed among the impalad hosts.

The queries are listed in reverse chronological order, with the most recent at the top.

The ***Details*** link for each query displays alternative views of the query including a graphical representation of the plan, and the output of the `EXPLAIN`, `SUMMARY`, and `PROFILE` statements from impala-shell. On the ***Profile*** tab of the query ***Details*** page, you can export the query profile output to the Thrift, JSON, or text format.

rpcz

The page displays information about the duration of RPC communications of the current daemon with other Impala daemons.

sessions

The page displays information about the sessions currently connected to this impalad instance. For example, sessions can include connections from impala-shell or JDBC.

threadz

The page displays information about the threads used by the current daemon and shows which categories they are grouped into.

varz

The page shows the configuration settings that are used when this daemon communicates with other Hadoop components such as HDFS and YARN. The settings are collected from configuration files.

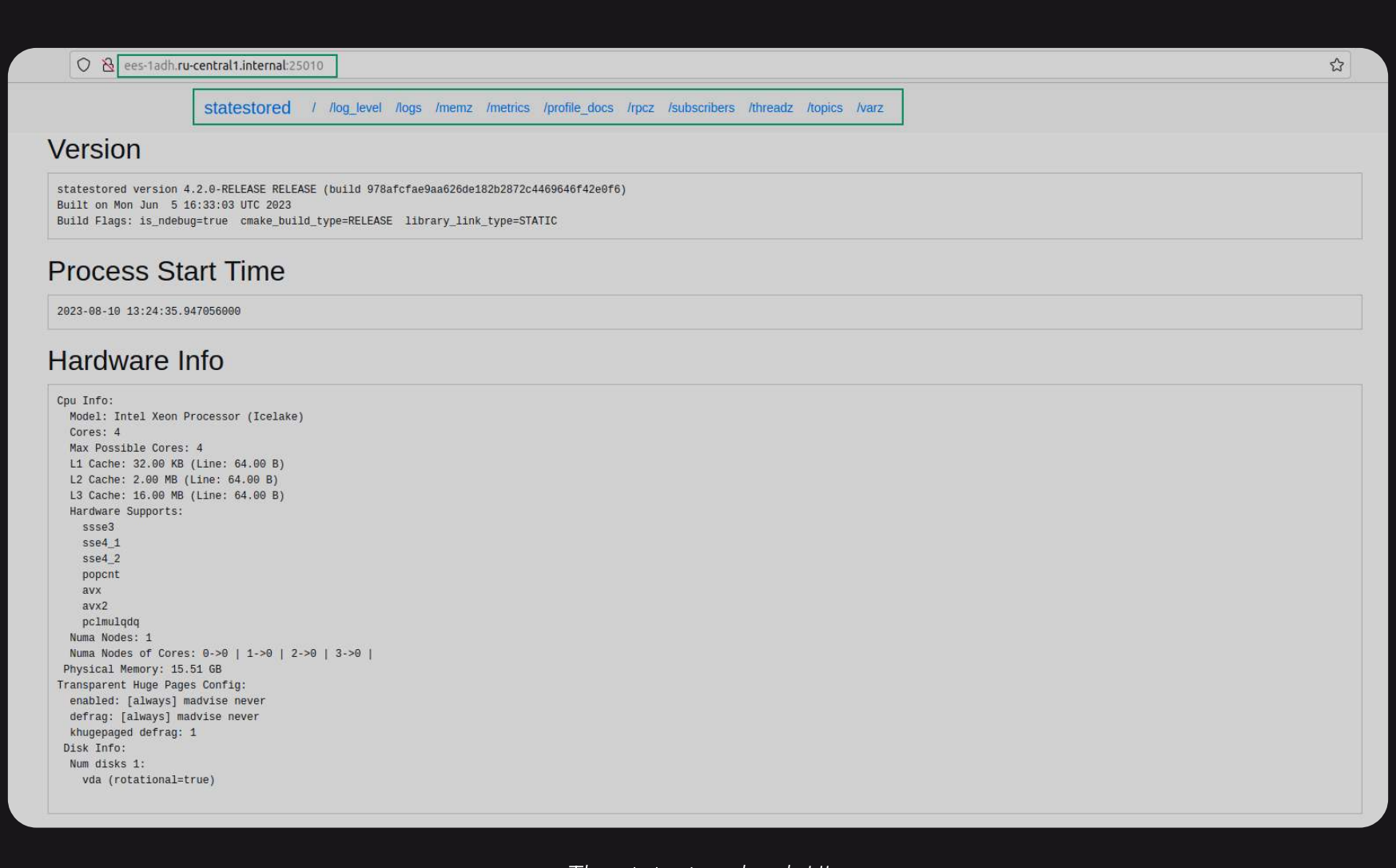
Prometheus metrics

The page contains metrics in the Prometheus format. Prometheus can consume this data for monitoring and alerting.

The menu does not contain the `/metrics_prometheus` option. To display this page, add `/metrics_prometheus` to the HTTP address in the browser.

Impala Statestore (statestored) web UI

The statestored web UI includes information about memory usage, configuration settings, and ongoing health checks. Since there is only a single instance of the statestored within a cluster, you can access the statestored web UI only on a host with the Impala Statestore component.



Version

```
statestored version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6)
Built on Mon Jun  5 16:33:03 UTC 2023
Build Flags: is_ndebug=true  cmake_build_type=RELEASE  library_link_type=STATIC
```

Process Start Time

```
2023-08-10 13:24:35.947056000
```

Hardware Info

```
Cpu Info:
  Model: Intel Xeon Processor (Icelake)
  Cores: 4
  Max Possible Cores: 4
  L1 Cache: 32.00 KB (Line: 64.00 B)
  L2 Cache: 2.00 MB (Line: 64.00 B)
  L3 Cache: 16.00 MB (Line: 64.00 B)
  Hardware Supports:
    ssse3
    sse4_1
    sse4_2
    popcnt
    avx
    avx2
    pclmulqdq
  Numa Nodes: 1
  Numa Nodes of Cores: 0->0 | 1->0 | 2->0 | 3->0 |
 Physical Memory: 15.51 GB
Transparent Huge Pages Config:
  enabled: [always] madvise never
  defrag: [always] madvise never
  khugepaged defrag: 1
 Disk Info:
  Num disks 1:
    vda (rotational=true)
```

The statestored web UI

The main statestored web UI page contains the following information about statestored:

- ***Version*** — the version of the daemon and information about when Impala was built and what build flags were used.
- ***Process start time*** — the time when the daemon was started.
- ***Hardware info*** — information about the host hardware.
- ***OS information*** — information about the operating system that is installed on the host.
- ***Process information*** — information about the daemon process.
- ***CGroup information*** — information about CGroup settings.

log_level

This page displays the current Java and backend log levels and allows you to change the log levels dynamically without restarting the daemon.

logs

This page shows the latest entries of the *statestored.INFO* log file, including the `info`, `warning`, and `error` logs. You can see the details of the most recent operations to determine whether the operations succeeded or encountered errors.

This page allows you to avoid searching log files in the file system.

memz

This page displays the summary and detailed information about memory usage by the statestored daemon.

metrics

The page displays the current set of metrics, counters, and flags representing various aspects of the daemon internal operations.

profile_docs

The page lists counters of the `PROFILE` statement.

rpcz

The page displays information about the duration of RPC communications of the current daemon with other Impala daemons.

subscribers

The page displays information about Impala daemons that registered with the statestored to receive and send updates.

threadz

The page displays information about the threads used by the current daemon and shows which categories they are grouped into.

topics

The page displays information about the metadata topics to which the other Impala daemons subscribed to receive updates.

varz

The page shows the configuration settings that are used when this daemon communicates with other Hadoop components such as HDFS and YARN. The settings are collected from configuration files.

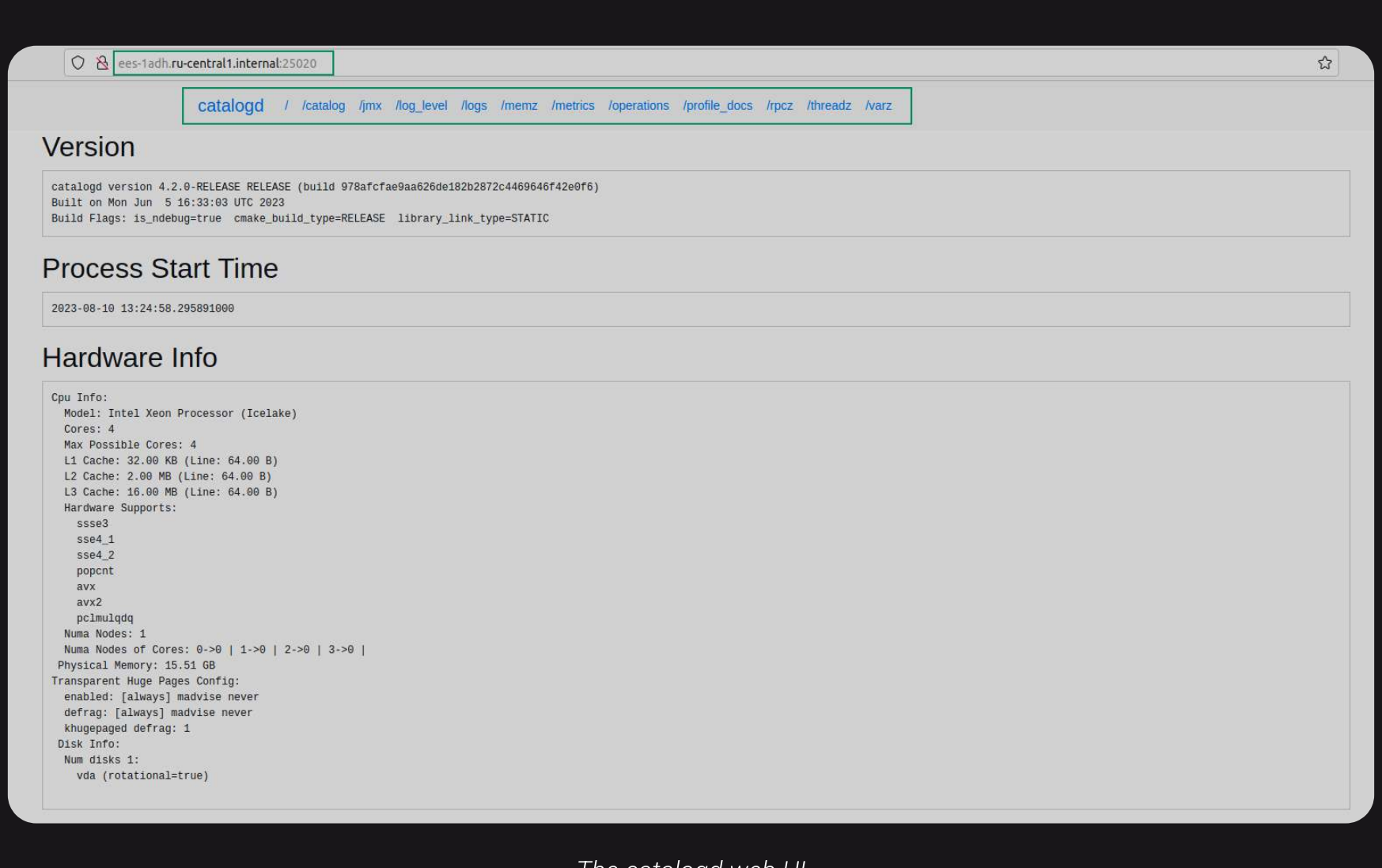
Prometheus metrics

The page contains metrics in the Prometheus format. Prometheus can consume this data for monitoring and alerting.

The menu does not contain the `/metrics_prometheus` option. To display this page, add `/metrics_prometheus` to the HTTP address in the browser.

Impala Catalog Service (catalogd) web UI

The catalogd web UI includes information about the databases, tables, and other objects managed by Impala, in addition to the resource usage and configuration settings of the catalogd. Since there is only a single instance of the catalogd within a cluster, you can access the catalogd web UI only on a host with the Impala Catalog Service.



Version

```
catalogd version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6)
Built on Mon Jun  5 16:33:03 UTC 2023
Build Flags: is_ndebug=true  cmake_build_type=RELEASE  library_link_type=STATIC
```

Process Start Time

```
2023-08-10 13:24:58.295891000
```

Hardware Info

```
Cpu Info:
  Model: Intel Xeon Processor (Icelake)
  Cores: 4
  Max Possible Cores: 4
  L1 Cache: 32.00 KB (Line: 64.00 B)
  L2 Cache: 2.00 MB (Line: 64.00 B)
  L3 Cache: 16.00 MB (Line: 64.00 B)
  Hardware Supports:
    ssse3
    sse4_1
    sse4_2
    popcnt
    avx
    avx2
    pclmulqdq
  Numa Nodes: 1
  Numa Nodes of Cores: 0->0 | 1->0 | 2->0 | 3->0 |
 Physical Memory: 15.51 GB
Transparent Huge Pages Config:
  enabled: [always] madvise never
  defrag: [always] madvise never
  khugepaged defrag: 1
 Disk Info:
  Num disks 1:
    vda (rotational=true)
```

The catalogd web UI

The main catalogd web UI page contains the following information about catalogd:

- ***Version*** — the version of the daemon and information about when Impala was built and what build flags were used.
- ***Process start time*** — the time when the daemon was started.
- ***Hardware info*** — information about the host hardware.
- ***OS information*** — information about the operating system that is installed on the host.
- ***Process information*** — information about the daemon process.
- ***CGroup information*** — information about CGroup settings.

catalog

The ***Catalog*** page displays a list of databases and associated tables. You can use this page to determine which database includes a particular table, check the spelling of a database or table name, search for the same table names in multiple databases. The main use case in debugging would be to check if the impalad instance knows about a table from a particular database.

In addition to impalad ***catalog*** page, the page contains lists of tables with the highest memory requirements, the highest number of metadata operations, the most numbers of files, and with the longest metadata loading time.

jmx

This page displays monitoring information about various JVM subsystems in the JSON format, for example, memory pools, thread management, runtime, and others.

log_level

This page displays the current Java and backend log levels and allows you to change the log levels dynamically without restarting the daemon.

logs

This page shows the latest entries of the *catalogd.INFO* log file, including the `info`, `warning`, and `error` logs. You can see the details of the most recent operations to determine whether the operations succeeded or encountered errors.

This page allows you to avoid searching log files in the file system.

memz

This page displays the summary and detailed information about memory usage by the catalogd daemon.

metrics

The page displays the current set of metrics, counters, and flags representing various aspects of the daemon internal operations.

operations

The page displays information about catalogd operations.

profile_docs

The page lists counters of the `PROFILE` statement.

rpcz

The page displays information about the duration of RPC communications of the current daemon with other Impala daemons.

threadz

The page displays information about the threads used by the current daemon and shows which categories they are grouped into.

varz

The page shows the configuration settings that are used when this daemon communicates with other Hadoop components such as HDFS and YARN. The settings are collected from configuration files.

Prometheus metrics

The page contains metrics in the Prometheus format. Prometheus can consume this data for monitoring and alerting.

The menu does not contain the `/metrics_prometheus` option. To display this page, add `/metrics_prometheus` to the HTTP address in the browser.

Performance tuning

Eugenia Kuzina

Contents

To Table of Contents

- Statements for performance tuning
  - EXPLAIN
  - SUMMARY
  - PROFILE
- The queries page of the impalad UI

Impala provides multiple operators to get query performance information. It also has web UIs that display performance statistics. The most useful is the queries page of the web UI impalad. It contains query plans, `PROFILE` and `SUMMARY` reports, and other information on query performance.

Statements for performance tuning

Impala provides the following statements that can be used for performance tuning:

- EXPLAIN — displays the execution plan for a query.
- SUMMARY — displays a summary report of the last executed query.
- PROFILE — generates a detailed, low-level report for the last query.

EXPLAIN

The EXPLAIN command displays the execution plan that the Impala generates for the specified statement. The command does not execute the query.

The plan includes the low-level mechanisms that Impala will use to read the data. It describes how operations will be distributed among nodes in the cluster, and how intermediate results will be combined to produce the final result set. You can review these details before running the query to check that the query will be executed efficiently. If this plan does not meet your requirements, you can modify the query to improve the performance. For example, you can change conditions in the `WHERE` clause, the order of tables in the `JOIN` statement, add subqueries, change partitioning for a table, collect column statistics and table statistics in Hive (see COMPUTE STATS Statement).

For example, the query `explain select count(*) from table1;` has the following output:

```
Query: explain select count(*) from table1
+------------------------------------------------------------+
| Explain String                                             |
+------------------------------------------------------------+
| Max Per-Host Resource Reservation: Memory=8.00KB Threads=2 |
| Per-Host Resource Estimates: Memory=10MB                   |
| Codegen disabled by planner                                |
|                                                            |
| PLAN-ROOT SINK                                             |
| |                                                          |
| 03:AGGREGATE [FINALIZE]                                    |
| |  output: sum(count(*))                                   |
| |                                                          |
| 02:EXCHANGE [UNPARTITIONED]                                |
| |                                                          |
| 01:AGGREGATE                                               |
| |  output: count(*)                                        |
| |                                                          |
| 00:SCAN HDFS [default.table1]                              |
|    HDFS partitions=1/1 files=5 size=3.99KB                 |
|    row-size=8B cardinality=5                               |
+------------------------------------------------------------+
```

Read the `EXPLAIN` plan from bottom to top:

- The last part (`00:SCAN HDFS [default.table1]`) of the plan shows the low-level details, such as the expected amount of data that will be read. It allows you to estimate the effectiveness of your partitioning strategy and table scan time based on the total data size.
- In the next step, you can see the operations that will be parallelized and performed on each Impala node.
- The next levels show how intermediate result sets are combined and transmitted from one node to another.

You can use the EXPLAIN_LEVEL query option to control the level of detail provided in the `EXPLAIN` output. The allowed values of `EXPLAIN_LEVEL` are:

- `0` or `MINIMAL` — displays a list with one line per operation. It is primarily useful for checking the join order in long queries where the regular `EXPLAIN` output is too long to read easily.
- `1` or `STANDARD` — the default level that shows the logical level of the query execution.
- `2` or `EXTENDED` — includes additional information to illustrate how the query planner uses statistics in decision-making processes.
- `3` or `VERBOSE` — provides the maximum level of detail. It shows how a query is split into fragments that are assembled into a pipeline for each node. This information is useful for low-level performance testing and tuning Impala, not for rewriting SQL code at the user level.

At all `EXPLAIN` detail levels, the plan contains a warning if any tables in the query are missing statistics. Use the COMPUTE STATS statement to collect statistics for each table and suppress this warning.

The `EXPLAIN` plan is also printed at the beginning of the query PROFILE report described below.

SUMMARY

The `SUMMARY` command displays timings of query execution steps. It allows you to determine potential performance problems. You can run `SUMMARY` after a query is executed to see actual performance characteristics of this query.

Example:

```sql
SELECT AVG(salary) FROM table1 WHERE position like 'manager%';

summary;
```

The result:

```
+--------------+--------+--------+----------+----------+-------+------------+----------+---------------+-----------------+
| Operator     | #Hosts | #Inst  | Avg Time | Max Time | #Rows | Est. #Rows | Peak Mem | Est. Peak Mem | Detail          |
+--------------+--------+--------+----------+----------+-------+------------+----------+---------------+-----------------+
| 03:AGGREGATE | 1      | 1      | 1.03ms   | 1.03ms   | 1     | 1          | 48.00 KB | -1 B          | MERGE FINALIZE  |
| 02:EXCHANGE  | 1      | 1      | 0ns      | 0ns      | 1     | 1          | 0 B      | -1 B          | UNPARTITIONED   |
| 01:AGGREGATE | 1      | 1      |30.79ms   | 30.79ms  | 1     | 1          | 80.00 KB | 10.00 MB      |                 |
| 00:SCAN HDFS | 1      | 1      | 5.45s    | 5.45s    | 2.21M | -1         | 64.05 MB | 432.00 MB     | default.table1  |
+--------------+--------+--------+----------+----------+-------+------------+----------+---------------+-----------------+
```

The different stages of the query and their timings are shown with actual and estimated values used in the query `EXPLAIN` plan. The `AVG()` function is computed for a subset of data on each node (stage 01) and then the aggregated results from all nodes are combined at the end (stage 03). You can see which stages took the most time, and whether any estimates were substantially different from the actual values.

Impala provides the MT_DOP query option, which specifies the degree of intra-node parallelism used for certain operations that can benefit from multithreaded execution. If `MT_DOP` is set to a value larger than 0, the `#Inst` column in the output shows the number of fragment instances. Impala decomposes each query into smaller units distributed across the cluster, and these units are called fragments. When the `MT_DOP` query option is set to 0, the `#Inst` column shows the same value as the `#Hosts` column, since there is exactly one fragment for each host.

The `SUMMARY` output is also displayed in the query PROFILE report described below.

PROFILE

The `PROFILE` command generates a detailed, low-level report that shows how the last query was executed. The report contains many parameters. You can find `PROFILE` output fragments below.

```
Query Runtime Profile:
Query (id=36493a18ee92e275:1c04f39000000000):
   - InactiveTotalTime: 0.000ns
   - TotalTime: 0.000ns
  Summary:
    Session ID: ed440ba9e107f951:3ae2c27f67a7d6ae
    Session Type: HIVESERVER2
    HiveServer2 Protocol Version: V6
    Start Time: 2023-08-24 13:23:10.491153000
    End Time: 2023-08-24 13:23:10.604175000
    Query Type: QUERY
    Query State: FINISHED
    Impala Query State: FINISHED
    Query Status: OK
    Impala Version: impalad version 4.2.0-RELEASE RELEASE (build 978afcfae9aa626de182b2872c4469646f42e0f6)
    User: admin
    Connected User: admin
    Delegated User:
    Network Address: 10.92.6.52:47426
    Default Db: default
    Sql Statement: SELECT AVG(field1) FROM table1 WHERE field3 like 'row%'
    Coordinator: ees-1adh.ru-central1.internal:27000
    Query Options (set by configuration): EXPLAIN_LEVEL=VERBOSE,TIMEZONE=UTC,CLIENT_IDENTIFIER=Impala Shell v4.2.0-RELEASE
(978afcf) built on Mon Jun  5 16:33:03 UTC 2023,DEFAULT_FILE_FORMAT=PARQUET
    Query Options (set by configuration and planner):
EXPLAIN_LEVEL=VERBOSE,NUM_NODES=1,NUM_SCANNER_THREADS=1,RUNTIME_FILTER_MODE=OFF,MT_DOP=0,TIMEZONE=UTC,CLIENT_IDENTIFIER=Impala
Shell v4.2.0-RELEASE (978afcf) built on Mon Jun  5 16:33:03 UTC 2023,DEFAULT_FILE_FORMAT=PARQUET,SPOOL_QUERY_RESULTS=0
...

Estimated Per-Host Mem: 16793600
Request Pool: default-pool
Per Host Min Memory Reservation: ees-1adh.ru-central1.internal:27000(16.00 KB)
Per Host Number of Fragment Instances: ees-1adh.ru-central1.internal:27000(1)
Admission result: Admitted immediately
Cluster Memory Admitted: 16.02 MB
Executor Group: empty group (using coordinator only)
ExecSummary:
Operator       #Hosts  #Inst  Avg Time  Max Time  #Rows  Est. #Rows  Peak Mem  Est. Peak Mem  Detail
------------------------------------------------------------------------------------------------------------
F00:ROOT            1      1   0.000ns   0.000ns                             0              0
01:AGGREGATE        1      1   0.000ns   0.000ns      1           1  37.00 KB       16.00 KB  FINALIZE
00:SCAN HDFS        1      1   5.000ms   5.000ms     15           1  85.00 KB       16.00 MB  default.table1
    Errors:
    Query Compilation: 8.502ms
       - Metadata of all 1 tables cached: 1.585ms (1.585ms)
       - Analysis finished: 3.044ms (1.459ms)
       - Authorization finished (noop): 3.203ms (159.571us)
       - Value transfer graph computed: 3.362ms (158.312us)
       - Single node plan created: 6.038ms (2.676ms)
       - Distributed plan created: 6.121ms (83.217us)
       - Lineage info computed: 6.312ms (190.851us)
       - Planning finished: 8.502ms (2.189ms)
    Query Timeline: 113.002ms
       - Query submitted: 0.000ns (0.000ns)
       - Planning finished: 10.000ms (10.000ms)
       - Submit for admission: 10.000ms (0.000ns)
       - Completed admission: 10.000ms (0.000ns)
       - Ready to start on 1 backends: 10.000ms (0.000ns)
       - All 1 execution backends (1 fragment instances) started: 11.000ms (1.000ms)
       - Rows available: 16.000ms (5.000ms)
       - First row fetched: 112.002ms (96.001ms)
       - Last row fetched: 112.002ms (0.000ns)
       - Released admission control resources: 113.002ms (1.000ms)
       - Unregister query: 113.002ms (0.000ns)
     - AdmissionControlTimeSinceLastUpdate: 18.000ms
     - ComputeScanRangeAssignmentTimer: 0.000ns
     - InactiveTotalTime: 0.000ns
     - TotalTime: 0.000ns
...
```

It shows physical query details (the number of bytes read, maximum memory usage, etc.) for each node. You can use this information to determine if the query is I/O-bound or CPU-bound, if some network condition is imposing a bottleneck, or if a slowdown is affecting some nodes but not others.

You can find descriptions of `PROFILE` parameters on the ***/profile_docs*** page in the impalad web UI. Use the following address syntax to access this page: `http://<server-hostname>:<port>/profile_docs`. For example, `http://ees-1adh.ru-central1.internal:25000/profile_docs`.

Time values in the profile output reflect the wall-clock time taken by an operation. For values that represent system time or user time, a measurement system is reflected in the metric name, such as `ScannerThreadsSysTime` or `ScannerThreadsUserTime`. For example, a multi-threaded I/O operation can show a small value for wall-clock time, while the corresponding system time is larger. It represents the sum of the CPU time taken by each thread. A wall-clock time value also can be larger because it includes waiting time, while the corresponding system and user time values only measure the time while an operation actively uses CPU.

Impala displays the `EXPLAIN` and `SUMMARY` outputs at the beginning of the `PROFILE` report to allow you to examine both the logical and physical levels of the query.

The result of the `PROFILE` command contains the `Per Node Profiles` section. It includes the following metrics that can be controlled by the RESOURCE_TRACE_RATIO query option:

- **CpuIoWaitPercentage** — the percentage of time that the CPU (or CPUs) were idle during which the system had pending disk I/O requests.
- **CpuSysPercentage** — the percentage of CPU used by the system.
- **CpuUserPercentage** — the percentage of CPU used by a user.
- **HostDiskReadThroughput** — data read by the host as part of the execution of this query, by the HDFS data node, and by other processes running on the same system.
- **HostDiskWriteThroughput** — data written by the host as part of the execution of this query, by the HDFS data node, and by other processes running on the same system.
- **HostNetworkRx** — data received by the host as part of the execution of this query, other queries, and other processes running on the same system.
- **HostNetworkTx** — data transmitted by the host as part of the execution of this query, other queries, and other processes running on the same system.

At present, the ***/profile_docs*** page does not include these metrics.

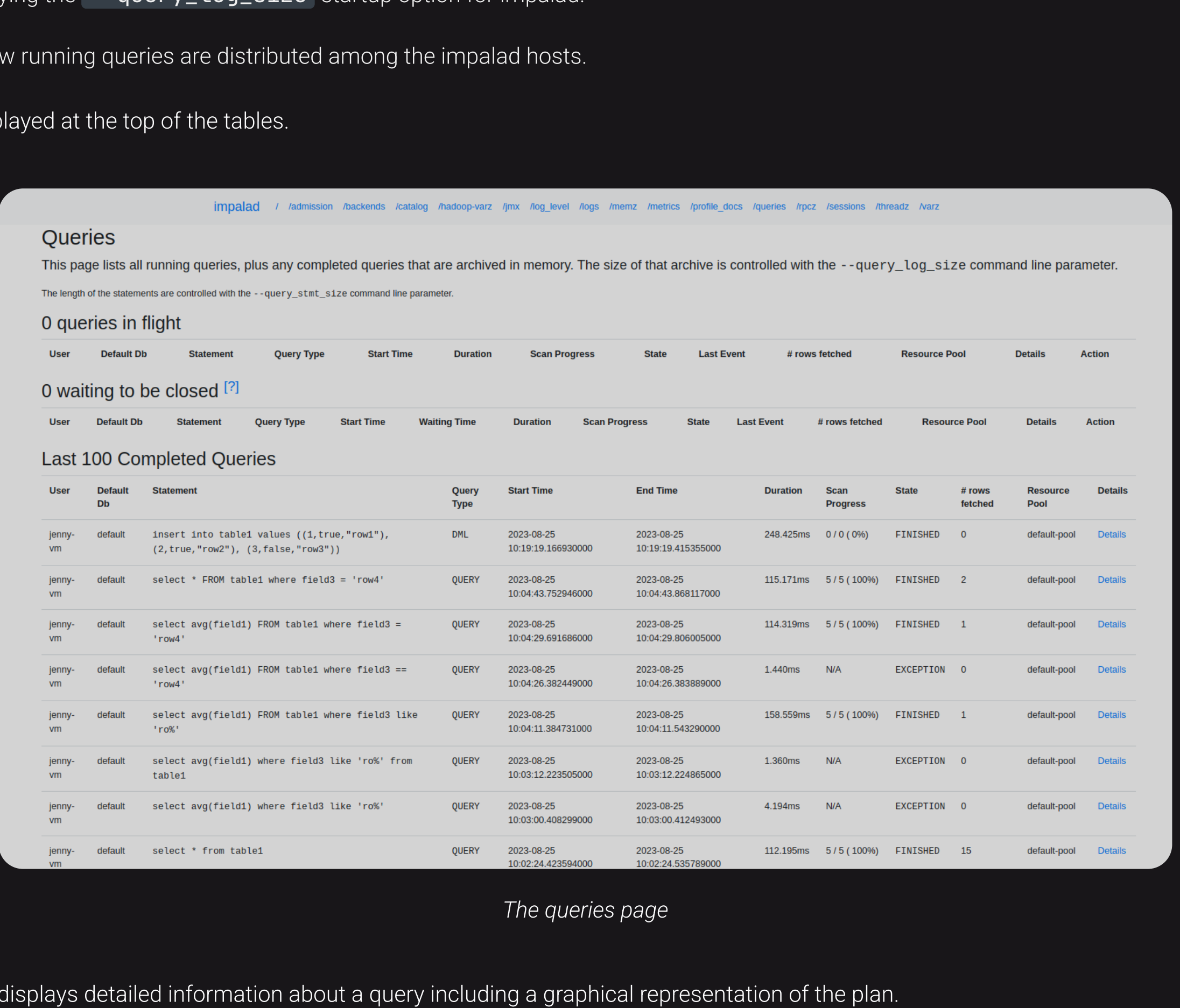
The queries page of the impalad UI

The ***queries*** page collects statistics of queries. Use the following address syntax to access this page: `http://<server-hostname>:<port>/queries`. For example, `http://ees-1adh.ru-central1.internal:25000/queries`.

The ***queries*** page groups queries in several tables:

- ***Queries in flight*** — running queries.
- ***Waiting to be closed*** — queries that have completed their execution, but have not been closed yet.
- ***Last 100 completed queries*** — the latest completed queries. You can control the number of queries and therefore the amount of memory allocated to store information about completed queries, by specifying the `--query_log_size` startup option for impalad.
- ***Query Locations*** — shows how running queries are distributed among the impalad hosts.

The most recent queries are displayed at the top of the tables.



Queries

This page lists all running queries, plus any completed queries that are archived in memory. The size of that archive is controlled with the `--query_log_size` command line parameter.

The length of the statements are controlled with the `--query_stmt_size` command line parameter.

0 queries in flight

User	Default Db	Statement	Query Type	Start Time	Duration	Scan Progress	State	Last Event	# rows fetched	Resource Pool	Details	Action

0 waiting to be closed [?]

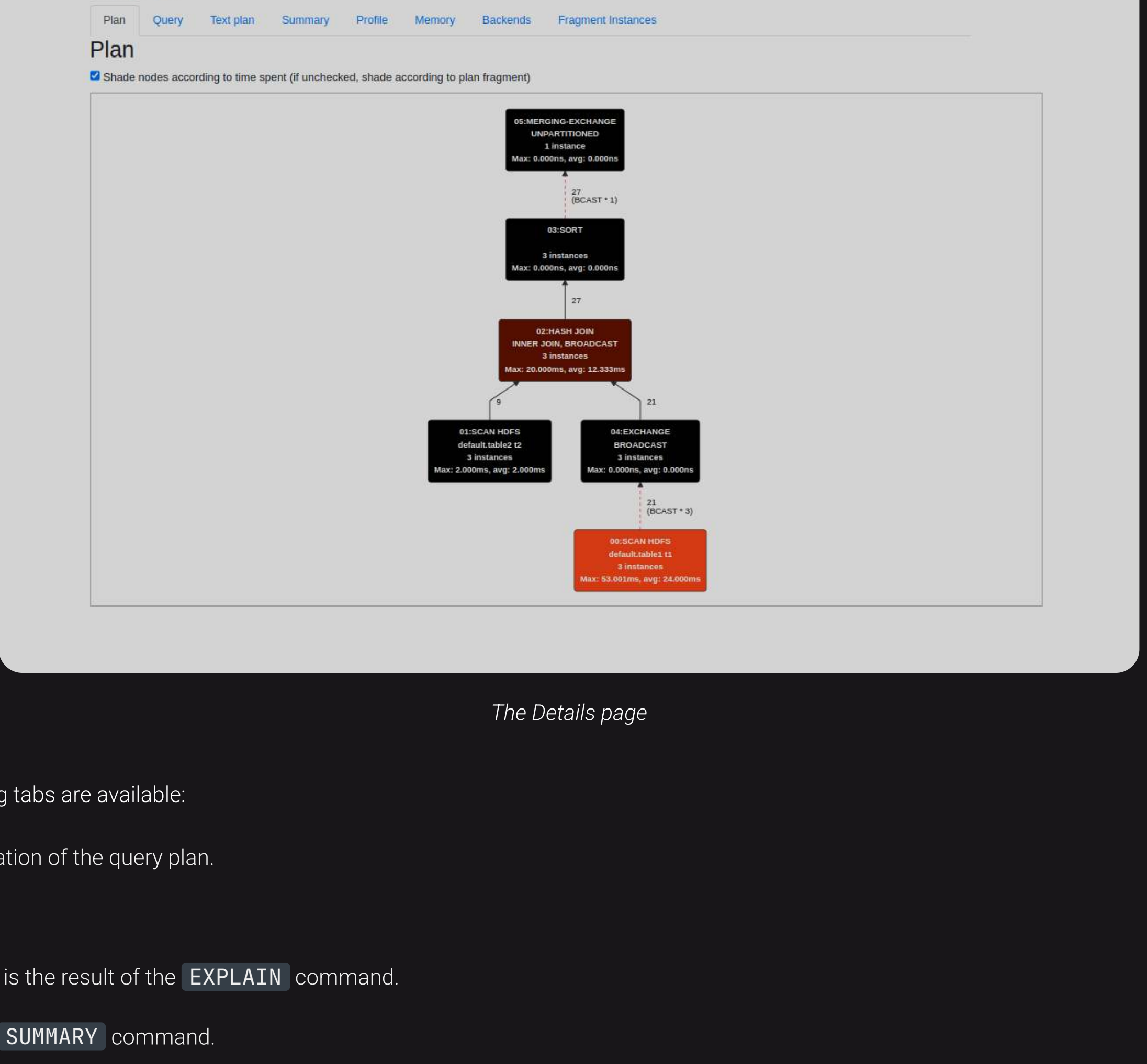
User	Default Db	Statement	Query Type	Start Time	Waiting Time	Duration	Scan Progress	State	Last Event	# rows fetched	Resource Pool	Details	Action

Last 100 Completed Queries

User	Default Db	Statement	Query Type	Start Time	End Time	Duration	Scan Progress	State	# rows fetched	Resource Pool	Details
jenny-vm	default	insert into table1 values ((1,true,"row1"), (2,true,"row2"), (3,false,"row3"))	DML	2023-08-25 10:19:19.166930000	2023-08-25 10:19:19.415355000	248.425ms	0 / 0 (0%)	FINISHED	0	default-pool	Details
jenny-vm	default	select * FROM table1 where field3 = 'row4'	QUERY	2023-08-25 10:04:43.752946000	2023-08-25 10:04:43.868117000	115.171ms	5 / 5 (100%)	FINISHED	2	default-pool	Details
jenny-vm	default	select avg(field1) FROM table1 where field3 = 'row4'	QUERY	2023-08-25 10:04:29.691686000	2023-08-25 10:04:29.806005000	114.319ms	5 / 5 (100%)	FINISHED	1	default-pool	Details
jenny-vm	default	select avg(field1) FROM table1 where field3 == 'row4'	QUERY	2023-08-25 10:04:26.382449000	2023-08-25 10:04:26.383889000	1.440ms	N/A	EXCEPTION	0	default-pool	Details
jenny-vm	default	select avg(field1) FROM table1 where field3 like 'ro%'	QUERY	2023-08-25 10:04:11.384731000	2023-08-25 10:04:11.543290000	158.559ms	5 / 5 (100%)	FINISHED	1	default-pool	Details
jenny-vm	default	select avg(field1) where field3 like 'ro%' from table1	QUERY	2023-08-25 10:03:12.223505000	2023-08-25 10:03:12.224865000	1.360ms	N/A	EXCEPTION	0	default-pool	Details
jenny-vm	default	select avg(field1) where field3 like 'ro%'	QUERY	2023-08-25 10:03:00.408299000	2023-08-25 10:03:00.412493000	4.194ms	N/A	EXCEPTION	0	default-pool	Details
jenny-vm	default	select * from table1	QUERY	2023-08-25 10:02:24.423594000	2023-08-25 10:02:24.535789000	112.195ms	5 / 5 (100%)	FINISHED	15	default-pool	Details

The queries page

The ***Details*** page for each query displays detailed information about a query including a graphical representation of the plan.



impalad
/
/admission
/backends
/catalog
/hadoop-varz
/jmx
/log_level
/logs
/memz
/metrics
/profile_docs
/queries
/rpcz
/sessions
/threadz
/varz
Plan
Query
Text plan
Summary
Profile
Memory
Backends
Fragment Instances
05:MERGING-EXCHANGE UNPARTITIONED 1 instance Max: 0.000ns, avg: 0.000ns
27 (BCAST * 1)
03:SORT 3 instances Max: 0.000ns, avg: 0.000ns
27
02:HASH JOIN INNER JOIN, BROADCAST 3 instances Max: 20.000ms, avg: 12.333ms
9
21
01:SCAN HDFS default.table2 t2 3 instances Max: 2.000ms, avg: 2.000ms
04:EXCHANGE BROADCAST 3 instances Max: 0.000ns, avg: 0.000ns
21 (BCAST * 3)
00:SCAN HDFS default.table1 t1 3 instances Max: 53.001ms, avg: 24.000ms

Query f64208f0cb1d7176:66b3bfac00000000

Status: OK

Plan

☑ Shade nodes according to time spent (if unchecked, shade according to plan fragment)

The Details page

On the ***Details*** page, the following tabs are available:

- ***Plan*** — a graphical representation of the query plan.
- ***Query*** — query text.
- ***Text plan*** — a query plan that is the result of the `EXPLAIN` command.
- ***Summary*** — the result of the `SUMMARY` command.
- ***Profile*** — the result of the `PROFILE` command.
- ***Memory*** — memory consumption. It is available during the query execution.
- ***Backends*** — used Impala instances. It is available during the query execution.
- ***Fragment instances*** — information about fragment instances. It is available during the query execution.

On the ***Profile*** tab, you can export the query profile output to the Thrift, JSON, or text format.

Impala with Hive

Eugenia Kuzina

Contents

To Table of Contents

- Use the same table in Hive and Impala
- Collect table and column statistics for query optimization
  - Table statistics
  - Column statistics

Impala can use existing Apache Hive infrastructure to perform long-running, batch-oriented SQL queries. Impala stores table definitions in MySQL or PostgreSQL database known as Metastore, the same database where Hive keeps this type of data. Impala can access tables defined or loaded by Hive, if all columns use Impala-supported data types, file formats, and compression codecs.

Use the same table in Hive and Impala

As described above, you can work with the same table in Hive and Impala. For example, create a table in Hive (see Work with Hive tables). The Hive shell is available on the host where the *Hive HiveServer2* component is installed.

```
$ sudo -u hive hive
```

```
CREATE TABLE employees (
employee_id INT,
full_name STRING,
hours INT,
rate  INT)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ',';

INSERT INTO TABLE
    employees
VALUES
     (1, 'James Wilson', 8, 500),
    (2, 'Mary Burton', 6, 450),
    (3, 'Patricia Robinson', 7, 430);
```

After that you can utilize impala-shell to select data:

```
$ impala-shell
```

```
SELECT * FROM employees;
```

The result:

```
Query: select * from employees
Query submitted at: 2023-09-05 14:32:38 (Coordinator: http://ees-1adh.ru-central1.internal:25000)
Query progress can be monitored at: http://ees-1adh.ru-central1.internal:25000/query_plan?
query_id=dd477956d9bde7ca:61358fec00000000
+-------------+-------------------+-------+------+
| employee_id | full_name         | hours | rate |
+-------------+-------------------+-------+------+
| 1           | James Wilson      | 8     | 500  |
| 2           | Mary Burton       | 6     | 450  |
| 3           | Patricia Robinson | 7     | 430  |
+-------------+-------------------+-------+------+
Fetched 3 row(s) in 0.21s
```

Add rows to the table in impala-shell:

```
insert into employees values ((4,"Robert Gray", 8, 480), (5,"Elizabeth Tucker", 7, 470));
```

Use the Hive shell to display data:

```
$ sudo -u hive hive
```

```
SELECT * FROM employees;
```

The result:

```
OK
1   James Wilson    8   500
2   Mary Burton 6   450
3   Patricia Robinson   7   430
4   Robert Gray 8   480
5   Elizabeth Tucker    7   470
Time taken: 0.23 seconds, Fetched: 5 row(s)
```

Rows added through Impala are displayed.

The initial focus on query features and performance means that Impala can read more types of data with the `SELECT` statement than it can write with the `INSERT` statement. To query data using the Avro, RCFile, or SequenceFile file formats, you should load the data using Hive.

Collect table and column statistics for query optimization

Impala can do better optimization for complex or multi-table queries when it has statistics about the amount of data and data distribution. For example, the optimization of `JOIN` queries requires statistics to determine which table is larger. It can be done by comparing the number of rows and the average row size of each table.

The Impala query optimizer can use table statistics and column statistics. Originally, you gathered this information with the `ANALYZE TABLE` statement in Hive. In Impala, use the COMPUTE STATS statement instead. `COMPUTE STATS` requires less setup, is more reliable, and does not require switching back and forth between impala-shell and the Hive shell.

Table statistics

The Impala query planner uses statistics of entire tables and partitions. This information includes physical characteristics: a number of rows, a number of data files, a total size of data files, and a file format. For partitioned tables, statistics are calculated for each partition and for the entire table.

The statistics metadata is stored in the Metastore database, and can be updated by Impala or Hive. If a value is not available, `-1` is used as a placeholder. Some values, for example, the number and total sizes of data files, are always kept up to date because they can be calculated cheaply as part of collecting HDFS metadata.

The example below uses the SHOW TABLE STATS statement to display table statistics. For partitioned tables you can also use the `SHOW PARTITIONS` statement that displays exactly the same information as `SHOW TABLE STATS`.

```
show table stats test_table1;
```

The result:

```
+-------+--------+---------+--------------+-------------------+---------+-------------------+---------------------------------------
--------+
| #Rows | #Files |  Size   | Bytes Cached | Cache Replication |  Format | Incremental stats | Location
|
+-------+--------+---------+--------------+-------------------+---------+-------------------+---------------------------------------
--------+
| -1    |   52   | 12.62GB | NOT CACHED   | NOT CACHED        | PARQUET | false             |
hdfs://adh/apps/hive/warehouse/test_table1 |
```

The values for the number and sizes of files are shown. The number of rows is not known and is displayed as `-1`, because it requires a potentially expensive scan through the entire table.

Execute the `COMPUTE STATS` statement and check the table statistics:

```
compute stats test_table1;

show table stats test_table1;
```

The result:

```
+-------------+--------+---------+--------------+-------------------+---------+-------------------+--------------------------------
--------------+
|    #Rows    | #Files |  Size   | Bytes Cached | Cache Replication |  Format | Incremental stats | Location
|
+-------------+--------+---------+--------------+-------------------+---------+-------------------+--------------------------------
--------------+
| 541666666   |   52   | 12.62GB | NOT CACHED   | NOT CACHED        | PARQUET | false             |
hdfs://adh/apps/hive/warehouse/test_table1 |
```

The `COMPUTE STATS` statement adds the missing statistics values.

You can see the `Incremental stats` column in the output. The column shows whether the `Incremental stats` feature is used to collect statistics for the table. `Incremental stats` is a functionality for partitioned tables. When Impala computes incremental statistics for a partitioned table, by default, Impala only processes partitions that do not yet have incremental statistics. This way, you can keep statistics up to date without incurring the overhead of reprocessing the entire table each time. For a particular table, use either `COMPUTE STATS` or `COMPUTE INCREMENTAL STATS`, but never combine the two or alternate between them. If you need to switch from `COMPUTE STATS` to `COMPUTE INCREMENTAL STATS` or vice versa for a table, drop all statistics by running `DROP STATS` before changing the statistics type. For more information, see the ***Generating Table and Column Statistics*** section of the Table and Column Statistics article.

Column statistics

The Impala query planner can utilize statistics of individual columns when the metadata is available in the Metastore database. This technique is most valuable for columns compared across tables in JOIN queries to estimate how many rows the query will retrieve from each table. These statistics are also important for subqueries that use the `EXISTS` and `IN` operators. These operators are internally processed as `JOIN` queries.

The following example displays the column statistics for an unpartitioned Parquet table:

```
show column stats test_table2;
```

The result:

```
+--------+---------+------------------+--------+----------+----------+--------+---------+
| Column | Type    | #Distinct Values | #Nulls | Max Size | Avg Size | #Trues | #Falses |
+--------+---------+------------------+--------+----------+----------+--------+---------+
| field1 | INT     | -1               | -1     |  4       | 4.0      |  -1    |  -1     |
| field2 | BOOLEAN | -1               | -1     |  1       | 1.0      |  -1    |  -1     |
| field3 | STRING  | -1               | -1     | -1       | -1       |  -1    |  -1     |
+--------+---------+------------------+--------+----------+----------+--------+---------+
```

In the output, the maximum and average size values of numeric and fixed-size types are available because they are constants. The number of `Distinct Values` is not determined, since it requires a potentially expensive scan through the entire table, and the value is displayed as `-1`. The same applies to maximum and average sizes of variable-sized types, such as `STRING`. The `COMPUTE STATS` statement adds the missing values and updates statistics.

```
compute stats test_table2;

show column stats test_table2;
```

The result:

```
+--------+---------+------------------+--------+----------+----------+--------+---------+
| Column | Type    | #Distinct Values | #Nulls | Max Size | Avg Size | #Trues | #Falses |
+--------+---------+------------------+--------+----------+----------+--------+---------+
| field1 | INT     | 6                | 0      | 4        | 4.0      | -1     | -1      |
| field2 | BOOLEAN | 2                | 0      | 1        | 1.0      | 14     | 7       |
| field3 | STRING  | 6                | 0      | 4        | 4.0      | -1     | -1      |
+--------+---------+------------------+--------+----------+----------+--------+---------+
```

The `Trues` and `Falses` values are calculated only for fields of the `BOOLEAN` type.

For effective using of column statistics, Impala also needs to have table statistics. When you use the Impala `COMPUTE STATS` statement, both table and column statistics are gathered at the same time. If you run the Hive statement `ANALYZE TABLE COMPUTE STATISTICS FOR COLUMNS`, column statistics are computed. Impala can only use the column statistics if the table is unpartitioned. Impala is not able to utilize Hive-generated column statistics for a partitioned table.

Impala with HBase

Eugenia Kuzina

Contents

To Table of Contents

Example of using an HBase table

Performance considerations

Use cases

Limitations of the Impala and HBase integration

Impala allows you to execute SQL queries on HBase tables. From the perspective of an Impala user who has an RDBMS background, HBase is a key/value store where the value consists of multiple fields. The key is mapped to one column in the Impala table, and the various fields of the value are mapped to other columns in the Impala table. Since Impala and Hive share the same Metastore database, once you create the table in Hive, you can query it through Impala.

Example of using an HBase table

The example below demonstrates how to operate with HBase tables. In this example, the HBase `people_ages` table from the Bulk loading via built-in MapReduce jobs article is used.

1. To use an HBase table in Impala, it is necessary to create an external table in Hive shell.
   Specify the following options:

   - The `STORED BY` clause with the `org.apache.hadoop.hive.hbase.HBaseStorageHandler` storage handler.
   - The `TBLPROPERTIES` clause to map the created table to a corresponding table that exists in HBase. The `hbase.mapred.output.outputtable` property is required if you need to insert data into the table.
   - The type of the column corresponding to the HBase row key as `STRING`.

   ```
   $ sudo -u hive hive
   ```

   ```
   CREATE EXTERNAL TABLE hbase_people_ages (name string, age int)
   STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
   WITH SERDEPROPERTIES ("hbase.columns.mapping" = ":key,basic:age")
   TBLPROPERTIES ("hbase.table.name" = "people_ages", "hbase.mapred.output.outputtable" = "people_ages");
   ```

2. After creating a new table through Hive, execute the `INVALIDATE METADATA` statement in impala-shell to make Impala aware of the new table.

```
$ impala-shell
```

```
INVALIDATE METADATA hbase_people_ages;
```

3. Insert a new row to the table:

```
INSERT INTO hbase_people_ages VALUES ('Gary Morgan', 35), ('Jim Brian', 48);
```

4. Select data from the table:

```
SELECT * from hbase_people_ages WHERE name LIKE 'Gar%';
```

   The result:

```
+------------------+-----+
| name             | age |
+------------------+-----+
| Garcia Nathaniel | 61  |
| Gardner Helena   | 54  |
| Garner Andre     | 37  |
| Garner Juan      | 28  |
| Garner Mark      | 47  |
| Garrett Isabelle | 52  |
| Garrett Steven   | 24  |
| Garrett Willie   | 42  |
| Gary Morgan      | 35  |
| Garza Manuel     | 30  |
| Garza Paul       | 51  |
+------------------+-----+
Fetched 11 row(s) in 5.74s
```

For efficient queries, use the `WHERE` clause to find a single key value or a range of key values. Try to avoid queries that do full-table scans, which are efficient for regular Impala tables but inefficient in HBase.

> **NOTE**
>
> The Impala `INSERT … VALUES` syntax suits HBase tables because inserting a single row is an efficient operation for an HBase table. When you use the `INSERT … SELECT` syntax, HBase only stores the most recent version of each unique row key. If an `INSERT … SELECT` statement copies over multiple rows containing the same key value, the query only returns one row with each key column value. Although Impala does not have an `UPDATE` statement, you can achieve the same result by executing the `INSERT` statement with the same key value.

Performance considerations

Impala uses the HBase client API via Java Native Interface (JNI) to query data stored in HBase. This operation does not read HFiles directly. The extra communication overhead makes it important to choose what data to store in HBase, and construct queries that can retrieve the HBase data efficiently. To do this, consider the following recommendations:

- Use the HBase table for queries that return a single row or a small range of rows, not queries that perform a full table scan of an entire table. If a query references an HBase table and does not contain the `WHERE` clause, that is a strong indicator of an inefficient query.
- HBase may offer acceptable performance for storing small dimension tables. A table should be small enough that executing a full table scan for every query is efficient enough. HDFS is also appropriate for dimension tables that do not need to support update, delete, or insert queries with small numbers of rows.
- Query predicates are applied to row keys as start and stop keys, thereby limiting the scope of a particular search. If a row key field is not mapped to a column type of `STRING`, comparison operations do not work.
- If predicates are used for non-key columns, Impala returns more rows than HBase, which can utilize the `SingleColumnValueFilter` filter. When a row key predicate is applied to a single row, HBase finds and returns that row. When a non-key predicate is used, even if it only applies to a single row, HBase must scan the entire table to find the correct result.
- For performance reasons, it is important to declare a key column as `STRING` in the `CREATE EXTERNAL TABLE` statement. You can also define other columns with the `INT` or `BOOLEAN` types as `STRING` since Impala can optimize this type more effectively in HBase tables. For more information, see the ***Interpreting EXPLAIN Output for HBase Queries*** section of the Using Impala to Query HBase Tables.

Use cases

The following are representative use cases for using Impala to query HBase tables:

- Store rapidly incrementing counters. For example, how many times a web page has been viewed, the number of user connections or votes for a post on social networks. HBase is efficient for capturing this type of data: the append-only storage mechanism is efficient for writing each change to disk, and a query always returns the latest value.
- Store wide tables. Wide tables include many columns (thousands), which contain many attributes of an entity, for example, an online service user. Very often, in these tables, most of the column values are `NULL`, `0`, `FALSE`, empty string, or other blank and placeholder values. For example, a site user may never use some site features, not fill out a special field in the profile, or not visit a certain part of the site. A typical query against this type of table is to retrieve a single row with all the information about a subject, rather than calculating the sum, average, or filtering millions of rows as in Impala-managed tables.

Limitations of the Impala and HBase integration

The Impala integration with HBase has the following limitations and restrictions:

- If you run the `DROP TABLE` command for an Impala-managed table that is mapped to an HBase table, the underlying table is not removed in HBase.
- The `INSERT OVERWRITE` statement is not available for HBase tables. To add new data, or modify an existing row, insert a new row. To modify an existing row, use the same key value.
- If you execute the `CREATE TABLE LIKE` statement for a table mapped to an HBase table, the new table has the same underlying HBase table name. The new table is an alias for the old one, not a new table with an identical column structure.
- Copying data into an HBase table using the Impala `INSERT … SELECT` syntax might produce fewer new rows than are in the `SELECT` query result set. If the result set contains multiple rows with the same value for the key column, each row overwrites a previous row with the same key value. Since the order of the inserted rows is unpredictable, you cannot rely on this technique to preserve the latest version of a particular key value.
- Since the complex data types (`ARRAY`, `STRUCT`, and `MAP`) are currently only supported in Parquet tables, you cannot use these types in HBase tables that are queried through Impala.
- The `LOAD DATA` statement cannot be used with HBase tables.
- The `TABLESAMPLE` clause of the `SELECT` statement cannot be applied to HBase tables.

LDAP authentication in Impala

Sergei Tikhomirov

Contents

To Table of Contents

Enable LDAP

Check

Impala allows you to authenticate in impala-shell via LDAP for the Impala Daemon (impalad) connection.

Enable LDAP

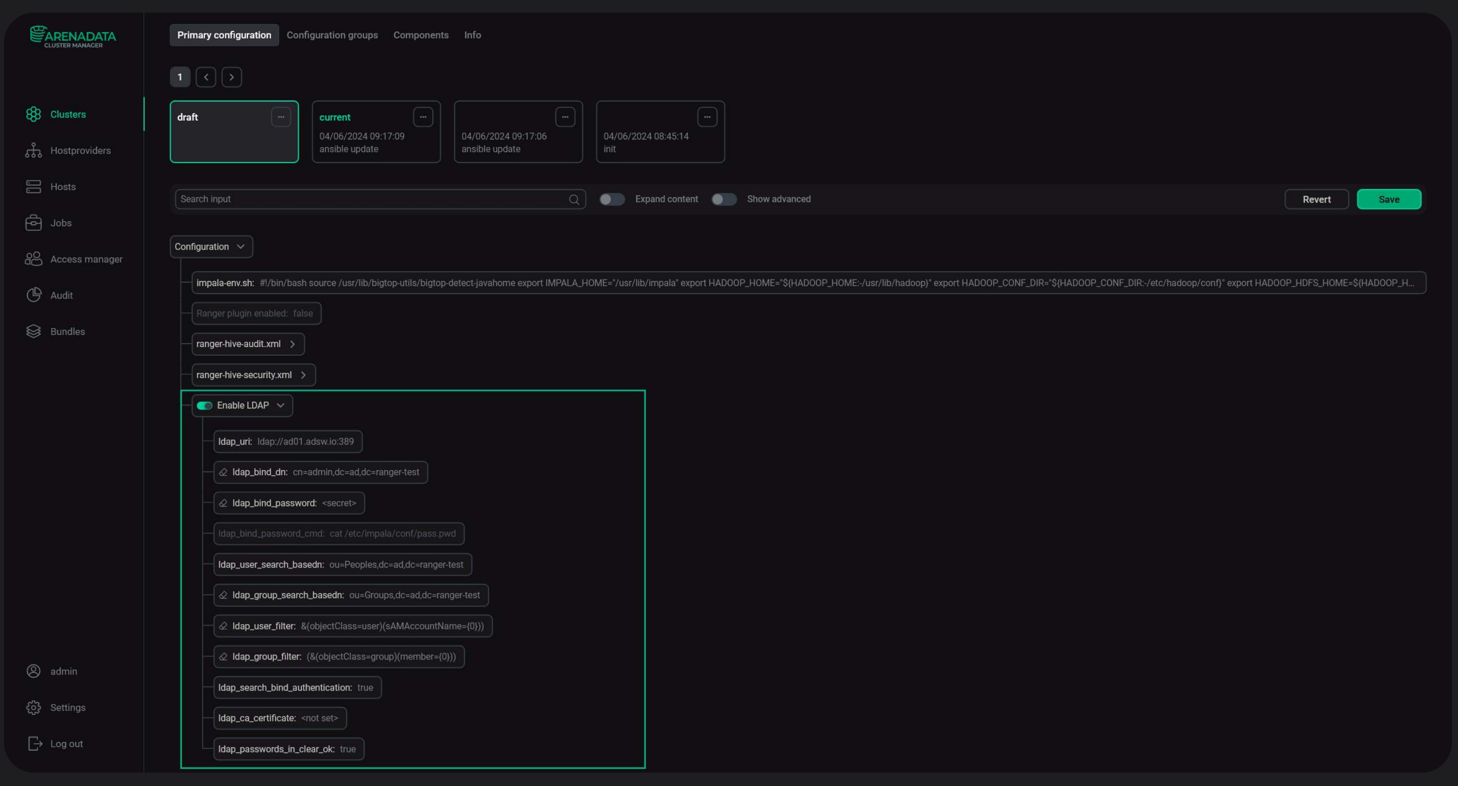
To enable LDAP authentication, follow the steps below:

1. On the ***Clusters*** page, select your ADH cluster and proceed to the ***Services*** tab in the cluster menu.
2. Select the ***Impala*** service and activate the ***Enable LDAP*** parameter group.
3. Fill in the LDAP parameters for Active Directory or FreeIPA and restart Impala.

IMPORTANT

The ***ldap_baseDN***, ***ldap_domain***, and ***ldap_bind_pattern*** parameters are mutually exclusive.

Active Directory



ARENADATA
Primary configuration
Configuration groups
Components
Info
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
draft
current
04/06/2024 09:17:09
ansible update
04/06/2024 09:17:06
ansible update
04/06/2024 08:45:14
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
Ranger plugin enabled: false
ranger-hive-audit.xml
ranger-hive-security.xml
Enable LDAP

impala-env.sh: #!/bin/bash source /usr/lib/bigtop-utils/bigtop-detect-javahome export IMPALA_HOME="/usr/lib/impala"

ldap_uri: ldap://ad01.adsw.io:389

ldap_bind_dn: cn=admin,dc=ad,dc=ranger-test

ldap_bind_password: <secret>

ldap_bind_password_cmd: cat /etc/impala/conf/pass.pwd

ldap_user_search_basedn: ou=Peoples,dc=ad,dc=ranger-test

ldap_group_search_basedn: ou=Groups,dc=ad,dc=ranger-test

ldap_user_filter: &(objectClass=user)(sAMAccountName={0}))

ldap_group_filter: (&(objectClass=group)(member={0}))

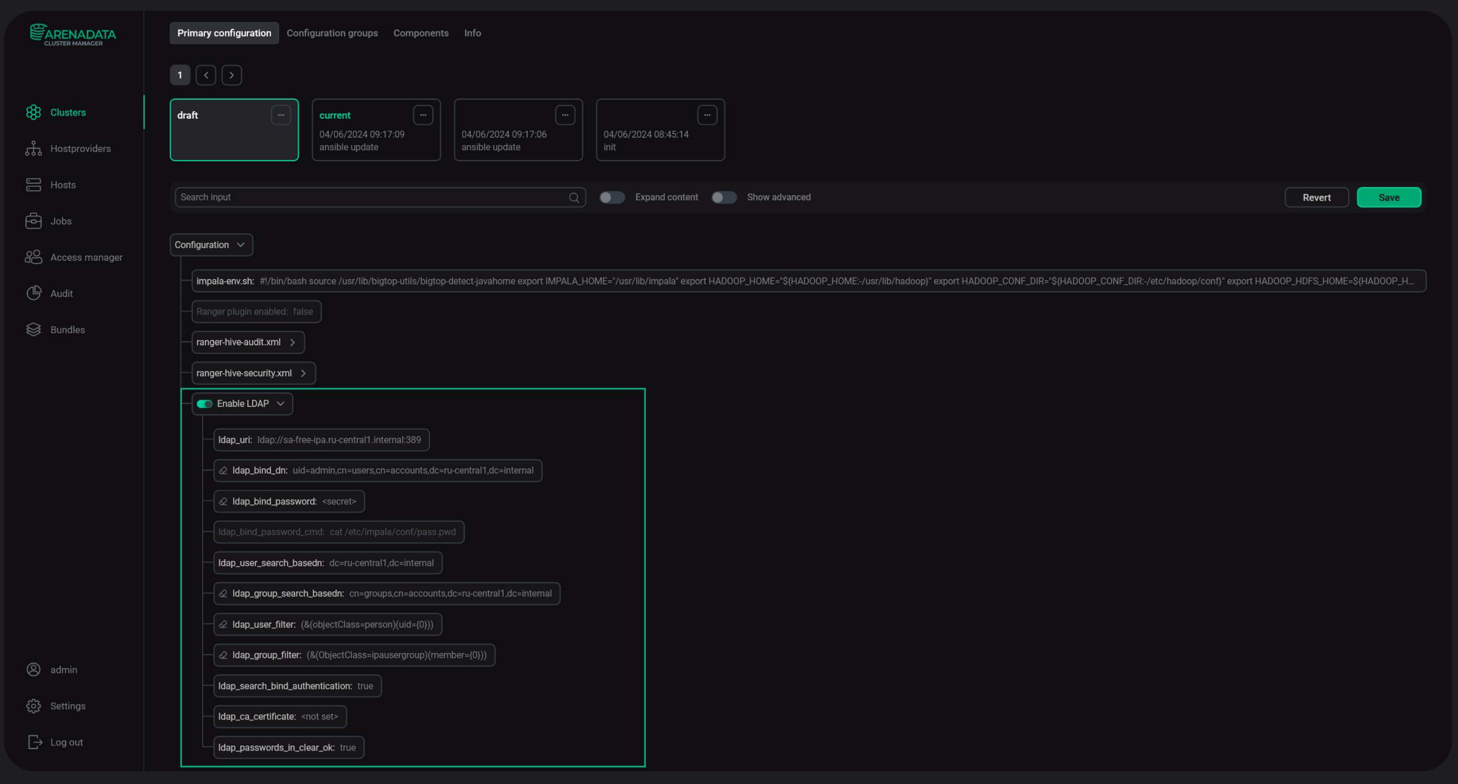
ldap_search_bind_authentication: true

ldap_ca_certificate: <not set>

ldap_passwords_in_clear_ok: true

LDAP AD configuration

FreeIPA



ARENADATA
Primary configuration
Configuration groups
Components
Info
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
draft
current
04/06/2024 09:17:09
ansible update
04/06/2024 09:17:06
ansible update
04/06/2024 08:45:14
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
Ranger plugin enabled: false
ranger-hive-audit.xml
ranger-hive-security.xml
Enable LDAP

impala-env.sh: #!/bin/bash source /usr/lib/bigtop-utils/bigtop-detect-javahome export IMPALA_HOME="/usr/lib/impala"

ldap_uri: ldap://sa-free-ipa.ru-central1.internal:389

ldap_bind_dn: uid=admin,cn=users,cn=accounts,dc=ru-central1,dc=internal

ldap_bind_password: <secret>

ldap_bind_password_cmd: cat /etc/impala/conf/pass.pwd

ldap_user_search_basedn: dc=ru-central1,dc=internal

ldap_group_search_basedn: cn=groups,cn=accounts,dc=ru-central1,dc=internal

ldap_user_filter: (&(objectClass=person)(uid={0}))

ldap_group_filter: (&(ObjectClass=ipausergroup)(member={0}))

ldap_search_bind_authentication: true

ldap_ca_certificate: <not set>

ldap_passwords_in_clear_ok: true

LDAP FreeIPA configuration

Check

Impala-shell requires a configuration file or runtime arguments like username and password. For convenience, you can add the `user` and `ldap_password_cmd` parameters to the configuration file:

```
[impala]
use_ldap=true
auth_creds_ok_in_clear=true
user=<username>
ldap_password_cmd=<pwd_print_command>
```

where:

- `<username>` is a short username;
- `<pwd_print_command>` is a command which prints a password. For example, if you want to store the password in the *impalarc* file, use `ldap_password_cmd=echo -n myPassword`. Alternatively, if you store (or plan to store) the password in a file, use the `ldap_password_cmd=cat /etc/impala/conf/user-pass.pwd` template with any file extension.

Instead of dealing with it manually, you can set the configuration files right in ADCM:

1. In the ***Impala*** service menu, navigate to the ***Components*** tab and click ***Impala Client***.
2. Edit the ***impala-shell-env.sh*** and ***impalarc*** files. The former is responsible for setting the system variables, while the latter contains the configuration.

NOTE

By default, the bundle doesn't save the user password declared in *impalarc*, so it's up to you whether to store a user password and where to store it. The bundle only saves the admin password for LDAP searches and authentication in a file.

To apply the LDAP settings from a config file, use the command below:

```
$ impala-shell --config_file=<path_to_config_file>
```

The config file setting is not necessary as, by default, the file defined in *impala-shell-env.sh* is used. If you want to use a config from a custom location, edit the ***impala-shell-env.sh*** parameter in ADCM by overwriting the `IMPALA_SHELL_GLOBAL_CONFIG_FILE` variable with the path to your custom config. To read about using impala-shell with custom configuration, see the Use impala-shell to connect to Impala article.

Work with Iceberg tables in Impala

Konstantin Alpashkin

Contents

To Table of Contents

- Iceberg catalogs
- DDL commands
  - Create a table
  - CREATE TABLE AS SELECT
  - Delete a table
  - Iceberg V2 tables
  - Schema evolution
  - Create a partitioned table
- DML commands
  - Insert data
  - Delete data
  - Update data
- Time travel
- Roll back
- Iceberg data types mapping

Apache Iceberg is an open, high-performance format for large analytic tables. The ADH Impala service adopts this format allowing you to work with Iceberg tables using SQL and perform analytics over them.

Iceberg catalogs

In Iceberg architecture, a catalog is a named repository of tables and their metadata. Using Impala, you can manage Iceberg tables in the following Iceberg catalogs:

- **HiveCatalog**. The default catalog used by Impala. Uses Hive Metastore to store table metadata. This catalog is used in the examples throughout the article.
- **HadoopCatalog**. A path-based catalog that assumes that there is a catalog location in HDFS where individual Iceberg tables are stored.
- **HadoopTables**. A location-based catalog that assumes that there is a location in HDFS that contains several Iceberg tables.

Also, you can use custom catalogs for existing tables. However, automatic metadata updates do not work for tables in a custom catalog. You have to manually call `REFRESH` on the table when it changes outside Impala.

DDL commands

Create a table

To create an Iceberg table in the default HiveCatalog, append the `STORED AS ICEBERG` clause to the standard Impala `CREATE TABLE` statement.

Example:

```
CREATE TABLE default.impala_ice_test (
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
STORED AS ICEBERG;
```

Run the `DESCRIBE FORMATTED` command to retrieve information about the newly created table:

```
DESCRIBE FORMATTED default.impala_ice_test;
```

Sample output

```
Query: describe formatted impala_ice_test
+------------------------------+-----------------------------------------------------+---------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
------------------------------------+
| name                         | type                                                | comment
|
+------------------------------+-----------------------------------------------------+---------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
------------------------------------+
| # col_name                   | data_type                                           | comment
|
|                              | NULL                                                | NULL
|
| txn_id                       | int                                                 | NULL
|
| acc_id                       | int                                                 | NULL
|
| txn_value                    | double                                              | NULL
|
| txn_date                     | date                                                | NULL
|
|                              | NULL                                                | NULL
|
| # Detailed Table Information | NULL                                                | NULL
|
| Database:                    | default                                             | NULL
|
| OwnerType:                   | USER                                                | NULL
|
| Owner:                       | admin                                               | NULL
|
| CreateTime:                  | Thu Jul 04 10:50:39 UTC 2024                        | NULL
|
| LastAccessTime:              | Sun Jan 25 08:15:21 UTC 1970                        | NULL
|
| Retention:                   | 2147483647                                          | NULL
|
| Location:                    | hdfs://adh/apps/hive/warehouse/impala_ice_test      | NULL
|
| Erasure Coding Policy:       | NONE                                                | NULL
|
| Table Type:                  | EXTERNAL_TABLE                                      | NULL
|
| Table Parameters:            | NULL                                                | NULL
|
|                              | EXTERNAL                                            | TRUE
|
|                              | OBJCAPABILITIES                                     | EXTREAD,EXTWRITE
|
|                              | accessType                                          | 8
|
|                              | current-schema                                      | {\"type\":\"struct\",\"schema-
id\":0,\"fields\":[{\"id\":1,\"name\":\"txn_id\",\"required\":false,\"type\":\"int\"},
{\"id\":2,\"name\":\"acc_id\",\"required\":false,\"type\":\"int\"},
{\"id\":3,\"name\":\"txn_value\",\"required\":false,\"type\":\"double\"},
{\"id\":4,\"name\":\"txn_date\",\"required\":false,\"type\":\"date\"}]} |
|                              | engine.hive.enabled                                | true
|
|                              | external.table.purge                               | TRUE
|
|                              | metadata_location                                   |
hdfs://adh/apps/hive/warehouse/impala_ice_test/metadata/00000-c3406504-fe59-4464-8b1f-01971fc3b949.metadata.json
|
|                              | numFiles                                            | 1
|
|                              | snapshot-count                                      | 0
|
|                              | storage_handler                                     | org.apache.iceberg.mr.hive.
HiveIcebergStorageHandler
|
|                              | table_type                                          | ICEBERG
|
|                              | totalSize                                           | 1914
|
|                              | transient_lastDdlTime                               | 1720090239
|
|                              | uuid                                                | 5495460c-53fe-4fb7-9120-0d5dad7fd022
|
|                              | write.format.default                               | parquet
|
|                              | NULL                                                | NULL
|
| # Storage Information        | NULL                                                | NULL
|
| SerDe Library:               | org.apache.iceberg.mr.hive.HiveIcebergSerDe        | NULL
|
| InputFormat:                 | org.apache.iceberg.mr.hive.HiveIcebergInputFormat  | NULL
|
| OutputFormat:                | org.apache.iceberg.mr.hive.HiveIcebergOutputFormat | NULL
|
| Compressed:                  | No                                                  | NULL
|
| Sort Columns:                | []                                                  | NULL
|
|                              | NULL                                                | NULL
|
| # Constraints                | NULL                                                | NULL
|
+------------------------------+-----------------------------------------------------+---------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
------------------------------------+
```

The `table_type` parameter in the output indicates that the table has been created as an Iceberg table.

By default, Impala creates Iceberg tables using the Parquet data format. To use ORC or AVRO, set the table property `write.format.default` as shown in the following example:

```
CREATE TABLE default.impala_ice_test_orc (
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
STORED AS ICEBERG
TBLPROPERTIES('write.format.default'='ORC');
```

TIP

More Iceberg table properties can be found in the Impala documentation.

CREATE TABLE AS SELECT

You can also use the `CREATE TABLE AS <tbl_name>` (CTAS) syntax to create an Iceberg table based on an existing table.

```
CREATE TABLE default.impala_ice_test_ctas
STORED AS ICEBERG
AS SELECT txn_value, txn_date FROM default.impala_ice_test;
```

Sample DESCRIBE FORMATTED output

```
DESCRIBE FORMATTED default.impala_ice_test_ctas;
+------------------------------+------------------------------------------------------+----------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
----------------------------+
| name                         | type                                                 | comment
|
+------------------------------+------------------------------------------------------+----------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
----------------------------+
| # col_name                   | data_type                                            | comment
|
|                              | NULL                                                 | NULL
|
| txn_value                    | double                                               | NULL
|
| txn_date                     | date                                                 | NULL
|
|                              | NULL                                                 | NULL
|
| # Detailed Table Information | NULL                                                 | NULL
|
| Database:                    | default                                              | NULL
|
| OwnerType:                   | USER                                                 | NULL
|
| Owner:                       | admin                                                | NULL
|
| CreateTime:                  | Thu Jul 04 11:07:31 UTC 2024                         | NULL
|
| LastAccessTime:              | Sun Jan 25 08:32:12 UTC 1970                         | NULL
|
| Retention:                   | 2147483647                                           | NULL
|
| Location:                    | hdfs://adh/apps/hive/warehouse/impala_ice_test_ctas | NULL
|
| Erasure Coding Policy:       | NONE                                                 | NULL
|
| Table Type:                  | EXTERNAL_TABLE                                       | NULL
|
| Table Parameters:            | NULL                                                 | NULL
|
|                              | EXTERNAL                                             | TRUE
|
|                              | OBJCAPABILITIES                                      | EXTREAD,EXTWRITE
|
|                              | accessType                                           | 8
|
|                              | current-schema                                       | {\"type\":\"struct\",\"schema-
id\":0,\"fields\":[{\"id\":1,\"name\":\"txn_value\",\"required\":false,\"type\":\"double\"},
{\"id\":2,\"name\":\"txn_date\",\"required\":false,\"type\":\"date\"}]} |
|                              | engine.hive.enabled                                 | true
|
|                              | external.table.purge                                | TRUE
|
|                              | metadata_location                                    |
hdfs://adh/apps/hive/warehouse/impala_ice_test_ctas/metadata/00000-c3871ac4-ca86-4c6d-be6c-116f3bf7cdf9.metadata.json
|
|                              | numFiles                                             | 1
|
|                              | snapshot-count                                       | 0
|
|                              | storage_handler                                      | org.apache.iceberg.mr.hive.
HiveIcebergStorageHandler
|
|                              | table_type                                           | ICEBERG
|
|                              | totalSize                                            | 1531
|
|                              | transient_lastDdlTime                                | 1720091251
|
|                              | uuid                                                 | 5b792d0b-4e67-4a2b-957c-86e41970c977
|
|                              | write.format.default                                | parquet
|
|                              | NULL                                                 | NULL
|
| # Storage Information        | NULL                                                 | NULL
|
| SerDe Library:               | org.apache.iceberg.mr.hive.HiveIcebergSerDe         | NULL
|
| InputFormat:                 | org.apache.iceberg.mr.hive.HiveIcebergInputFormat   | NULL
|
| OutputFormat:                | org.apache.iceberg.mr.hive.HiveIcebergOutputFormat  | NULL
|
| Compressed:                  | No                                                   | NULL
|
| Sort Columns:                | []                                                   | NULL
|
|                              | NULL                                                 | NULL
|
| # Constraints                | NULL                                                 | NULL
|
+------------------------------+------------------------------------------------------+----------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
----------------------------+
```

Delete a table

You can delete a table using the `DROP TABLE <tbl_name>` command. Example:

```
DROP TABLE default.impala_ice_test;
```

If the table property `external.table.purge` is set to `true`, the command also deletes the data files from HDFS. By default, the `external.table.purge` property is set to `true` when a table is created using the Impala's `CREATE TABLE` command. If the table is created using `CREATE EXTERNAL TABLE` (the table already exists and gets registered in a catalog), this property is set to `false`, so `DROP TABLE` doesn't remove table data, but only removes the table from the catalog.

Iceberg V2 tables

The Iceberg V2 table format supports row-level updates for `DELETE` and `UPDATE` operations using the "merge-on-read" approach. Using this format, instead of rewriting existing data files, Iceberg creates delete files that store information about the deleted records. These files actually contain file paths and positions of the deleted rows (position deletes).

To create an Iceberg table that uses the Iceberg V2 format, specify the `format-version=2` table property when creating a table.

Example:

```
CREATE TABLE default.impala_ice_test_v2 (
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
STORED AS ICEBERG
TBLPROPERTIES('format-version'='2');
```

You can also upgrade an existing Iceberg V1 table to the Iceberg V2 version using `ALTER TABLE`. An example is shown below.

```
ALTER TABLE default.impala_ice_test
SET TBLPROPERTIES('format-version'='2');
```

Schema evolution

Iceberg supports schema evolution that allows reordering, deleting, or changing columns without affecting the readability of old (pre-evolution) data files.

NOTE

Impala does not support schema evolution for AVRO-formatted tables. For more information, see Iceberg documentation.

Below are Iceberg table schema evolution commands supported by Impala.

ALTER TABLE … RENAME TO

The following command renames an Iceberg table.

```
ALTER TABLE default.impala_ice_test
RENAME TO default.impala_ice_test_new;
```

ALTER TABLE … ADD COLUMN

You can add a column to an existing Iceberg table as shown below.

```
ALTER TABLE default.impala_ice_test
ADD COLUMN `comment` string;
```

Sample output:

```
+----------------------------+
| summary                    |
+----------------------------+
| Column(s) have been added. |
+----------------------------+
```

ALTER TABLE … DROP COLUMN

You can delete a column in an existing Iceberg table as shown below.

```
ALTER TABLE default.impala_ice_test
DROP COLUMN `comment`;
```

Sample output:

```
+--------------------------+
| summary                  |
+--------------------------+
| Column has been dropped. |
+--------------------------+
```

ALTER TABLE … CHANGE COLUMN

You can change a column name and the column type, assuming the new type is compatible with the old one.

```
ALTER TABLE default.impala_ice_test
CHANGE txn_id txn_id BIGINT;
```

The output:

```
+--------------------------+
| summary                  |
+--------------------------+
| Column has been altered. |
+--------------------------+
```

Create a partitioned table

With Impala, you can create partitioned Iceberg tables using traditional value-based partitioning. The syntax is straightforward and is shown below:

```
CREATE TABLE default.impala_ice_test_partitioned (
    txn_id int,
    txn_value double,
    txn_date date)
PARTITIONED BY (acc_id int)
STORED AS ICEBERG;
```

Also, you can create a partitioned table using one or more partition transforms. An example is shown below.

```
CREATE TABLE default.impala_ice_test_partitioned_transform (
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
PARTITIONED BY SPEC (
    BUCKET(3, acc_id), ①
    day(txn_date)) ②
STORED AS ICEBERG;
```

① The first part of the transformation expression instructs Iceberg to distribute inserted data among three buckets. For each `acc_id` value to be inserted, Iceberg calculates a hash and gets `mod 3` result (`hash(acc_id) mod 3`) that specifies the destination bucket.

② Then, within each of the 3 buckets, Iceberg creates partitions by date.

Impala creates the following HDFS directory structure for a table created by the above command.

```
[admin@ka-adh-2 ~]$ hdfs dfs -ls -R /apps/hive/warehouse
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01 /apps/hive/warehouse/impala_ice_test_partitioned_transform
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01 /apps/hive/warehouse/impala_ice_test_partitioned_transform/data
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=0
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=0/txn_date_day=2023-01-03
-rw-r--r--   3 impala hadoop       1153 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=0/txn_date_day=2023-01-03/314da83d47eee296-
145c34e600000000_1592985449_data.0.parq
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=1
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=1/txn_date_day=2023-01-02
-rw-r--r--   3 impala hadoop       1153 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=1/txn_date_day=2023-01-02/314da83d47eee296-
145c34e600000000_584411471_data.0.parq
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=2
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=2/txn_date_day=2023-01-01
-rw-r--r--   3 impala hadoop       1153 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/data/acc_id_bucket=2/txn_date_day=2023-01-01/314da83d47eee296-
145c34e600000000_1148089692_data.0.parq
drwxrwxr-x   - impala hadoop          0 2024-07-05 14:01 /apps/hive/warehouse/impala_ice_test_partitioned_transform/metadata
-rw-r--r--   3 impala hadoop       2385 2024-07-05 14:00
/apps/hive/warehouse/impala_ice_test_partitioned_transform/metadata/00000-b04f03f7-f770-44a1-825b-8e84e4392f41.metadata.json
-rw-r--r--   3 impala hadoop       3564 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/metadata/00001-13512042-67f5-4327-933c-978d41be5df1.metadata.json
-rw-r--r--   3 impala hadoop       6539 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/metadata/24a8910e-e616-4d4b-ace5-46987f83fe9a-m0.avro
-rw-r--r--   3 impala hadoop       3804 2024-07-05 14:01
/apps/hive/warehouse/impala_ice_test_partitioned_transform/metadata/snap-4359281668985942129-1-24a8910e-e616-4d4b-ace5-
46987f83fe9a.avro
```

You can view partitions using the `SHOW PARTITIONS` command. A sample output is below.

```
+---------------------------------------------+----------------+-----------------+
| Partition                                   | Number Of Rows | Number Of Files |
+---------------------------------------------+----------------+-----------------+
| {"acc_id_bucket":"0","txn_date_day":"19360"} | 1              | 1               |
| {"acc_id_bucket":"0","txn_date_day":"19361"} | 1              | 1               |
| {"acc_id_bucket":"0","txn_date_day":"19723"} | 1              | 1               |
| {"acc_id_bucket":"0","txn_date_day":"19725"} | 1              | 1               |
| {"acc_id_bucket":"1","txn_date_day":"19359"} | 1              | 1               |
| {"acc_id_bucket":"1","txn_date_day":"19360"} | 1              | 1               |
| {"acc_id_bucket":"2","txn_date_day":"19358"} | 1              | 1               |
| {"acc_id_bucket":"2","txn_date_day":"19361"} | 1              | 1               |
| {"acc_id_bucket":"2","txn_date_day":"19392"} | 1              | 1               |
+---------------------------------------------+----------------+-----------------+
```

Iceberg supports partition evolution that allows changing a table's partitioning spec even without the need of rewriting existing data files. You can change a table's partitioning by using the `ALTER TABLE SET PARTITION SPEC` statement as shown in the example below.

```
ALTER TABLE default.impala_ice_test_partitioned
SET PARTITION SPEC (DAY(txn_date));
```

The output:

```
+-------------------------+
| summary                 |
+-------------------------+
| Updated partition spec. |
+-------------------------+
```

DML commands

Insert data

With Impala, you can write data to an Iceberg table using the `INSERT INTO` and `INSERT OVERWRITE` commands.

Example:

```
INSERT INTO default.impala_ice_test VALUES
(1, 1001, 150.50, '2023-01-04'),
(2, 1002, 200.50, '2023-01-03'),
(3, 1003, 50.00, '2024-01-03');
```

Using `INSERT OVERWRITE`, you can replace data in a table with the result of a query. For example:

```
INSERT OVERWRITE TABLE default.impala_ice_test_tmp
SELECT * from default.impala_ice_test;
```

NOTE

`INSERT OVERWRITE` is not allowed for tables that use the `BUCKET` partition transform.

Delete data

You can delete data from Iceberg V2 tables using the `DELETE` command. Example:

```
DELETE FROM default.impala_ice_test
WHERE txn_id = 1;
```

NOTE

The `DELETE` operation is supported only for Iceberg V2 tables. Attempting to delete rows in a V1 table throws an exception.

Update data

With Impala, you can update Iceberg V2 tables as shown in the example below.

```
UPDATE default.impala_ice_test
SET txn_value = txn_value + 100;
```

NOTE

The `UPDATE` operation is supported only for Iceberg V2 tables. Attempting to update rows in a V1 table throws an exception.

You can also use the `UPDATE FROM` statement to update a target Iceberg table based on a source table (the source can be a non-Iceberg table).

Example:

```
UPDATE default.impala_ice_test
SET default.impala_ice_test.acc_id = src.acc_id,
    default.impala_ice_test.txn_date = src.txn_date,
FROM default.impala_ice_test, default.impala_ice_test_src src
WHERE default.impala_ice_test.txn_id = src.txn_internal_id;
```

If there are multiple matches in the `WHERE` clause, Impala throws an error. There are several limitations on using the `UPDATE FROM` command, like Parquet-only format support, updating partitioned columns issues, and alike. For more information, see Impala documentation.

Time travel

Iceberg tables support the time travel feature that allows you to query data from a specific table snapshot that was created at some point in the past and can be referenced by an ID or timestamp. You can run time travel queries using the `FOR SYSTEM_TIME AS OF <timestamp>` and `FOR SYSTEM_VERSION AS OF <snapshot-id>` clauses. Several usage examples are presented below.

The following example queries data from the closest snapshot that is older than the specified timestamp:

```
SELECT * FROM default.impala_ice_test
FOR SYSTEM_TIME AS OF '2024-07-02 12:00:00';
```

The result:

```
+--------+--------+-----------+------------+
| txn_id | acc_id | txn_value | txn_date   |
+--------+--------+-----------+------------+
| 2      | 1002   | 63.5      | 2024-02-04 |
| 1      | 1001   | 110.0     | 2023-01-01 |
+--------+--------+-----------+------------+
```

The next query identifies the snapshot to work with by subtracting time units from the current time:

```
SELECT * FROM default.impala_ice_test
FOR SYSTEM_TIME AS OF now() - interval 1 minute;
```

The output:

```
+--------+--------+-----------+------------+
| txn_id | acc_id | txn_value | txn_date   |
+--------+--------+-----------+------------+
| 2      | 1002   | 63.5      | 2024-02-04 |
+--------+--------+-----------+------------+
```

The following query retrieves data from a snapshot explicitly identified by ID:

```
SELECT * FROM default.impala_ice_test
FOR SYSTEM_VERSION AS OF 7308000224696874146;
```

The result:

```
+--------+--------+-----------+------------+
| txn_id | acc_id | txn_value | txn_date   |
+--------+--------+-----------+------------+
| 1      | 1001   | 10.0      | 2023-01-01 |
+--------+--------+-----------+------------+
```

To get information about the snapshots available for a given table, use the `DESCRIBE HISTORY` command as shown in the following example:

```
DESCRIBE HISTORY default.impala_ice_test;
```

Sample output:

```
+-------------------------------+---------------------+---------------------+---------------------+
| creation_time                 | snapshot_id         | parent_id           | is_current_ancestor |
+-------------------------------+---------------------+---------------------+---------------------+
| 2024-04-07 21:41:56.959000000 | 7308000224696874146 | NULL                | TRUE                |
| 2024-04-07 21:42:19.537000000 | 3739750708916322181 | 7308000224696874146 | TRUE                |
| 2024-04-07 22:04:58.577000000 | 3634103131974849338 | 3739750708916322181 | TRUE                |
+-------------------------------+---------------------+---------------------+---------------------+
```

Roll back

Whenever an Iceberg table gets modified, Iceberg creates a snapshot for that table. Using snapshots, you can roll back the table state to some version in the past labeled by the snapshot ID. When a rollback is done for a table, a new snapshot is created with the same snapshot ID, but with a new creation timestamp. By using the `ALTER TABLE <tbl_name> EXECUTE ROLLBACK` statement, you can roll back a table to a previous snapshot as shown below.

For example, assume there is an Iceberg table with the following contents:

```
+--------+--------+-----------+------------+
| txn_id | acc_id | txn_value | txn_date   |
+--------+--------+-----------+------------+
| 2      | 1002   | 63.5      | 2024-02-04 |
| 1      | 1001   | 110.0     | 2023-01-01 |
+--------+--------+-----------+------------+
```

The following command rolls back the table state to a specific snapshot using the snapshot ID:

```
ALTER TABLE default.impala_ice_test
EXECUTE ROLLBACK(7308000224696874146);
```

After executing this command, running `SELECT` returns a result set the table used to store at the moment of creation of the specified snapshot. For example:

```
+--------+--------+-----------+------------+
| txn_id | acc_id | txn_value | txn_date   |
+--------+--------+-----------+------------+
| 1      | 1001   | 10.0      | 2023-01-01 |
+--------+--------+-----------+------------+
```

The following command performs a rollback to a snapshot whose creation timestamp is older than the specified timestamp:

```
ALTER TABLE default.impala_ice_test
EXECUTE ROLLBACK('2024-07-05 12:00:00');
```

By default, Iceberg accumulates snapshots until they are deleted by a user command. You can expire unnecessary snapshots by using the `ALTER TABLE … EXECUTE expire_snapshots(<timestamp>)` statement.

Examples:

```
ALTER TABLE default.impala_ice_test
EXECUTE expire_snapshots('2024-07-05 12:00:00');
```

```
ALTER TABLE default.impala_ice_test
EXECUTE expire_snapshots(now() - interval 10 days);
```

Iceberg data types mapping

The following table describes the mapping of Iceberg data types and SQL types in Impala.

Iceberg type	SQL type in Impala
boolean	BOOLEAN
int	INTEGER
long	BIGINT
float	FLOAT
double	DOUBLE
decimal(P, S)	DECIMAL(P, S)
date	DATE
time	—
timestamp	TIMESTAMP
timestamptz	Only read support via TIMESTAMP
string	STRING
uuid	—
fixed(L)	—
binary	—
struct	STRUCT (read only)
list	ARRAY (read only)
map	MAP (read only)

Impala service management via ADCM

Eugenia Kuzina

To Table of Contents

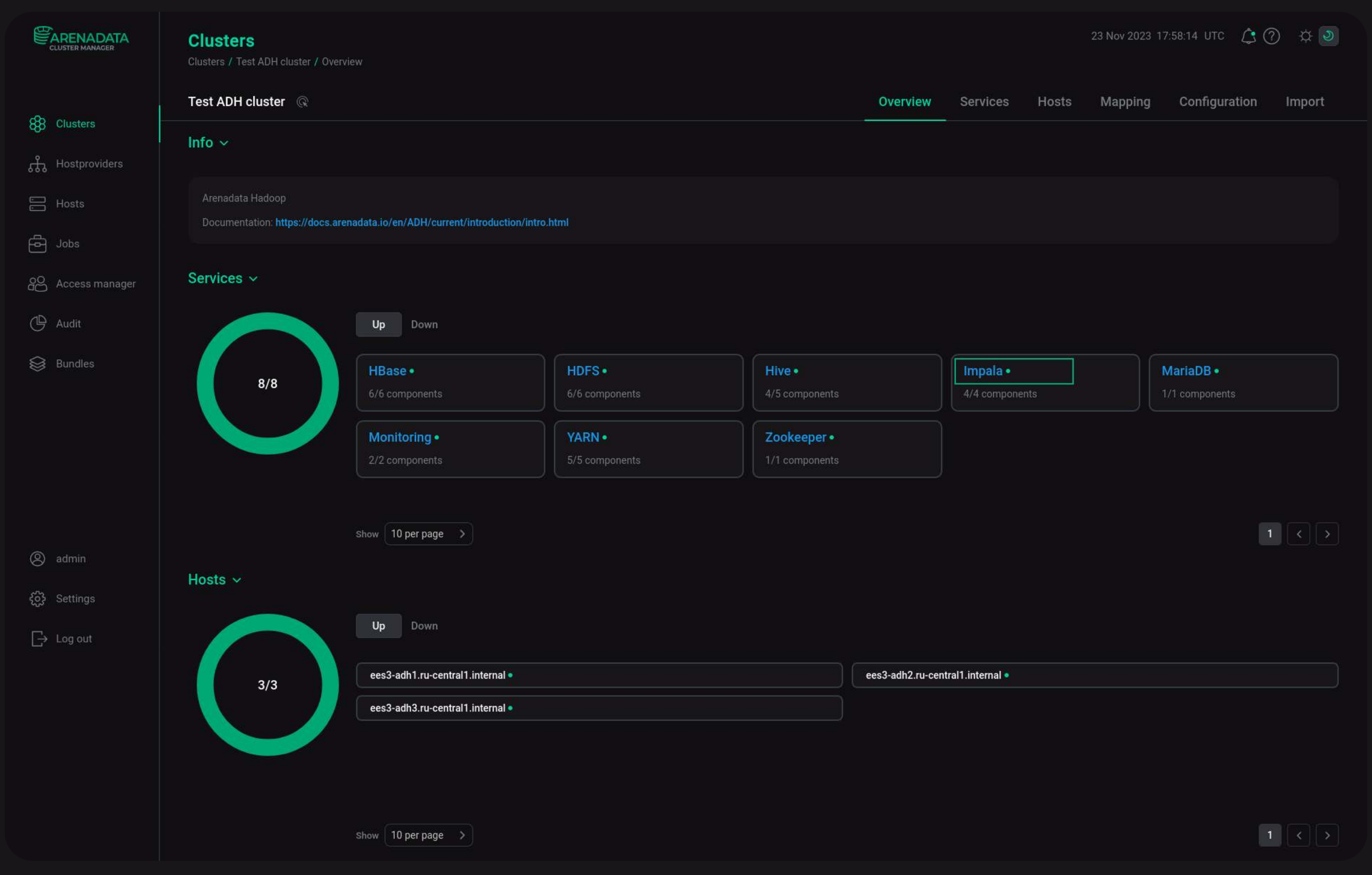
The ADCM UI provides actions to manage the Impala service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Impala service are listed in the table below.

Impala service actions

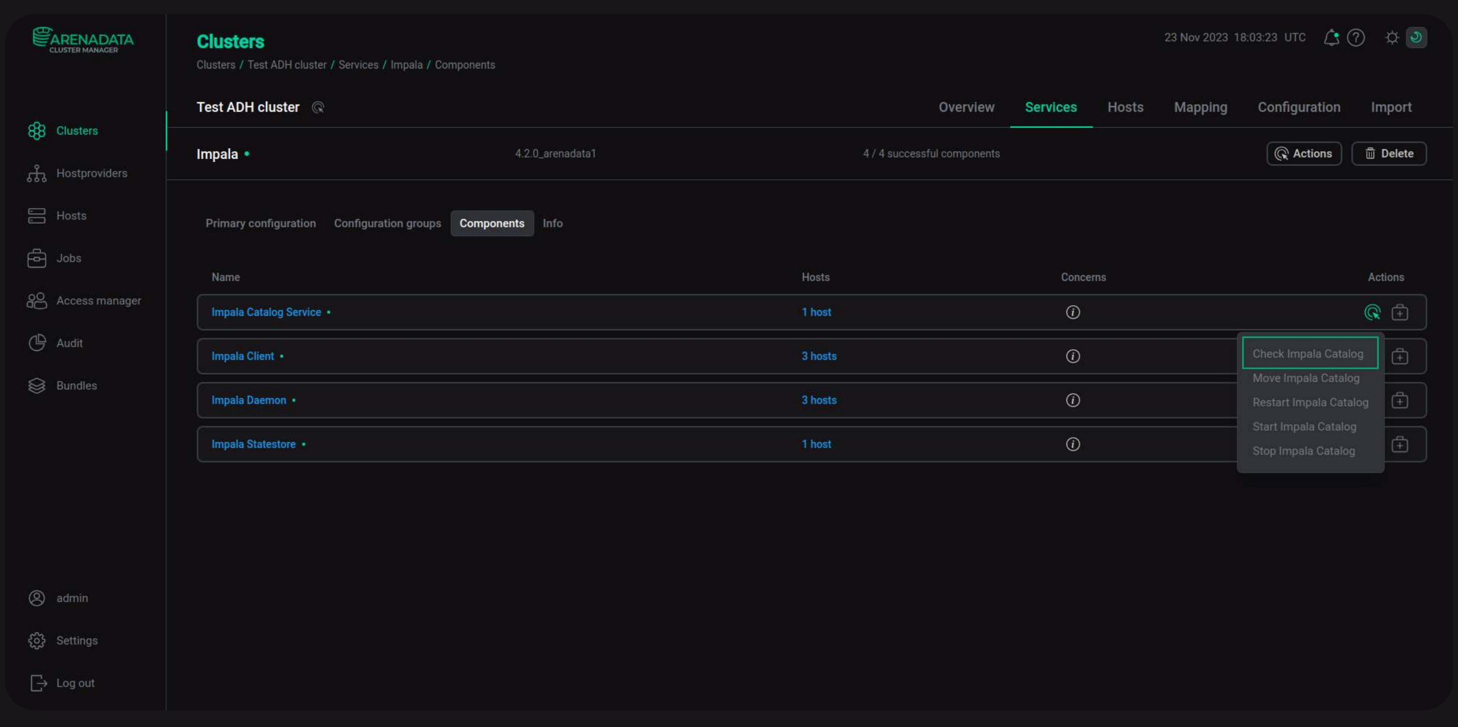
Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Impala components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

Impala components also have actions that operate with corresponding daemons. To execute these actions, click the ***Impala*** service on the ***Services*** cluster tab. Alternatively, click ***Impala*** next to the ***Services*** diagram on the ***Overview*** tab.



The "Overview" cluster tab with "Services" diagram

The ***Components*** tab of the Impala service is displayed. Click the icon ⚙ in the ***Action*** column for a particular component, and select an action from the drop-down list.



Actions of Impala components

The actions available for each component are listed in the tables below.

Impala Daemon actions

Action	Description
Add/Remove Daemon(s)	Allows you to remove or add the Impala Daemon component to the cluster hosts. When you run this action, a window with components and hosts is displayed, and you can redistribute the Impala Daemon component among the hosts as described in the Add components article. The Impala Daemon component can be installed on multiple cluster hosts
Check Impala Daemon	Verifies whether all Impala Daemons in the cluster work correctly
Restart Impala Daemon	Restarts all Impala Daemons in the cluster
Start Impala Daemon	Starts all Impala Daemons in the cluster
Stop Impala Daemon	Stops all Impala Daemons in the cluster

Impala Statestore actions

Action	Description
Check Impala Statestore	Verifies whether the Impala Statestore daemon works correctly
Move Impala Statestore	Allows you to move the Impala Statestore component from one host to another in the same cluster. When you run this action, a window with components and hosts is displayed, and you can delete the Impala Statestore component from a host and add it to another as described in the Add components article. The Impala Statestore component can only be installed on one host in a cluster
Restart Impala Statestore	Restarts the Impala Statestore daemon
Start Impala Statestore	Starts the Impala Statestore daemon
Stop Impala Statestore	Stops the Impala Statestore daemon

Impala Catalog Service actions

Action	Description
Check Impala Catalog	Verifies whether the Impala Catalog daemon works correctly
Move Impala Catalog	Allows you to move the Impala Catalog Service component from one host to another in the same cluster. When you run this action, a window with components and hosts is displayed, and you can delete the Impala Catalog Service component from a host and add it to another as described in the Add components article. The Impala Catalog Service component can only be installed on one host in a cluster
Restart Impala Catalog	Restarts the Impala Catalog Service daemon
Start Impala Catalog	Starts the Impala Catalog Service daemon
Stop Impala Catalog	Stops the Impala Catalog Service daemon

Impala Client actions

Action	Description
Add/Remove Client(s)	Allows you to remove or add the Impala Client component to the cluster hosts. When you run this action, a window with components and hosts is displayed, and you can redistribute the Impala Client component among the hosts as described in the Add components article. The Impala Client component can be installed on multiple cluster hosts

Kyuubi UI overview

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Management
  - Session
  - Operation
  - Engine
  - Server
- Swagger

Kyuubi has a built-in web server that enables access to Kyuubi REST API, server status information, and details about sessions, operations, and working engines.

> **NOTE**
>
> The Kyuubi web UI is an experimental feature and is not available by default. To enable it, set the value of the `kyuubi.frontend.protocols` to `THRIFT_BINARY,REST` manually or via ADCM.

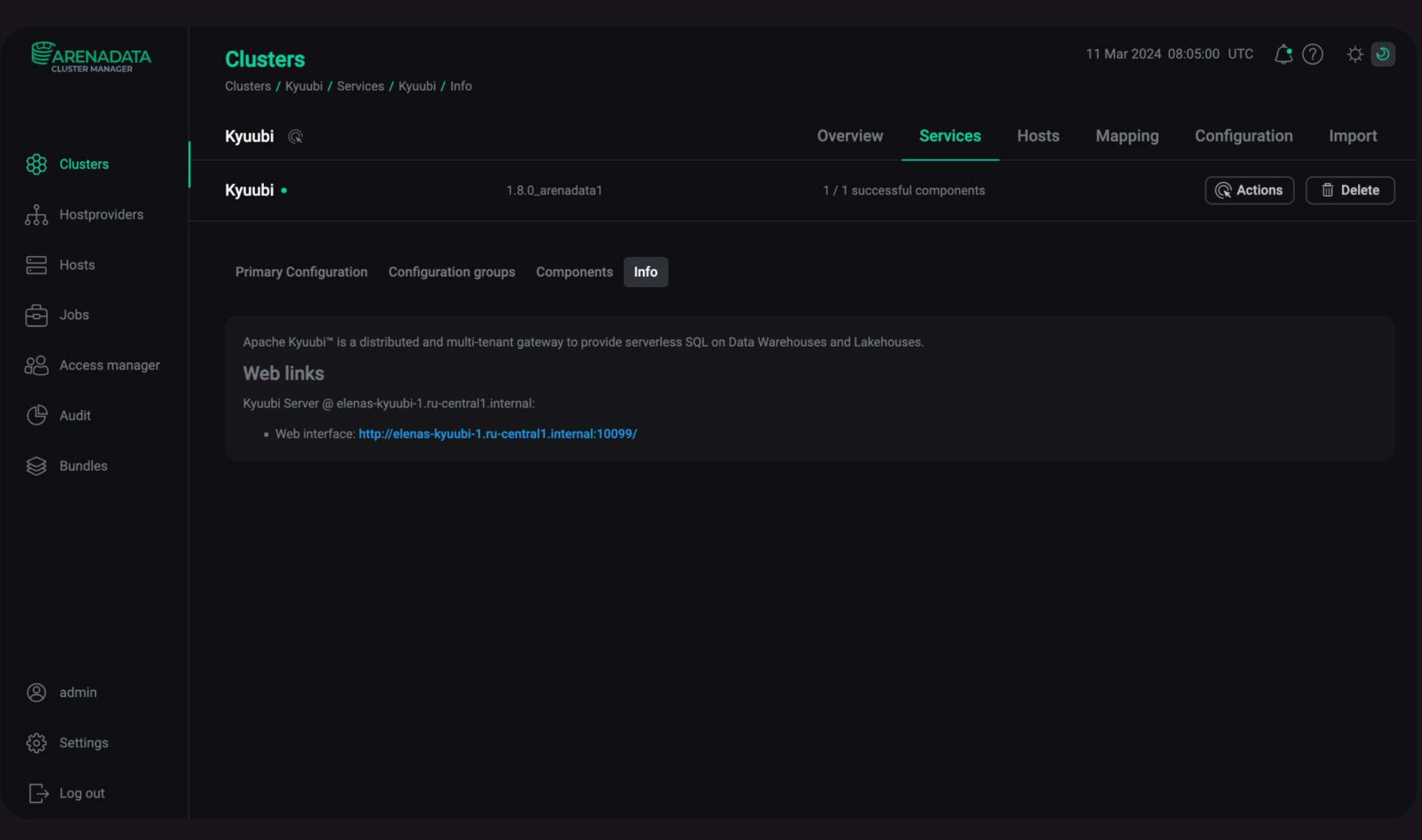
The web interface is available from the URL: *http://<server-hostname>:10099/*, where `<server-hostname>` is the IP address or FQDN of the host with the installed Kyuubi server.

For secure clusters, use `https://` instead of `http://`.

You can also find the link to Kyuubi UI in ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click on ***Kyuubi***.
3. Select ***Info***.

To change the default web UI port, provide a new value in the `kyuubi.frontend.rest.bind.port` property in the *kyuubi-defaults.conf* configuration file manually or via ADCM.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / Kyuubi / Services / Kyuubi / Info
11 Mar 2024 08:05:00 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Kyuubi
Overview
Services
Hosts
Mapping
Configuration
Import
Kyuubi
1.8.0_arenadata1
1 / 1 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info

Apache Kyuubi™ is a distributed and multi-tenant gateway to provide serverless SQL on Data Warehouses and Lakehouses.

Web links

Kyuubi Server @ elenas-kyuubi-1.ru-central1.internal:

- Web interface: http://elenas-kyuubi-1.ru-central1.internal:10099/

Kyuubi service web interface link in ADCM

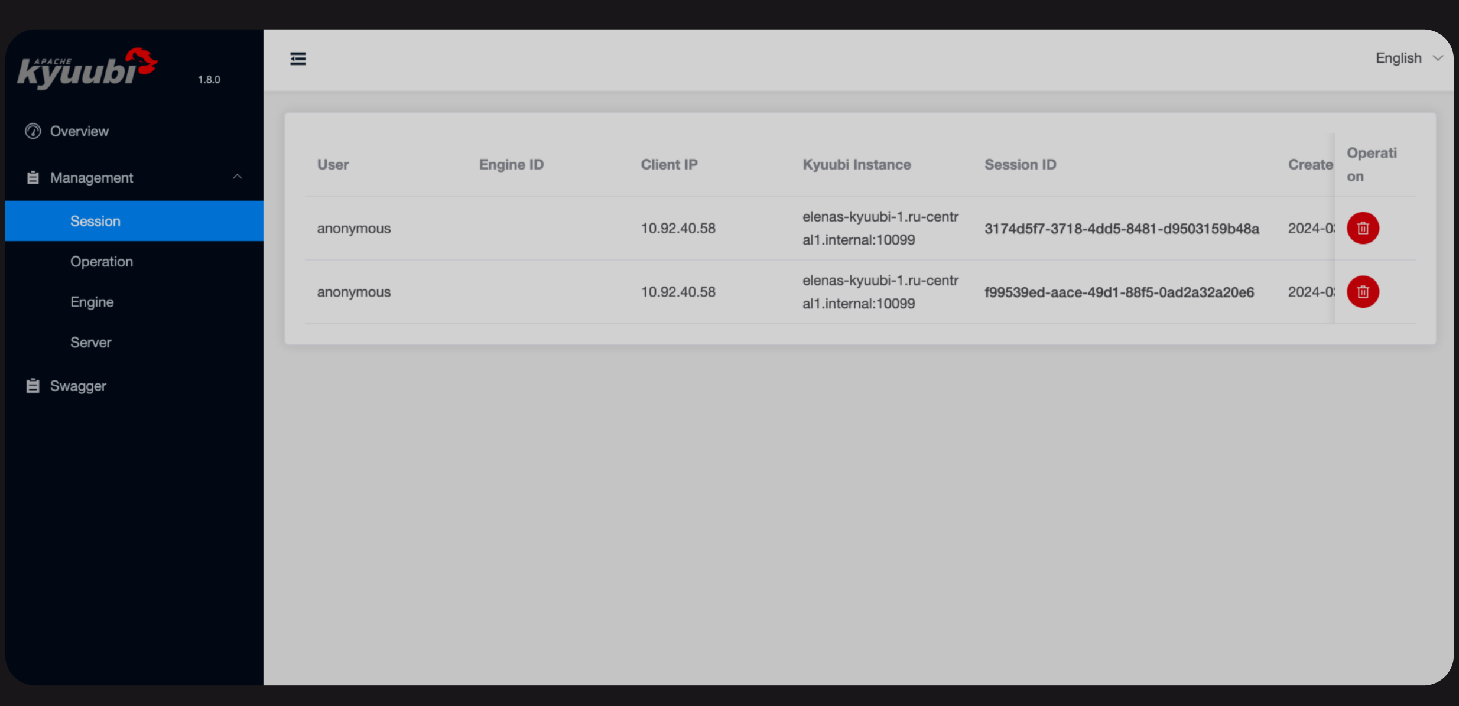
Overview

The ***Overview*** page is currently empty.

Management

The ***Management*** section displays information about sessions, operations, Kyuubi engines, and Kyuubi servers.

Session



Apache kyuubi
1.8.0
Overview
Management
Session
Operation
Engine
Server
Swagger
English

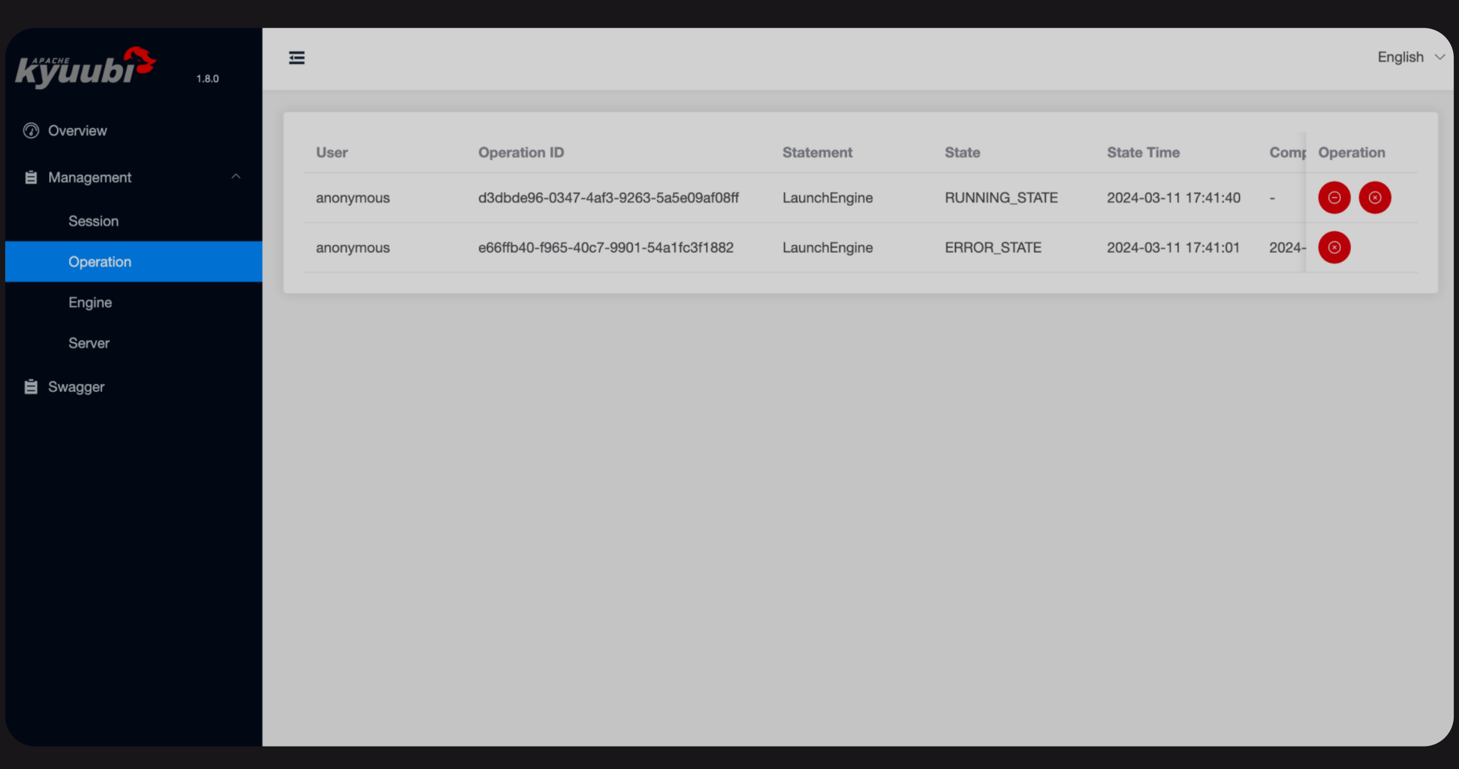
User	Engine ID	Client IP	Kyuubi Instance	Session ID	Create	Operation
anonymous		10.92.40.58	elenas-kyuubi-1.ru-central1.internal:10099	3174d5f7-3718-4dd5-8481-d9503159b48a	2024-0	
anonymous		10.92.40.58	elenas-kyuubi-1.ru-central1.internal:10099	f99539ed-aace-49d1-88f5-0ad2a32a20e6	2024-0	

Session page

The ***Session*** page displays a table with details about user sessions:

- ***User*** — the name of the user who created the session;
- ***Engine ID*** — the ID of the engine used by the user;
- ***Client IP*** — the IP address of the client that made a request;
- ***Kyuubi Instance*** — the FQDN and port of the Kyuubi server, which registered the session;
- ***Session ID*** — the ID of the session created by the user;
- ***Create Time*** — the time when the session has been created;
- ***Operation*** — the actions available for that session via web UI.

Operation



Apache kyuubi
1.8.0
Overview
Management
Session
Operation
Engine
Server
Swagger
English

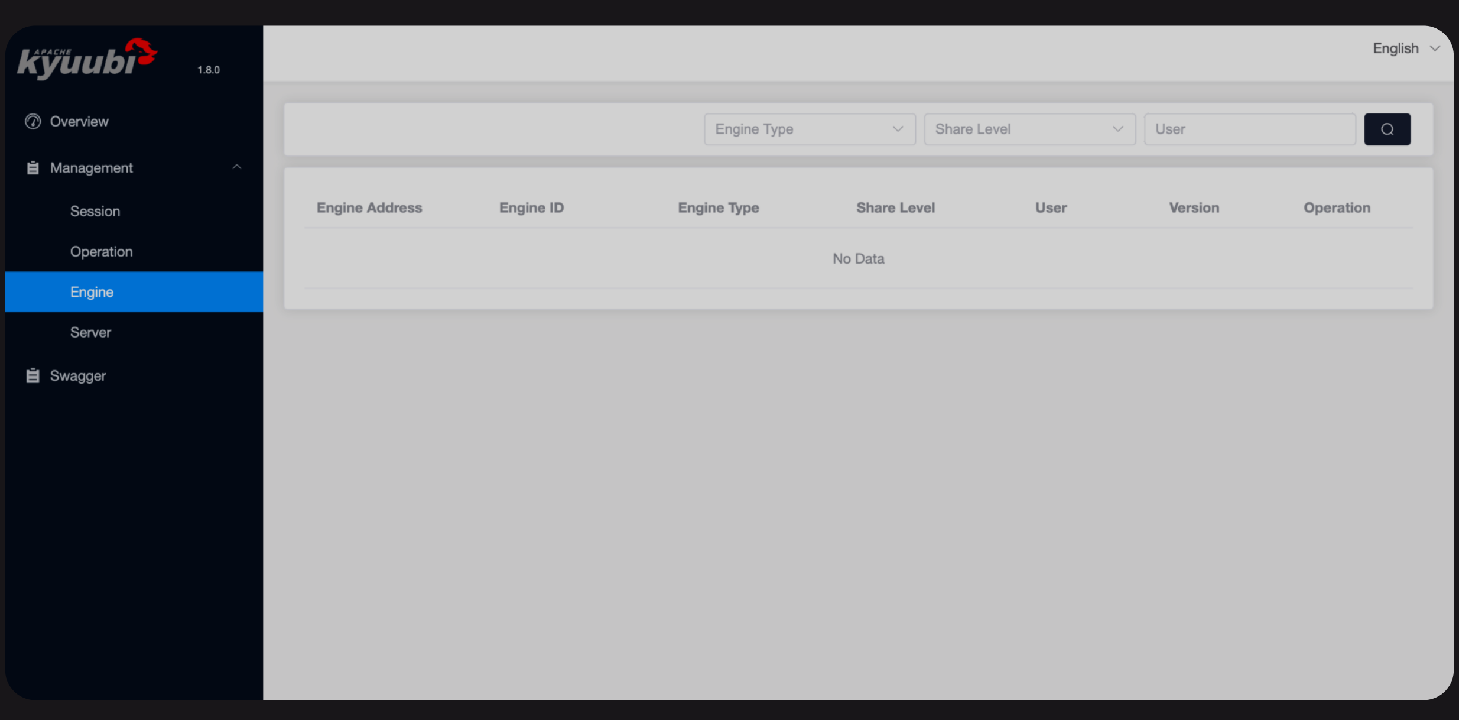
User	Operation ID	Statement	State	State Time	Comp	Operation
anonymous	d3dbde96-0347-4af3-9263-5a5e09af08ff	LaunchEngine	RUNNING_STATE	2024-03-11 17:41:40	-	
anonymous	e66ffb40-f965-40c7-9901-54a1fc3f1882	LaunchEngine	ERROR_STATE	2024-03-11 17:41:01	2024-	

Operation page

The ***Operation*** page displays a table with details about operations performed by users:

- ***User*** — the name of the user who started the operation;
- ***Operation ID*** — the ID of the operation requested by the user;
- ***Statement*** — the SQL of the statement;
- ***State*** — the current state of the operation;
- ***State Time*** — the amount of time the operation has spent in its current state;
- ***Completed Time*** — the end time of the operation;
- ***Duration*** — the time that the operation has been running;
- ***Operation*** — the actions available for that operation via web UI.

Engine



Apache kyuubi
1.8.0
Overview
Management
Session
Operation
Engine
Server
Swagger
English
Engine Type
Share Level
User

Engine Address	Engine ID	Engine Type	Share Level	User	Version	Operation
No Data						

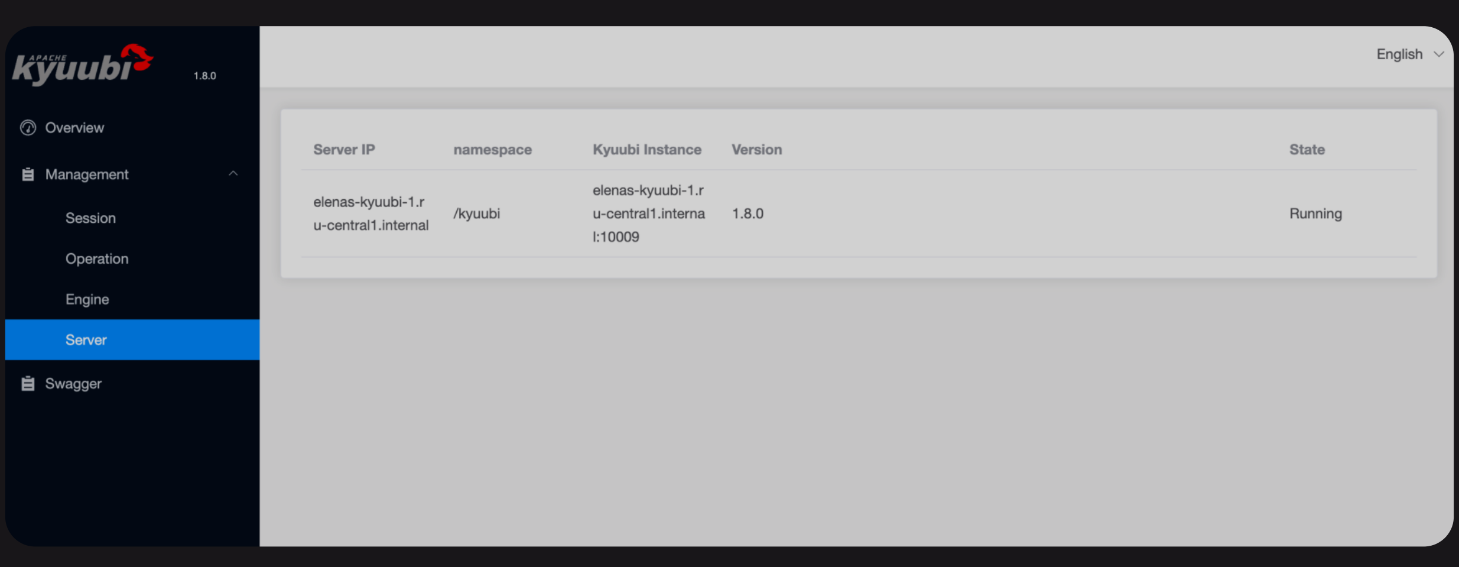
Engine page

The ***Engine*** page displays a table with details about Kyuubi engines:

- ***Engine Address*** — the IP address of the engine's server;
- ***Engine ID*** — the ID of the engine;
- ***Engine Type*** — the type of the engine (the current version of web UI only shows SPARK-SQL engines);
- ***Share Level*** — the share level of the engine;
- ***User*** — the name of the user who created the engine;
- ***Version*** — the version of the Kyuubi server associated with this engine;
- ***Operation*** — the actions available for that engine via web UI.

> TIP
>
> Not all available engines are displayed by default. You can select which engines to display by setting the filter conditions at the top of the page.

Server



Apache kyuubi
1.8.0
Overview
Management
Session
Operation
Engine
Server
Swagger
English

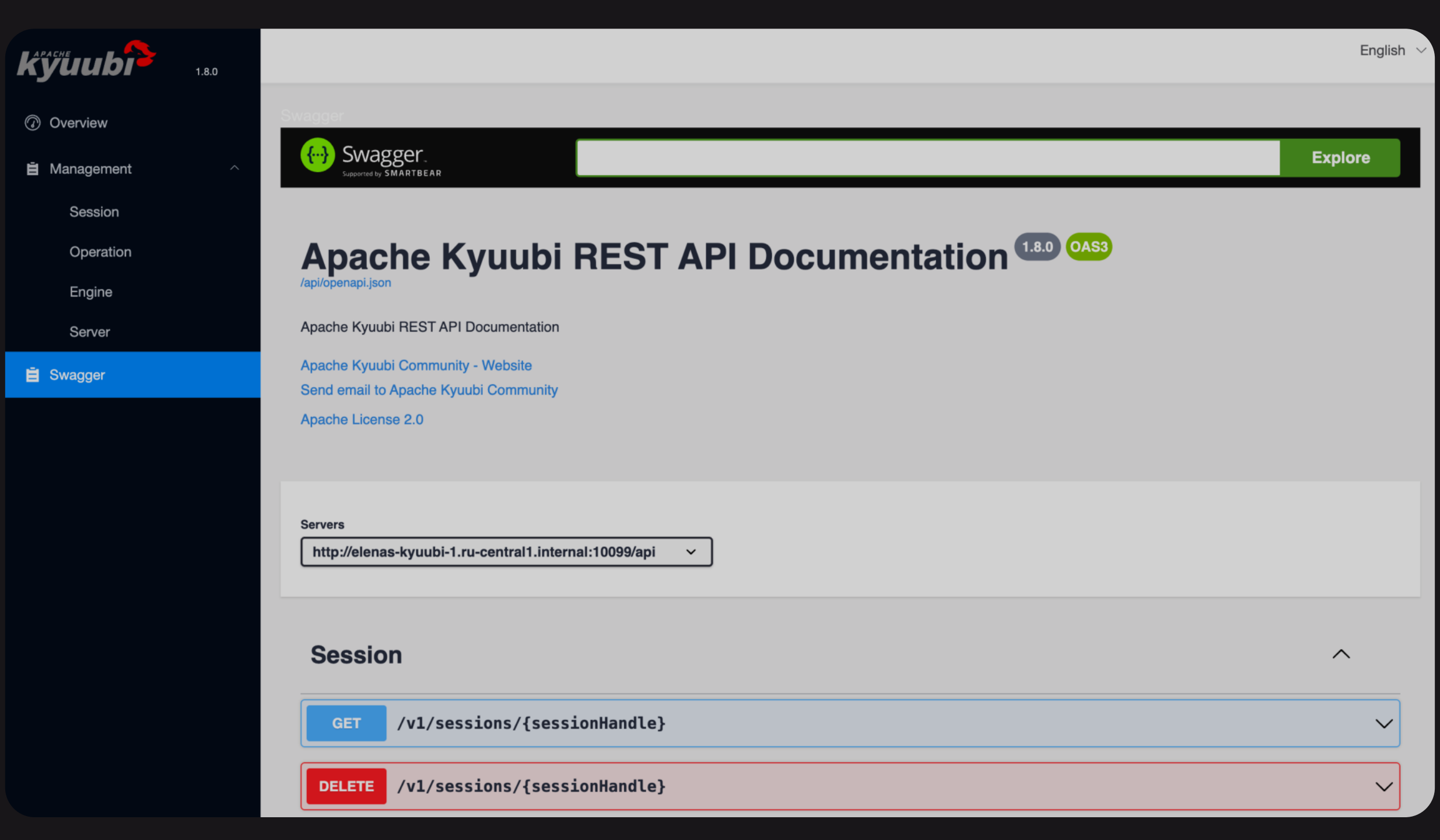
Server IP	namespace	Kyuubi Instance	Version	State
elenas-kyuubi-1.ru-central1.internal	/kyuubi	elenas-kyuubi-1.ru-central1.internal:10009	1.8.0	Running

Server page

The ***Server*** page displays a table with details about Kyuubi servers:

- ***Server IP*** — the Kyuubi server's IP address;
- ***namespace*** — the namespace to which the server belongs;
- ***Kyuubi Instance*** — the FQDN and port of the server;
- ***Version*** — the version of the Kyuubi server;
- ***State*** — the current state of the server (for example, `Running`).

Swagger



Apache kyuubi
1.8.0
Overview
Management
Session
Operation
Engine
Server
Swagger
English
Swagger
Supported by SMARTBEAR
Explore
1.8.0
OAS3
Session
GET
DELETE

Apache Kyuubi REST API Documentation

/api/openapi.json

Apache Kyuubi REST API Documentation

Apache Kyuubi Community - Website

Send email to Apache Kyuubi Community

Apache License 2.0

Servers: http://elenas-kyuubi-1.ru-central1.internal:10099/api

GET `/v1/sessions/{sessionHandle}`

DELETE `/v1/sessions/{sessionHandle}`

Swagger page

The ***Swagger*** page displays an interactive Kyuubi REST API reference. It enables sending requests to Kyuubi via web interface.

To send a request, click at the desired method, select ***Try it out***, enter parameters if necessary, and click ***Execute***.

LDAP authentication in Kyuubi

Sergei Tikhomirov

Contents

To Table of Contents

Enable LDAP

Check

Kyuubi can be configured to enable frontend LDAP authentication for clients (e.g. BeeLine shell) or JDBC/ODBC drivers.

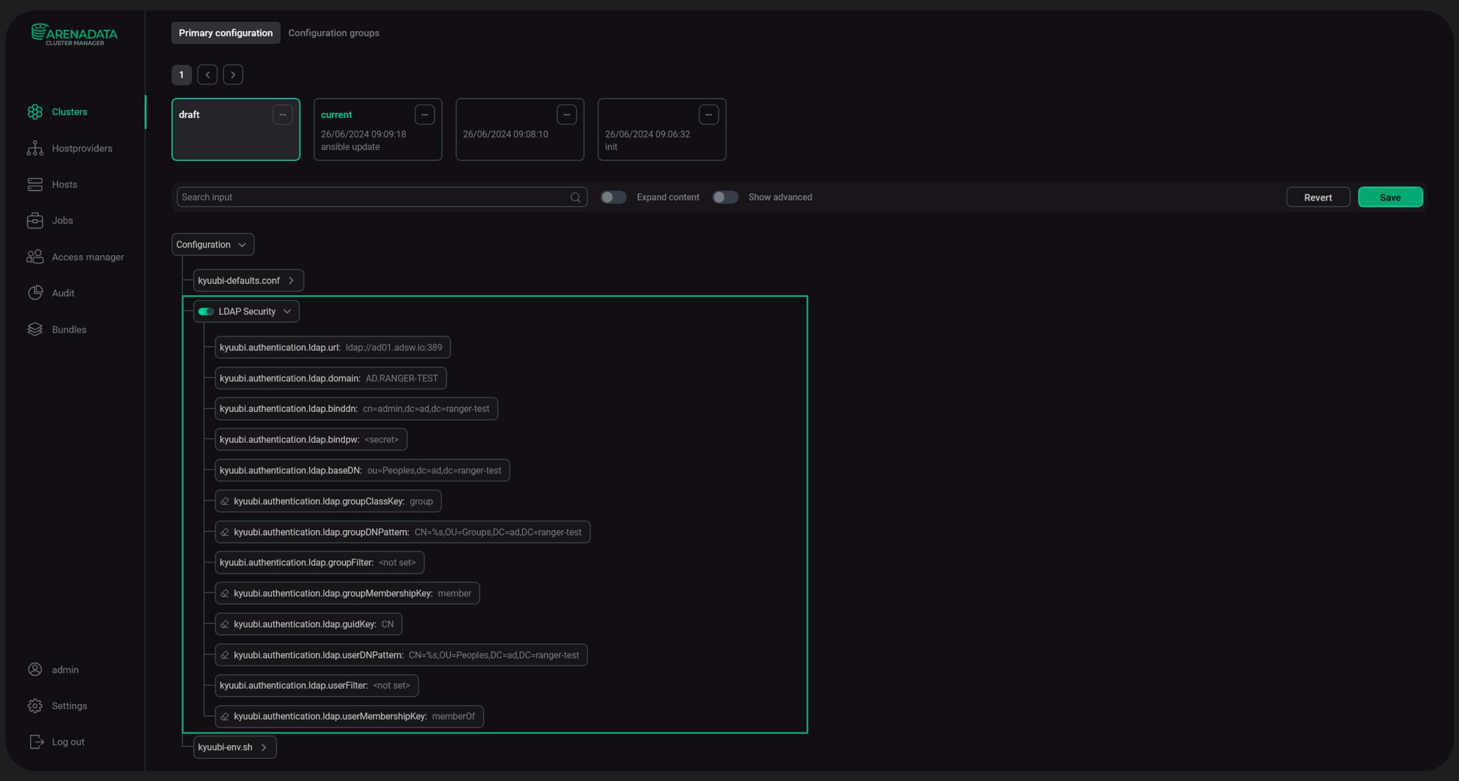
> **NOTE**
>
> If you enable Kerberos on your cluster, the LDAP authentication for Kyuubi will be automatically turned on.

Enable LDAP

To enable LDAP authentication, follow the steps below:

1. On the ***Clusters*** page, select your ADH cluster and proceed to the ***Services*** tab in the cluster menu.
2. Select the ***Kyuubi*** service and head to the ***Components*** tab.
3. In the ***Kyuubi Server*** component configuration, activate the ***LDAP Security*** parameter group.
4. Fill in the LDAP parameters for Active Directory or FreeIPA and restart Kyuubi.

Active Directory

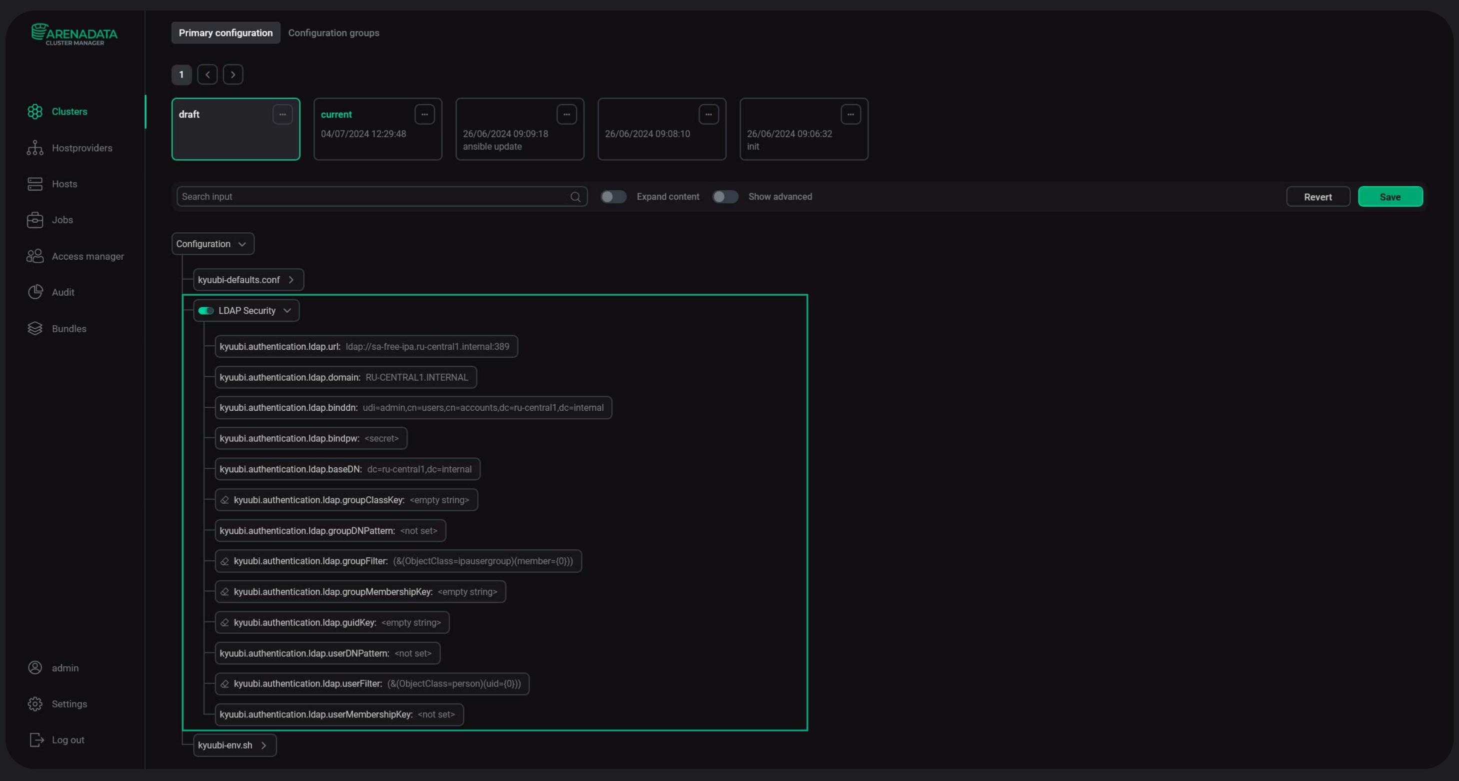


ARENADATA
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
1
draft
current
26/06/2024 09:09:18
ansible update
26/06/2024 09:08:10
26/06/2024 09:06:32
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
kyuubi-defaults.conf
LDAP Security
kyuubi-env.sh

kyuubi.authentication.ldap.url: ldap://ad01.adsw.io:389
kyuubi.authentication.ldap.domain: AD.RANGER-TEST
kyuubi.authentication.ldap.binddn: cn=admin,dc=ad,dc=ranger-test
kyuubi.authentication.ldap.bindpw: <secret>
kyuubi.authentication.ldap.baseDN: ou=Peoples,dc=ad,dc=ranger-test
kyuubi.authentication.ldap.groupClassKey: group
kyuubi.authentication.ldap.groupDNPattern: CN=%s,OU=Groups,DC=ad,DC=ranger-test
kyuubi.authentication.ldap.groupFilter: <not set>
kyuubi.authentication.ldap.groupMembershipKey: member
kyuubi.authentication.ldap.guidKey: CN
kyuubi.authentication.ldap.userDNPattern: CN=%s,OU=Peoples,DC=ad,DC=ranger-test
kyuubi.authentication.ldap.userFilter: <not set>
kyuubi.authentication.ldap.userMembershipKey: memberOf

Example of an LDAP AD configuration

FreeIPA



ARENADATA
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
1
draft
current
04/07/2024 12:29:48
26/06/2024 09:09:18
ansible update
26/06/2024 09:08:10
26/06/2024 09:06:32
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
kyuubi-defaults.conf
LDAP Security
kyuubi-env.sh

kyuubi.authentication.ldap.url: ldap://sa-free-ipa.ru-central1.internal:389
kyuubi.authentication.ldap.domain: RU-CENTRAL1.INTERNAL
kyuubi.authentication.ldap.binddn: udi=admin,cn=users,cn=accounts,dc=ru-central1,dc=internal
kyuubi.authentication.ldap.bindpw: <secret>
kyuubi.authentication.ldap.baseDN: dc=ru-central1,dc=internal
kyuubi.authentication.ldap.groupClassKey: <empty string>
kyuubi.authentication.ldap.groupDNPattern: <not set>
kyuubi.authentication.ldap.groupFilter: (&(ObjectClass=ipausergroup)(member={0}))
kyuubi.authentication.ldap.groupMembershipKey: <empty string>
kyuubi.authentication.ldap.guidKey: <empty string>
kyuubi.authentication.ldap.userDNPattern: <not set>
kyuubi.authentication.ldap.userFilter: (&(ObjectClass=person)(uid={0}))
kyuubi.authentication.ldap.userMembershipKey: <not set>

Example of an LDAP FreeIPA configuration

Check

To check if the LDAP authentication works, run a BeeLine shell command like the one below:

```
$ beeline -u 'jdbc:hive2://stikhomirov-adh-1.ru-central1.internal:2181,stikhomirov-adh-2.ru-central1.internal:2181,stikhomirov-adh-3.ru-central1.internal:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/67/kyuubi?kyuubi.engine.type=HIVE_SQL' -n <name> -p <password>
```

where:

- `<name>` is an LDAP username.
- `<password>` is a password for the username.

The correct output of the command should be:

```
Connecting to jdbc:hive2://stikhomirov-adh-1.ru-central1.internal:2181,stikhomirov-adh-2.ru-central1.internal:2181,stikhomirov-adh-3.ru-central1.internal:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/67/kyuubi?kyuubi.engine.type=HIVE_SQL
24/03/06 14:21:03 [main]: INFO jdbc.HiveConnection: Connected to stikhomirov-adh-1.ru-central1.internal:10009
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
```

Kyuubi service management via ADCM

Sergei Tikhomirov

To Table of Contents

The ADCM UI provides actions to manage the Kyuubi service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Kyuubi service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Kyuubi components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

IMPORTANT

In order to use a query engine (e.g. Flink SQL/Hive/Spark SQL), it’s required to map a corresponding service’s client onto the Kyuubi’s host. For common configurations, just HDFS and YARN clients are required.

Kyuubi has just one component — ***Kyuubi Server***, which has actions of its own.

The Kyuubi Server component actions

Action	Description
Add/Remove Kyuubi Server(s)	Adds/removes the Kyuubi Server component(s). Running this action opens the component-host mapping interface where you can distribute the Kyuubi Server components
Check	Checks the Kyuubi Server component for errors
Restart	Restarts the Kyuubi Server component. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all component configurations defined in ADCM settings will be applied on the component startup. Otherwise, the component starts without applying configurations from ADCM
Start	Starts the Kyuubi Server component. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all component configurations defined in ADCM settings will be applied on the component startup. Otherwise, the component starts without applying configurations from ADCM
Stop	Stops the Kyuubi Server component

Quick start with Mapreduce

Albert Bagdasaryan

Contents

To Table of Contents

Mapper code
Reducer code
Driver code

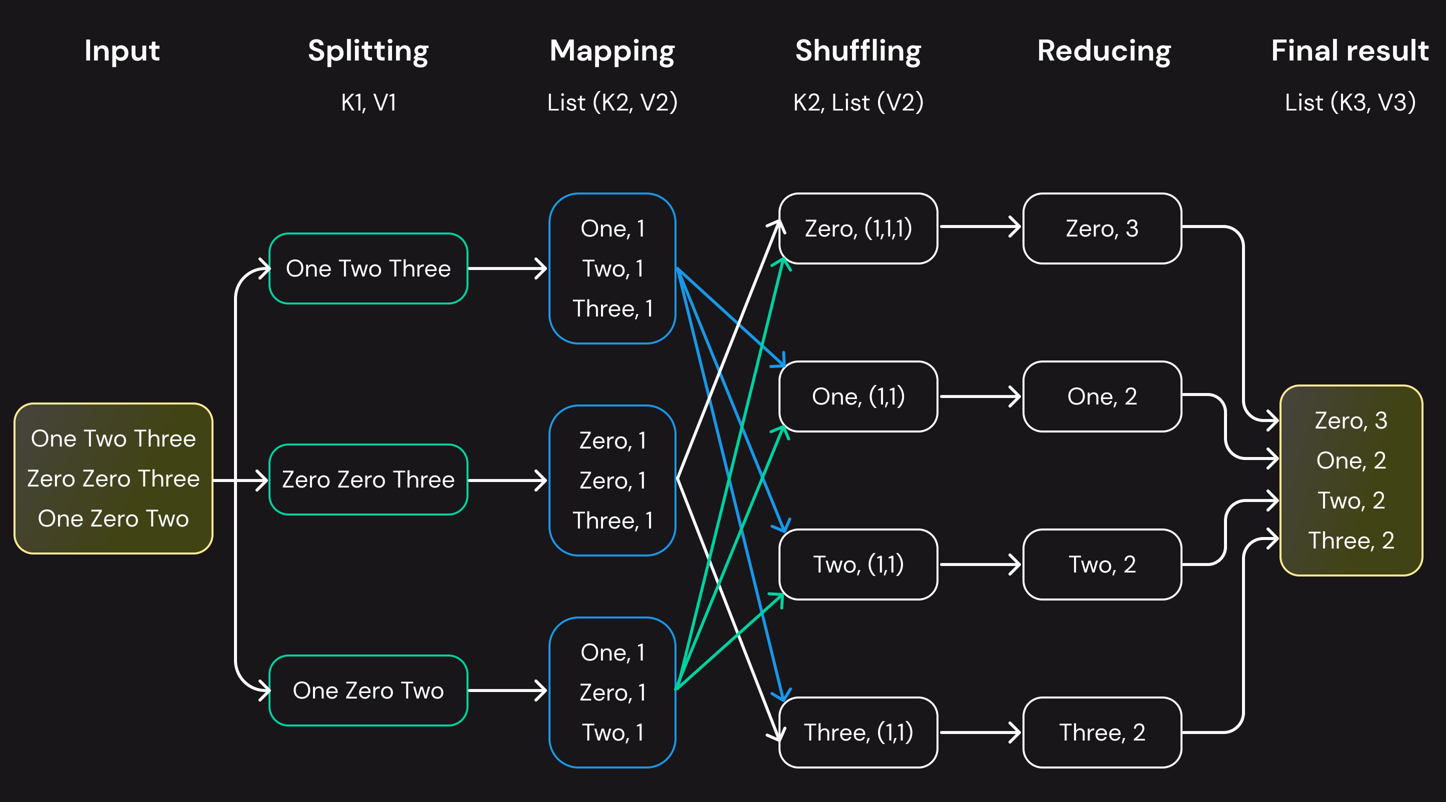
A MapReduce application typically consists of three code parts:

- Mapper code
- Reducer code
- Driver code

Those parts are usually based on classes with the same names:

- **Mapper**. This class is responsible for the first stage in data processing. `InputSplit` is the logical representation of the data. It describes a unit of work that contains a single map task in a MapReduce job. `RecordReader` processes each input record and generates a respective key-value pair. Hadoop's Mapper store saves this intermediate data to the local disk.
- **Reducer**. This class is responsible for the second stage. The MapReduce service sends the intermediate output generated by the mappers to the reducers that process it and generate the final output to be saved in HDFS.
- **Driver**. This class is responsible for setting up a MapReduce job to run in Hadoop.

In this example, we will count words in the *sample.txt* text file using MapReduce. Let us find the unique words and the number of occurrences of those unique words. Data processing goes through several stages as presented in the following diagram.



MapReduce workflow

1. We divide input data into three splits as shown in the diagram. This will distribute the work among all DataNodes.
2. We tokenize the words in each of the mappers and give a hardcoded value `1` to every word (known as token). The rationale behind giving a hardcoded value equal to `1` is that every word itself occurs once.
3. A list of key-value pairs are created where the keys are individual words and values are `1`. For the first line ("One Two Three") we have 3 key-value pairs: ("One", 1), ("Two", 1), and ("Three", 1). The mapping process is the same for all involved DataNodes.
4. After the map phase, a partition process starts to sort and shuffle the intermediate data, so that all pairs with the same key are sent to one reducer.
5. After the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding exactly to that key. For example: ("One", [1,1]), ("Zero", [1,1,1]), and so on.
6. Each reducer counts the values that are present in that list of values. As shown in the diagram, a reducer gets a list of values, for example, [1,1] for the key "One". Then it counts the number of ones in the list and gives the final output as ("One", 2).
7. Finally, all the output key and value pairs are collected and written to the output file.

Mapper code

The mapper code performs the following operations:

1. Create a class `Map` that extends the `Mapper` class from the MapReduce library.
2. Define the data types of input and output keys and values in the class declaration using angle brackets. Both the input and output of the map method in the class is a key and value pair.

   Input:

   - The key is the offset of a line in the text file: `LongWritable` type.
   - The value is an individual line: `Text` type.

   Output:

   - The key is a tokenized word: `Text` type.
   - We have the hardcoded value in our case which is 1: `IntWritable` type.
     Output example: ("Dear", 1), ("Bear", 1), and so on.

The result is a Java code that breaks an input line into words and assigns them a hardcoded value equal to 1.

Reducer code

The reducer code performs the following operations:

1. Create a class `Reduce` that extends class `Reducer` similar to what we did for mapper.
2. Define the data types of input and output keys and values in the class declaration using angle brackets as done for the mapper. Both the input and output are key and value pairs.

   Input:

   - The key is a unique word generated at the sorting and shuffling phase: `Text` type.
   - The value is a list of integers corresponding to each key: `IntWritable` type.
     Input example: ("Bear", [1, 1]), ("Car", [1, 1, 1]), and so on.

   Output:

   - The key one of unique words present in the input text file: `Text` type.
   - The value is the number of occurrences of each of the unique words: `IntWritable` type.
     Output example: ("Bear", 2), ("Car", 3), and so on.

The discussed code aggregates the values present in the list of each key and produces the final answer. In general, a single reducer is created for each of the unique words, and you can specify the number of reducers in *mapred-site.xml*.

Driver code

The driver code performs the following operations:

1. Set the configuration of our MapReduce job that will run in Hadoop.
2. Specify the name of the job and the data types for input and output of the mapper and reducer.
3. Specify the names of the mapper and reducer classes.
4. Specify the path to the folders with input and output data.

The `setInputFormatClass()` method specifies how a mapper will read the input data or what will be the unit of work. Here we have chosen the `TextInputFormat` type, so that the mapper reads a single line at a time from the input text file.

The `main()` method is the entry point for the driver. In this method, we invoke a new `Configuration` object for the job.

Connect to MapReduce via CLI

Elena Kostyuchenko

Contents

To Table of Contents

Overview
Usage examples

Overview

MapReduce provides a command line interface implemented as a *bin/mapred* script. To interact with MapReduce via CLI, connect to a cluster host with a MapReduce component via SSH and run the desired MapReduce CLI command.

For example, check the version by running:

```
$ mapred version
```

Output example:

```
Hadoop 3.2.4
Source code repository git@ssh.gitlab.adsw.io:arenadata/infrastructure/code/ci/prj_adh.git -r
3cb85f40e394dcfb50fe77310908cce385381ba2
Compiled by jenkins on 2024-03-20T17:53Z
Compiled with protoc 2.5.0
From source with checksum ee031c16fe785bbb35252c749418712
This command was run using /usr/lib/hadoop/hadoop-common-3.2.4.jar
```

Running `mapred` without any arguments prints the list of all MapReduce commands.

The commands have the following syntax:

```
$ mapred [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [COMMAND_OPTIONS]
```

Where:

- `SHELL_OPTIONS` — Hadoop shell options listed in the Shell Options section of Hadoop Commands Reference.
- `GENERIC_OPTIONS` — standard options supported by multiple commands. For more information, see the Generic Options section of Hadoop Commands Reference.
- `COMMAND_OPTIONS` — command-specific options described in references for each command.

Usage examples

Typically, a MapReduce job is launched using the yarn jar or hadoop jar commands. But you can use MapReduce CLI, for example, to get information about specific jobs and control their execution.

To see the list of all jobs, run:

```
$ mapred job -list all
```

Output example:

```
Total jobs:13
                  JobId              JobName         State           StartTime      UserName           Queue      Priority
UsedContainers  RsvdContainers      UsedMem         RsvdMem         NeededMem         AM info
 job_1713528166269_0001      QuasiMonteCarlo     SUCCEEDED       1713529720429          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1713528166269_0001/
 job_1713770455913_0001      QuasiMonteCarlo     SUCCEEDED       1713770623388          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1713770455913_0001/
 job_1713770455913_0002      QuasiMonteCarlo     SUCCEEDED       1713771341014          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1713770455913_0002/
 job_1713964485222_0001      QuasiMonteCarlo          PREP       1713965480860          yarn         default       DEFAULT
1               0        2048M              0M             2048M      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1713964485222_0001/
 job_1701175565199_0001      QuasiMonteCarlo     SUCCEEDED       1701175708248          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1701175565199_0001/
 job_1705311887839_0001      QuasiMonteCarlo     SUCCEEDED       1705319626157          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1705311887839_0001/
 job_1705311887839_0002      QuasiMonteCarlo     SUCCEEDED       1705319754517          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1705311887839_0002/
 job_1705573015955_0002      QuasiMonteCarlo     SUCCEEDED       1705658611315          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1705573015955_0002/
 job_1704872668642_0003      QuasiMonteCarlo     SUCCEEDED       1704884498654          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1704872668642_0003/
 job_1712415098046_0001      QuasiMonteCarlo     SUCCEEDED       1712416353017          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1712415098046_0001/
 job_1704872668642_0004      QuasiMonteCarlo     SUCCEEDED       1704884603925          yarn         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1704872668642_0004/
 job_1704872668642_0001              distcp     SUCCEEDED       1704884156346          hdfs         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1704872668642_0001/
 job_1704872668642_0002              distcp     SUCCEEDED       1704884247607          hdfs         default       DEFAULT
N/A             N/A          N/A             N/A               N/A      http://elenas-adh2.ru-central1.
internal:8088/proxy/application_1704872668642_0002/
```

To see the status of a specific job, run:

```
$ mapred job -status <job-ID>
```

Where `job-ID` is the ID of the job for which you want to check the status.

Output example:

```
Job: job_1713964485222_0001
Job File: hdfs://adh/user/yarn/.staging/job_1713964485222_0001/job.xml
Job Tracking URL : http://elenas-adh2.ru-central1.internal:8088/proxy/application_1713964485222_0001/
Uber job : false
Number of maps: 16
Number of reduces: 1
map() completion: 0.625
reduce() completion: 0.0
Job state: RUNNING
retired: false
reason for failure:
Counters: 33
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=2306730
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2536
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=40
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
        Job Counters
                Launched map tasks=11
                Data-local map tasks=11
                Total time spent by all maps in occupied slots (ms)=19885
                Total time spent by all map tasks (ms)=19885
                Total vcore-milliseconds taken by all map tasks=19885
                Total megabyte-milliseconds taken by all map tasks=20362240
        Map-Reduce Framework
                Map input records=10
                Map output records=20
                Map output bytes=180
                Map output materialized bytes=280
                Input split bytes=1356
                Combine input records=0
                Spilled Records=20
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=406
                CPU time spent (ms)=3600
                Physical memory (bytes) snapshot=3162824704
                Virtual memory (bytes) snapshot=28058554368
                Total committed heap usage (bytes)=2188902400
                Peak Map Physical memory (bytes)=346021888
                Peak Map Virtual memory (bytes)=2810654720
        File Input Format Counters
                Bytes Read=1180
```

Use MapReduce REST APIs

Elena Kostyuchenko

Contents

To Table of Contents

Application Master REST API

History Server REST API

ADH MapReduce components support the ability to send REST API requests to Application Master and History Server.

Using MapReduce REST API, you can get information about the running and finished applications, jobs, and tasks. The same information is available in YARN UI.

Certain requests might require permissions. You can configure user permissions for your cluster in the ACL settings.

Application Master REST API

The Application Master accepts requests sent via a proxy. You can use Resource Manager as a proxy or configure a separate host.

The URL for requests can have one of the following formats (the output will be the same):

> *http://<proxy-address>:<port>/proxy/<app-ID>/ws/v1/mapreduce*
>
> *http://<proxy-address>:<port>/proxy/<app-ID>/ws/v1/mapreduce/info*

Where:

- `<proxy-address>` — the IP address of the proxy server;
- `<port>` — the port of the proxy server;
- `<app-ID>` — the ID of the application.

The example request to Application Master:

```
$ curl -L http://127.0.0.1:8088/proxy/application_1713528166269_0001/ws/v1/mapreduce/info -H "Accept: application/xml"
```

This command requests information about the running application with the ID `application_1713528166269_0001` in the XML format. You can request a JSON in the response by using the `Accept: application/json` header.

The Application Master responds with the following:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><info><appId>application_1713528166269_0001</appId>
<name>QuasiMonteCarlo</name><user>yarn</user><startedOn>1713529723410</startedOn><elapsedTime>38797</elapsedTime></info>
```

The full API specification is available at MapReduce Application Master REST API's.

History Server REST API

The History Server REST API provides information about the finished applications.

Here's an example of the command that returns the jobs list:

```
$ curl -X GET http://<history-server-address>:8042/ws/v1/history/mapreduce/jobs -H "Accept: application/json"
```

Where `<history-server-address>` is the IP address or FQDN of the History Server host.

This command requests information about the finished jobs in the JSON format. You can request an XML response by using the `Accept: application/xml` header.

There are additionall parameters that can be used to filter the output:

- `user` — name of the user that started the application;
- `state` — job state;
- `queue` — name of the queue;
- `limit` — number of application objects to return;
- `startedTimeBegin` — jobs that started after the specified time in milliseconds;
- `startedTimeEnd` — jobs that started before the specified time in milliseconds;
- `finishedTimeBegin` — jobs that finished after the specified time in milliseconds;
- `finishedTimeEnd` — jobs that finished before the specified time in milliseconds.

Here's an example command that requests the list of jobs that started between `1713770628450` and `1713770680980`:

```
$ curl -X GET http://127.0.0.1:8088:19888/ws/v1/history/mapreduce/jobs?startedTimeBegin=1713770628450&startedTimeEnd=1713770680980
-H "Accept: application/json"
```

The server responds with the object:

```
{"jobs":{"job":
[{"submitTime":1713770623388,"startTime":1713770628457,"finishTime":1713770680975,"id":"job_1713770455913_0001","name":"QuasiMont
eCarlo","queue":"default","user":"yarn","state":"SUCCEEDED","mapsTotal":16,"mapsCompleted":16,"reducesTotal":1,"reducesCompleted"
:1},
{"submitTime":1713771341013,"startTime":1713771345766,"finishTime":1713771398188,"id":"job_1713770455913_0002","name":"QuasiMonte
Carlo","queue":"default","user":"yarn","state":"SUCCEEDED","mapsTotal":16,"mapsCompleted":16,"reducesTotal":1,"reducesCompleted":
1}]}}
```

The full API specification is available at MapReduce History Server REST API's.

Logging in MapReduce

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Grep logs

Logging levels

Container logs

Archive logs

Overview

MapReduce records command output and processes events in text logs. This information can be useful when diagnosing technical issues.

MapReduce History Server stores logs locally, on the host it occupies. The logs have the *.log* extension and are located in the *var/log/hadoop-mapreduce/* directory. In the same directory, you can find *.out* files that contain component restart information.

All logs of all YARN components, including MapReduce logs, are also available in YARN UIs.

The logs naming convention is `hadoop-mapred-<component>-<host>.log`.

Where:

- `<component>` — a component name, for example, History Server;
- `<host>` — the FQDN of the component host.

The MapReduce logs configuration shares the same log4j.properties file with HDFS. You can edit this file to control the logging of MapReduce tasks.

Grep logs

You can search through the logs for a specific information, like error messages. To do this, connect to the host with the component whose logs you want to inspect, and use a `grep` command.

For example:

```
$ cat /var/log/hadoop-mapreduce/hadoop-mapred-historyserver-elenas-adh2.ru-central1.internal.log | grep -i -A3 -B1 error
```

This command searches for messages containing the word `error` in the History Server log, located on the `elenas-adh2.ru-central1.internal` host. The `-i` option allows you to ignore case distinctions. The `-A3 -B1` options expand the output to one line before and three lines after the line containing the error.

The example output:

```
2024-04-20 08:53:57,337 INFO org.apache.hadoop.mapreduce.v2.hs.JobHistory: Starting scan to move intermediate done files
2024-04-20 08:55:59,664 ERROR org.apache.hadoop.mapreduce.v2.hs.JobHistoryServer: RECEIVED SIGNAL 15: SIGTERM
2024-04-20 08:55:59,668 INFO org.apache.hadoop.metrics2.impl.MetricsSystemImpl: Stopping JobHistoryServer metrics system...
2024-04-20 08:55:59,668 INFO org.apache.hadoop.metrics2.impl.MetricsSystemImpl: JobHistoryServer metrics system stopped.
2024-04-20 08:55:59,669 INFO org.apache.hadoop.metrics2.impl.MetricsSystemImpl: JobHistoryServer metrics system shutdown complete.
```

Logging levels

MapReduce uses the Log4j2 library for logging which supports the following log levels (from least to most informative):

1. `FATAL` — indicates that an operation can't continue execution and will terminate.
2. `ERROR` — notifies that a program is not working correctly or has stopped.
3. `WARN` — warns about potential problems. This doesn't mean that a program is not working, but raises a concern.
4. `INFO` — informs regarding the program lifecycle or state.
5. `DEBUG` — prints debugging information about internal states of the program.
6. `TRACE` — prints messages tracing the execution flow of a program.

The Log4j2 loggers also accept logging level values: `OFF` — for switching off the logging, and `ALL` — for allowing all types of messages.

Setting one level of logging will enable this level and all levels above it. For example, if you set the logging level to `WARN`, then only warnings, errors, and fatal messages would get into the log files, but not `INFO`, `DEBUG`, and `TRACE`.

Container logs

Applications submitted to YARN generate their own logs. To read application logs, use the logs command.

For example:

```
$ yarn logs -applicationId application_1714649647710_0001
```

Archive logs

You can make an archive of MapReduce logs by using the archive-logs command. This process is similar to the YARN log aggregation feature.

The example command for creating an archive of ten log files:

```
$ mapred archive-logs -minNumberLogFiles 10
```

The default value for the `-minNumberLogFiles` is 20, which means that running the `mapred archive-logs` command will create an archive only if 20 or more logs are present.

Pluggable shuffle and sort in MapReduce

Elena Kostyuchenko

To Table of Contents

The sort and shuffle phases in MapReduce are performed by default Hadoop services, but you can configure a custom shuffle and sort implementation for your cluster.

One of the possible use cases is configuring a different application protocol, other than HTTP, such as RDMA for shuffling data; or replacing the sort logic with custom algorithms that enable hash aggregation and Limit-N query.

For more information on MapReduce, see the MapReduce overview and MapReduce architecture articles.

> **CAUTION**
>
> The pluggable shuffle and sort capabilities are experimental and unstable. This means that provided APIs may change and break compatibility in future versions of Hadoop.

The shuffling is done by the NodeManager's auxiliary service and the shuffle consumer plugin.

You can change the default auxiliary service, by changing the NodeManager configuration properties listed in the table below.

NodeManager's auxiliary service parameters

Parameter	Description	Default value
yarn.nodemanager.aux-services	The auxiliary service name	…,mapreduce_shuffle
yarn.nodemanager.aux-services.mapreduce_shuffle.class	The auxiliary service class name	org.apache.hadoop.mapred.ShuffleHandler
yarn.nodemanager.aux-services.<class_name>.classpath	The path to the local directory with the auxiliary service JAR files as well as the dependencies. The path can lead to a single JAR file, use the `{local_dir_to_jar}/*` pattern to load all JARs under the directory, or contain multiple values separated by `:`	—
yarn.nodemanager.aux-services.<class_name>.remote-classpath	The parameter to use instead of the `yarn.nodemanager.aux-services.<class_name>.classpath` if the auxiliary service JAR files are located in the remote file system	—

To change these properties on all hosts via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at ***YARN***.
3. Toggle the ***Show advanced*** option and find ***yarn-site.xml***.
4. Open the parameter drop-down list, select ***yarn.nodemanager.aux-services***, and add a new property.
5. Select the ***yarn.nodemanager.aux-services.mapreduce_shuffle.class*** and enter a new value.
6. Find the ***Custom yarn-site.xml*** parameter, select ***Add property***, and enter the new parameter name depending on the location of your auxiliary service JAR files: either `yarn.nodemanager.aux-services.<class_name>.classpath` or `yarn.nodemanager.aux-services.<class_name>.remote-classpath`.
7. Confirm changes to YARN configuration by clicking ***Save***.
8. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

To change the default shuffle consumer plugin, set a new value for the `mapreduce.job.reduce.shuffle.consumer.plugin.class` property.

Setting a new value for a single node's environment will change the shuffle method on a per-job basis. To set a new shuffle consumer plugin for all nodes, edit the same property in the *mapred-site.xml* configuration file via ADCM.

> **NOTE**
>
> If you change both the auxiliary service and the default shuffle consumer plugin, add a new service key to the `yarn.nodemanager.aux-services` property. For example, for a service with the name `mapred.shufflex`, the property defining the corresponding class must be `yarn.nodemanager.aux-services.mapreduce_shufflex.class`.

To change the default sorting method, set a new value for the output collector (`mapreduce.job.map.output.collector.class`) property.

You can specify a comma-separated list of output collector implementations. In this case, the map task will attempt to instantiate each in turn until one of the implementations successfully initializes. This feature can be useful if a given output collector implementation is only compatible with certain types of keys or values.

Setting a new value for a single node's environment will change the sorting method on a per-job basis. To set a new output collector for all nodes, edit the same property in the *mapred-site.xml* configuration file via ADCM.

To edit *mapred-site.xml* on all hosts via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at ***YARN***.
3. Toggle the ***Show advanced*** option and find ***Custom mapred-site.xml***.
4. Open the parameter drop-down list and select ***Add property***.
5. Enter the parameter name and the desired value.
6. Click ***Apply*** and confirm changes to YARN configuration by clicking ***Save***.
7. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

Encrypted shuffle in MapReduce

Elena Kostyuchenko

Contents

To Table of Contents

Overview

MapReduce configuration

YARN configuration

Keystore and truststore settings

Activation

Overview

In Hadoop, data encryption is possible during the shuffle phase in MapReduce and YARN. This feature employs HTTPS with optional client authentication, also referred to as bi-directional HTTPS or HTTPS with client certificates.

CAUTION

Encrypted shuffle significantly affects cluster performance. Consider reserving additional resources when using encrypted shuffle.

The encrypted shuffle function includes several optional security settings for your cluster:

- Configuration settings that switch the shuffle between HTTP and HTTPS.
- Configuration settings that define the keystore and truststore properties (location, type, passwords).
- A method to reload truststores across the cluster when a node is added or removed.

To enable encrypted shuffle for MapReduce and YARN, you need to update their configuration files and provide SSL certificate information in keystore and truststore settings.

The MapReduce and YARN configuration files (*core-site.xml* and *mapred-site.xml*) can be edited via ADCM. The keystore and truststore settings have to be updated manually.

Client certificates

The client certificate keystore file, which contains the private key, must be accessible to all users submitting jobs to the cluster. This means a rogue job could potentially access these keystore files and use the client certificates to establish a secure connection with a shuffle server.

A better safeguard for the data is the JobToken mechanism provided by the Hadoop environment. Each job uses its unique JobToken to retrieve only the shuffle data it owns. Without a valid JobToken, a rogue job cannot access shuffle data from the shuffle server.

If your cluster necessitates client certificates, make sure that browsers connecting to the web UIs have appropriately signed certificates. If your certificates are signed by a certificate authority (CA), ensure that the complete chain of CA certificates is included in the server's keystore.

MapReduce configuration

To configure the encrypted shuffle for MapReduce via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at ***HDFS***.
3. Toggle the ***Show advanced*** option and find ***core-site.xml***.
4. Open the parameter drop-down list, select and edit the necessary properties from the table below.
5. Confirm changes to the cluster configuration by clicking ***Save***.

Encrypted shuffle properties

Parameter	Description	Default value
hadoop.ssl.require.client.cert	The boolean value representing whether client certificates are required	false
hadoop.ssl.hostname.verifier	The type of hostname verification to use. Accepts the following values: `DEFAULT`, `STRICT`, `STRICT_IE6`, `DEFAULT_AND_LOCALHOST`, and `ALLOW_ALL`	DEFAULT
hadoop.ssl.keystores.factory.class	The KeyStoresFactory implementation to use. Currently, only the default implementation is available. It uses the properties located in *ssl-server.xml* and *ssl-client.xml*	org.apache.hadoop.security.ssl.FileBased KeyStoresFactory
hadoop.ssl.server.conf	The resource file with the SSL server keystore information	ssl-server.xml
hadoop.ssl.client.conf	The resource file with the SSL client keystore information	ssl-client.xml
hadoop.ssl.enabled.protocols	The supported SSL protocols. This parameter is used only by DataNode HTTP Server	TLSv1.2

The `hadoop.ssl.hostname.verifier` parameter supports the following verification types:

- `DEFAULT` — the hostname must match the first common name (CN) or any of the subject alt names (SAN). Names can include wildcards, for example, a hostname **.example.com* will match all subdomains, including *beta.test.example.com*.
- `DEFAULT_AND_LOCALHOST` — same as `DEFAULT` but also allows all hostnames of the type: `localhost`, `localhost.example`, or `127.0.0.1`.
- `STRICT` — same as `DEFAULT` but allows only wildcards of the same level. For instance, a hostname with a wildcard **.example.com* will only match subdomains at the same level, such as *test.example.com*, but not *beta.test.example.com*.
- `STRICT_IE6` — same as `STRICT`, but it also permits hostnames that match any of the common names (CN) within the server's X.509 certificate, not just the first one.
- `ALLOW_ALL` — disables the hostname verifier mechanism.

YARN configuration

You can enable the encrypted shuffle for YARN as well. It works only if the MapReduce parameters have been already configured.

To configure the encrypted shuffle for YARN via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at ***YARN***.
3. Toggle the ***Show advanced*** option.
4. Find the ***mapreduce.shuffle.ssl.enabled*** parameter and set its value to `true`.
5. Confirm changes to the YARN configuration by clicking ***Save***.

Keystore and truststore settings

Enabling the encrypted shuffle requires changing the keystore and truststore settings used by the MapReduce shuffle service.

These settings are located in the *ssl-server.xml* and *ssl-client.xml* files, in the *etc/hadoop/conf/* directory on your cluster hosts.

After you edit the properties, make sure that the following is true:

- the `mapred` user is the owner of both *ssl-server.xml* and *ssl-client.xml* files;
- the `mapred` user has exclusive read access to the SSL server configuration file;
- the `mapred` user has default permissions for the SSL client configuration file.

SSL client and server properties

Client parameter	Server parameter	Description	Default value
ssl.client.keystore.type	ssl.server.keystore.type	Keystore file type	jks
ssl.client.keystore.location	ssl.server.keystore.location	Keystore file location. The `mapred` must be the owner of this file and have exclusive read access to it	—
ssl.client.keystore.password	ssl.server.keystore.password	Keystore file password	—
ssl.client.truststore.type	ssl.server.truststore.type	Truststore file type	jks
ssl.client.truststore.location	ssl.server.truststore.location	Truststore file location. The `mapred` must be the owner of this file and have exclusive read access to it	—
ssl.client.truststore.password	ssl.server.truststore.password	Truststore file password	—
ssl.client.truststore.reload.interval	ssl.server.truststore.reload.interval	Defines, how often the truststores reload their configuration, in milliseconds	10000

If you copy a new truststore file over the old one, the system will re-read it, replacing the old certificates with the new ones.

This mechanism is beneficial for adding or removing nodes from the cluster or trusted clients. In such cases, the client or NodeManager certificate is added to (or removed from) all the truststore files in the system. The new configuration will then be implemented without the need to restart the NodeManagers.

Default ssl-client.xml

```xml
<?xml version="1.0"?>
<configuration>
        <property>
                <name>ssl.client.keystore.type</name>
                <value>jks</value>
        </property>
        <property>
                <name>ssl.client.truststore.reload.interval</name>
                <value>10000</value>
        </property>
        <property>
                <name>ssl.client.truststore.type</name>
                <value>jks</value>
        </property>
</configuration>
```

Default ssl-server.xml

```xml
<?xml version="1.0"?>
<configuration>
        <property>
                <name>ssl.server.keystore.type</name>
                <value>jks</value>
        </property>
        <property>
                <name>ssl.server.truststore.reload.interval</name>
                <value>10000</value>
        </property>
        <property>
                <name>ssl.server.truststore.type</name>
                <value>jks</value>
        </property>
</configuration>
```

Activation

After you have configured all necessary properties, activate the encrypted shuffle by restarting all NodeManagers via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services***.
3. Expand the ***Actions*** drop-down menu for ***YARN*** and select ***Restart***
4. Make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

Monitoring service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Monitoring service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Monitoring service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Monitoring components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM
Reconfigure and restart	Updates the configuration and restarts the service
Update scrape config files	Updates scrape configuration files, which define targets (ADH services) for collecting metrics by Prometheus. Use this action (or restart the Monitoring service) when you add new services or redistribute service components of an ADH cluster

Connect to Ozone via CLI

Sergei Tikhomirov

Contents

To Table of Contents

- Overview
- Examples
  - Volume commands
  - Bucket commands
  - ACL management

Overview

Ozone has a CLI tool that allows access to some specific functionality that can't be performed otherwise. These commands are normally performed once during the setup, like volume creation or ACL management.

The general way to invoke an Ozone shell command is as follows:

```
$ ozone sh <object> <action> [params]
```

where:

- `<object>` is an Ozone object such as bucket, key, prefix, snapshot, user, token, or volume.
- `<action>` is an action to perform with an object. For example, to create a volume, the action would be `create`.
- `[params]` are additional parameters that some actions require.

The CLI tool offers a help manual at both the object and action level. That is, to see possible actions for an object, call the following command:

```
$ ozone sh <object> --help
```

And to see the necessary parameters for an action, call:

```
$ ozone sh <object> <action> --help
```

NOTE

Currently, there's no online documentation for the Ozone shell objects and actions. It's recommended to refer to the help manual of the tool itself.

Examples

Volume commands

A functionality that is unique to the CLI tool includes the ability to create a volume with a quota. To do this, run the following command:

```
$ ozone sh volume create <volume_name> --quota=<quota_in_bytes>
```

To check if the volume was created successfully with the right quota, list all the volumes as follows:

```
$ ozone sh volume ls
```

This command returns a list of volumes with some information about them:

```
[ {
  "metadata" : { },
  "name" : "vol1",
  "admin" : "sergei",
  "owner" : "sergei",
  "quotaInBytes" : 1073741824,
  "quotaInNamespace" : -1,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-13T13:20:43.173Z",
  "modificationTime" : "2025-01-13T13:20:43.173Z",
  "acls" : [ {
    "type" : "USER",
    "name" : "sergei",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  }, {
    "type" : "GROUP",
    "name" : "sergei",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  } ],
  "refCount" : 0
} ]
```

Bucket commands

An exclusive ability to create buckets with encryption keys can be utilized with the following command:

```
$ ozone sh bucket create -k <key_name> <URI>
```

where:

- `<key_name>` is the name of an encryption key that was created with the `hadoop key create` command.
- `<URI>` is a bucket name that includes a volume name (e.g. `/vol1/bucket1`).

You can also control the bucket replication settings:

```
$ ozone sh bucket set-replication-config -t=<type> -r=<replication> <value>
```

where:

- `<type>` is a replication type. The supported types are `RATIS` and `EC`.
- `<replication>` is a replication definition. The valid values are based on the replication type. For RATIS it may be `ONE` or `THREE`, while for EC it should follow the `CODEC-DATA-PARITY-CHUNK_SIZE` pattern, e.g. `rs-3-2-1024k`.
- `<value>` is the URI of a bucket.

Consider the following command:

```
$ ozone sh bucket set-replication-config -t=RATIS -r=ONE vol1/bucket1
```

It will add the following information to the bucket:

```
"replicationConfig" : {
  "replicationFactor" : "ONE",
  "requiredNodes" : 1,
  "replicationType" : "RATIS"
}
```

ACL management

The CLI tool allows you to add and remove ACL for volumes and buckets. Below, you can find a sample command that adds an ACL to a volume:

```
$ ozone sh volume addacl -s=<URI> -a=<type>:<entity>:<permissions> <value>
```

where:

- `<store>` is a type of store — `OZONE` or `S3`.
- `<type>` is a system entity type like `user` or `group`.
- `<entity>` is a user or a group the permissions for which you want to control.
- `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available:
  - `r` — read the resource data.
  - `w` — write data to the resource.
  - `c` — create objects at the resource.
  - `d` — delete objects at the resource.
  - `l` — list objects stored at the resource.
  - `x` — read the access control list of the resource.
  - `y` — edit the access control list of the resource.
  - `a` — all of the above.
  - `n` — no permissions.
- `<value>` is the URI of a volume/bucket.

Consider the following volume before an ACL addition:

```
[ {
  "metadata" : { },
  "name" : "vol1",
  "admin" : "s_tikhomirov_krb1",
  "owner" : "s_tikhomirov_krb1",
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-22T21:23:28.796Z",
  "modificationTime" : "2025-01-22T21:23:28.796Z",
  "acls" : [ {
    "type" : "USER",
    "name" : "s_tikhomirov_krb1",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  } ],
  "refCount" : 0
} ]
```

Here, an ACL is present only for the user who has created it. Running the `addacl` command with the `-a=user:sergei:rwcdlxy` option adds the corresponding ACL:

```
[ {
  "metadata" : { },
  "name" : "vol1",
  "admin" : "s_tikhomirov_krb1",
  "owner" : "s_tikhomirov_krb1",
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-22T21:23:28.796Z",
  "modificationTime" : "2025-01-22T21:53:56.260Z",
  "acls" : [ {
    "type" : "USER",
    "name" : "s_tikhomirov_krb1",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  }, {
    "type" : "USER",
    "name" : "sergei",
    "aclScope" : "ACCESS",
    "aclList" : [ "READ", "WRITE", "CREATE", "LIST", "DELETE", "READ_ACL", "WRITE_ACL" ]
  } ],
  "refCount" : 0
} ]
```

You can remove an ACL by calling the `removeacl` command:

```
$ ozone sh volume removeacl -s=OZONE -a=user:sergei:rwcdlxy vol1
```

Ozone UIs overview

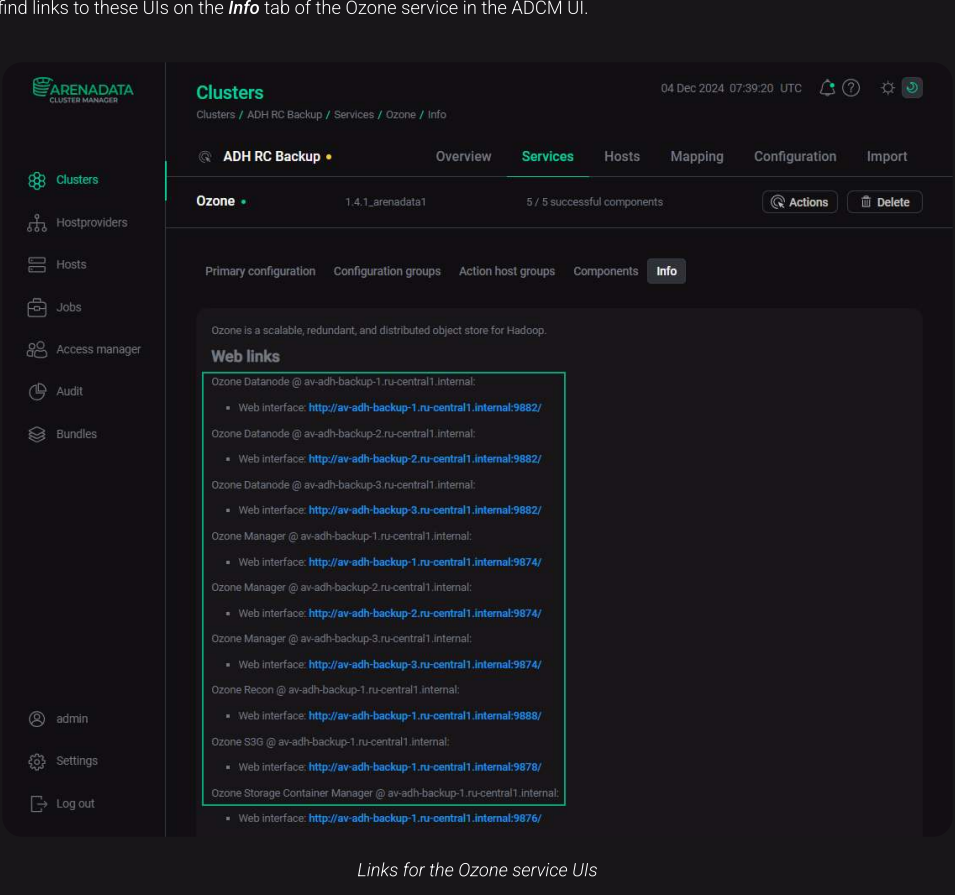
Vladimir Adamenkov

Contents

To Table of Contents

- Ozone Datanode UI
  - Overview
  - Metrics
  - Configuration
  - Documentation
  - Common tools
- Ozone Manager UI
  - Overview
  - Metrics
  - Configuration
  - Documentation
  - Common tools
- Ozone Recon UI
  - Overview
  - Volumes
  - Buckets
  - Datanodes
  - Pipelines
  - Containers
  - Insights
  - Disk Usage
- Ozone S3G UI
- Ozone Storage Container Manager UI
  - Overview
  - Metrics
  - Configuration
  - Documentation
  - Common tools

Each Ozone component has its own web UI which displays diagnostic and status information. If a component is distributed among several hosts, then each of those hosts will have a UI for that component. You can find links to these UIs on the ***Info*** tab of the Ozone service in the ADCM UI.



Ozone is a scalable, redundant, and distributed object store for Hadoop.

Web links

Ozone Datanode @ av-adh-backup-1.ru-central1.internal:
- Web interface: http://av-adh-backup-1.ru-central1.internal:9882/

Ozone Datanode @ av-adh-backup-2.ru-central1.internal:
- Web interface: http://av-adh-backup-2.ru-central1.internal:9882/

Ozone Datanode @ av-adh-backup-3.ru-central1.internal:
- Web interface: http://av-adh-backup-3.ru-central1.internal:9882/

Ozone Manager @ av-adh-backup-1.ru-central1.internal:
- Web interface: http://av-adh-backup-1.ru-central1.internal:9874/

Ozone Manager @ av-adh-backup-2.ru-central1.internal:
- Web interface: http://av-adh-backup-2.ru-central1.internal:9874/

Ozone Manager @ av-adh-backup-3.ru-central1.internal:
- Web interface: http://av-adh-backup-3.ru-central1.internal:9874/

Ozone Recon @ av-adh-backup-1.ru-central1.internal:
- Web interface: http://av-adh-backup-1.ru-central1.internal:9888/

Ozone S3G @ av-adh-backup-1.ru-central1.internal:
- Web interface: http://av-adh-backup-1.ru-central1.internal:9878/

Ozone Storage Container Manager @ av-adh-backup-1.ru-central1.internal:
- Web interface: http://av-adh-backup-1.ru-central1.internal:9876/

Links for the Ozone service UIs

A component web UI URL has the following form: `http://<node-hostname>:<component-port>/`.

Example: `http://adh-host1.ru-central1.internal:9876/`.

For secure clusters, use `https://` instead of `http://`.

The default web server ports are listed in the table below.

Ozone components UIs ports

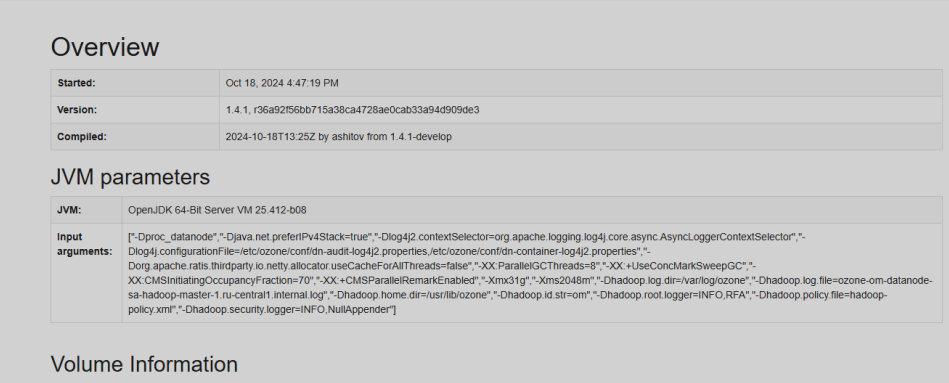
Component	Port
Ozone Datanode	9880
Ozone Manager	9874
Ozone Recon	9888
Ozone S3G	9878
Ozone Storage Container Manager	9876

Ozone Datanode UI

Overview

Start page of the Ozone Datanode UI is the ***Overview*** page. It contains the following information:

- Service details:
  - time started;
  - version and checksum;
  - time compiled, user, and source branch.
- Java virtual machine details: version and input arguments string.
- Volume information:
  - path to the volume directory;
  - storage type;
  - volume type;
  - disk usage statistics;
  - volume state.



Started:	Oct 18, 2024 4:47:19 PM
Version:	1.4.1, r36a92f56bb715a38ca4728ae0cab33a94d909de3
Compiled:	2024-10-18T13:25Z by ashitov from 1.4.1-develop

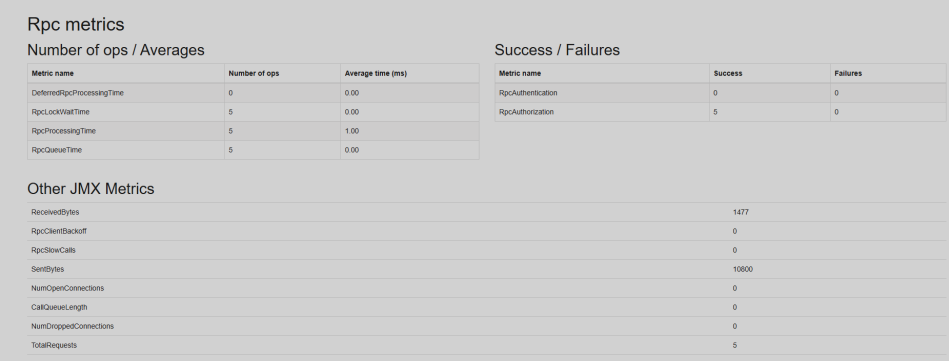
Directory	Storage Type	Volume Type	Used Space	Available Space	Reserved	Total Capacity	State
/srv/ozone/data/hdds	DISK	DATA_VOLUME	337.5 GB	662.39 GB	102.4 MB	999.99 GB	NORMAL

The Datanode UI Overview page

Metrics

The ***Metrics*** group contains the ***Rpc metrics*** page. This page contains the following information:

- Number of executed operations and their average times.
- Success and failure statistics on RPC authentication and authorization metrics.
- Other Java management extensions (JMX) metrics.

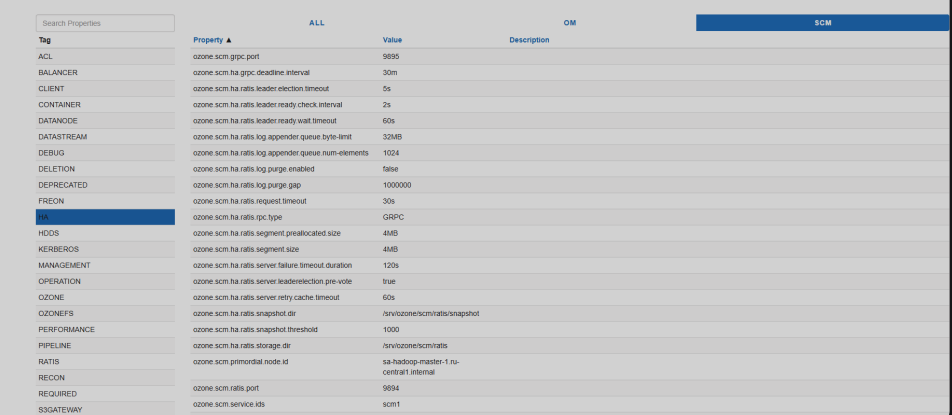


The Rpc metrics page

Configuration

The ***Configuration*** page contains a list of all configuration parameters pertaining to the Ozone service functioning. You can search for parameters by their name or value in the ***Search Properties*** field. Also, you can filter the parameter list by highlighting one or more tags listed in the leftmost part of the screen. To view the Ozone Manager parameters, click ***OM*** in the top part of the screen, or click ***SCM*** to view the Storage Container Manager parameters. Moreover, the parameter list is sortable by name and by value.

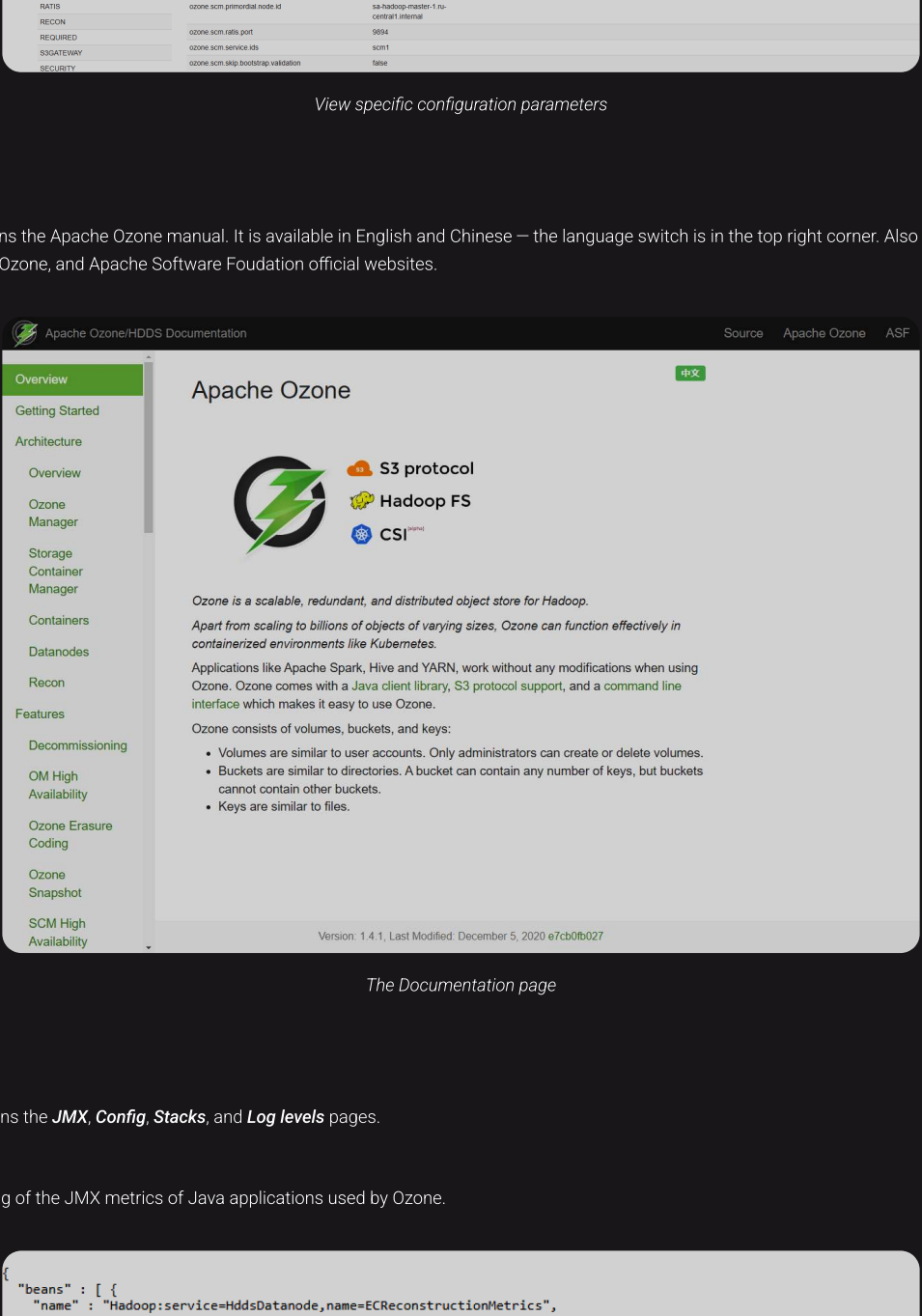
For example, to view the configuration parameter values related to the high availability mode in the Storage Container Manager component, highlight the ***HA*** tag and select the ***SCM*** tab.



View specific configuration parameters

Documentation

The ***Documentation*** page contains the Apache Ozone manual. It is available in English and Chinese — the language switch is in the top right corner. Also there are links to the Ozone source code repository, Apache Ozone, and Apache Software Foudation official websites.



Ozone is a scalable, redundant, and distributed object store for Hadoop.

Apart from scaling to billions of objects of varying sizes, Ozone can function effectively in containerized environments like Kubernetes.

Applications like Apache Spark, Hive and YARN, work without any modifications when using Ozone. Ozone comes with a Java client library, S3 protocol support, and a command line interface which makes it easy to use Ozone.

Ozone consists of volumes, buckets, and keys:

- Volumes are similar to user accounts. Only administrators can create or delete volumes.
- Buckets are similar to directories. A bucket can contain any number of keys, but buckets cannot contain other buckets.
- Keys are similar to files.

Version: 1.4.1, Last Modified: December 5, 2020 e7cb0fb027

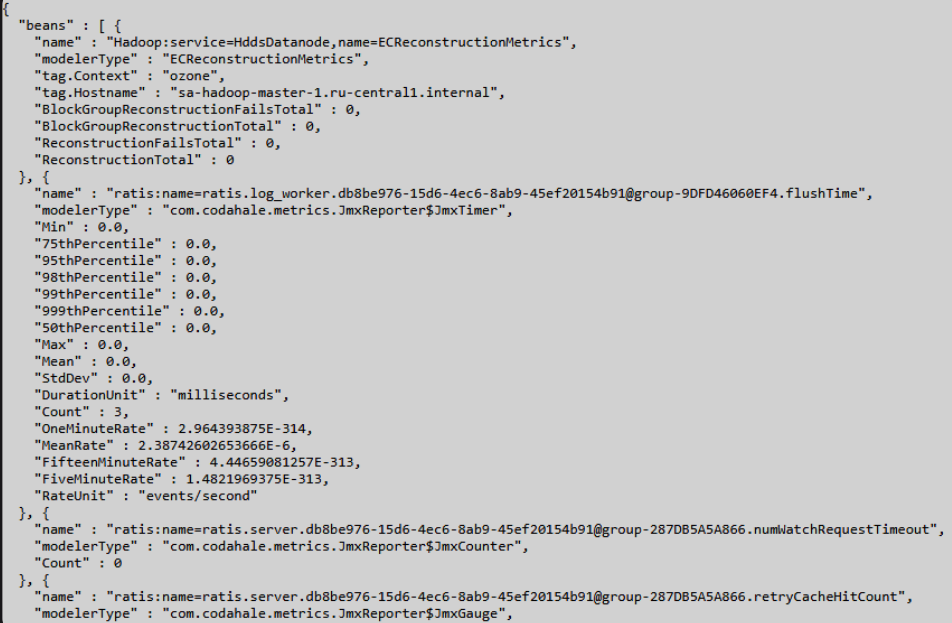
The Documentation page

Common tools

The ***Common tools*** group contains the ***JMX***, ***Config***, ***Stacks***, and ***Log levels*** pages.

JMX

The ***JMX*** page displays the listing of the JMX metrics of Java applications used by Ozone.

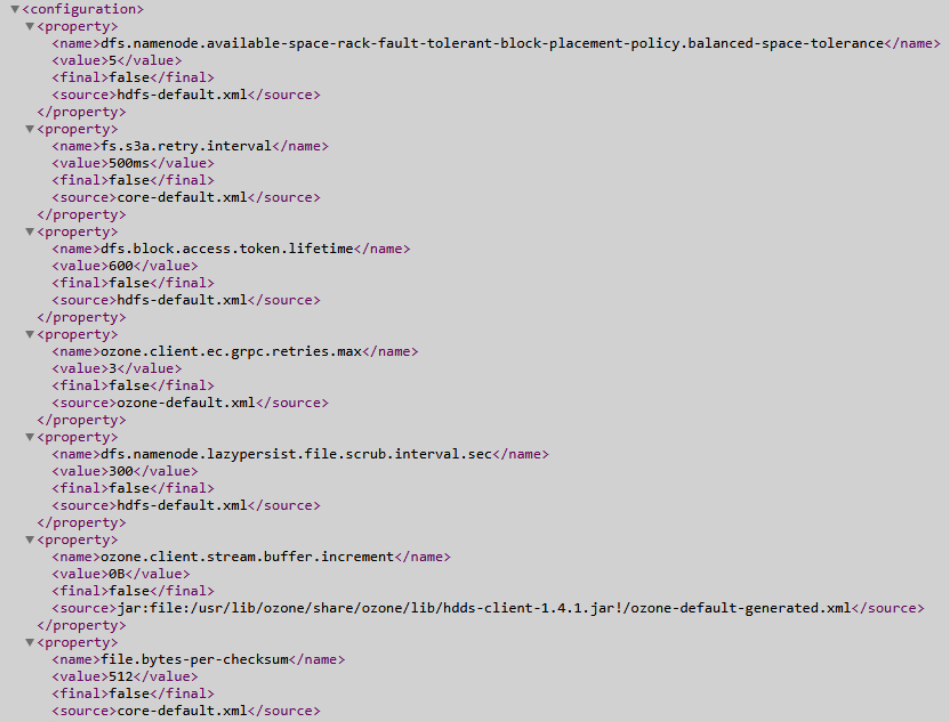


```
{
  "beans" : [ {
    "name" : "Hadoop:service=HddsDatanode,name=ECReconstructionMetrics",
    "modelerType" : "ECReconstructionMetrics",
    "tag.Context" : "ozone",
    "tag.Hostname" : "sa-hadoop-master-1.ru-central1.internal",
    "BlockGroupReconstructionFailsTotal" : 0,
    "BlockGroupReconstructionTotal" : 0,
    "ReconstructionFailsTotal" : 0,
    "ReconstructionTotal" : 0
  }, {
    "name" : "ratis:name=ratis.log_worker.db8be976-15d6-4ec6-8ab9-45ef20154b91@group-9DFD46060EF4.flushTime",
    "modelerType" : "com.codahale.metrics.JmxReporter$JmxTimer",
    "Min" : 0.0,
    "75thPercentile" : 0.0,
    "95thPercentile" : 0.0,
    "98thPercentile" : 0.0,
    "99thPercentile" : 0.0,
    "999thPercentile" : 0.0,
    "50thPercentile" : 0.0,
    "Max" : 0.0,
    "Mean" : 0.0,
    "StdDev" : 0.0,
    "DurationUnit" : "milliseconds",
    "Count" : 3,
    "OneMinuteRate" : 2.964393875E-314,
    "MeanRate" : 2.38742602653666E-6,
    "FifteenMinuteRate" : 4.44659081257E-313,
    "FiveMinuteRate" : 1.4821969375E-313,
    "RateUnit" : "events/second"
  }, {
    "name" : "ratis:name=ratis.server.db8be976-15d6-4ec6-8ab9-45ef20154b91@group-287DB5A5A866.numWatchRequestTimeout",
    "modelerType" : "com.codahale.metrics.JmxReporter$JmxCounter",
    "Count" : 0
  }, {
    "name" : "ratis:name=ratis.server.db8be976-15d6-4ec6-8ab9-45ef20154b91@group-287DB5A5A866.retryCacheHitCount",
    "modelerType" : "com.codahale.metrics.JmxReporter$JmxGauge",
    "Value" : 0
```

Excerpt from the JMX page listing

Config

The ***Config*** page contains the listing of all configuration parameters pertaining to the Ozone service functioning in XML format.

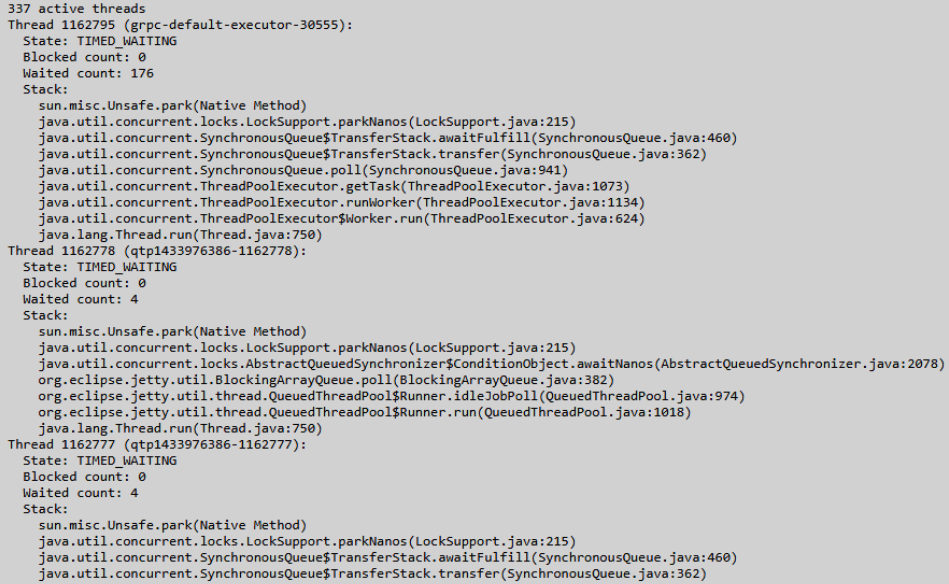


```
▼<configuration>
  ▼<property>
     <name>dfs.namenode.available-space-rack-fault-tolerant-block-placement-policy.balanced-space-tolerance</name>
     <value>5</value>
     <final>false</final>
     <source>hdfs-default.xml</source>
   </property>
  ▼<property>
     <name>fs.s3a.retry.interval</name>
     <value>500ms</value>
     <final>false</final>
     <source>core-default.xml</source>
   </property>
  ▼<property>
     <name>dfs.block.access.token.lifetime</name>
     <value>600</value>
     <final>false</final>
     <source>hdfs-default.xml</source>
   </property>
  ▼<property>
     <name>ozone.client.ec.grpc.retries.max</name>
     <value>3</value>
     <final>false</final>
     <source>ozone-default.xml</source>
   </property>
  ▼<property>
     <name>dfs.namenode.lazypersist.file.scrub.interval.sec</name>
     <value>300</value>
     <final>false</final>
     <source>hdfs-default.xml</source>
   </property>
  ▼<property>
     <name>ozone.client.stream.buffer.increment</name>
     <value>0B</value>
     <final>false</final>
     <source>jar:file:/usr/lib/ozone/share/ozone/lib/hdds-client-1.4.1.jar!/ozone-default-generated.xml</source>
   </property>
  ▼<property>
     <name>file.bytes-per-checksum</name>
     <value>512</value>
     <final>false</final>
     <source>core-default.xml</source>
   </property>
```

Excerpt from the Config page listing

Stacks

The ***Stacks*** page contains the list of thread stacks of the JVM used by the Ozone service.



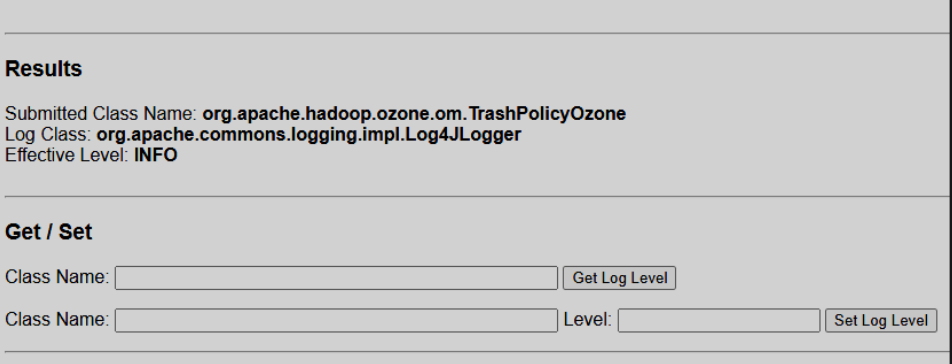
```
Process Thread Dump:
337 active threads
Thread 1162795 (grpc-default-executor-30555):
  State: TIMED_WAITING
  Blocked count: 0
  Waited count: 176
  Stack:
    sun.misc.Unsafe.park(Native Method)
    java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    java.util.concurrent.SynchronousQueue$TransferStack.awaitFulfill(SynchronousQueue.java:460)
    java.util.concurrent.SynchronousQueue$TransferStack.transfer(SynchronousQueue.java:362)
    java.util.concurrent.SynchronousQueue.poll(SynchronousQueue.java:941)
    java.util.concurrent.ThreadPoolExecutor.getTask(ThreadPoolExecutor.java:1073)
    java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1134)
    java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    java.lang.Thread.run(Thread.java:750)
Thread 1162778 (qtp1433976386-1162778):
  State: TIMED_WAITING
  Blocked count: 0
  Waited count: 4
  Stack:
    sun.misc.Unsafe.park(Native Method)
    java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    java.util.concurrent.locks.AbstractQueuedSynchronizer$ConditionObject.awaitNanos(AbstractQueuedSynchronizer.java:2078)
    org.eclipse.jetty.util.BlockingArrayQueue.poll(BlockingArrayQueue.java:382)
    org.eclipse.jetty.util.thread.QueuedThreadPool$Runner.idleJobPoll(QueuedThreadPool.java:974)
    org.eclipse.jetty.util.thread.QueuedThreadPool$Runner.run(QueuedThreadPool.java:1018)
    java.lang.Thread.run(Thread.java:750)
Thread 1162777 (qtp1433976386-1162777):
  State: TIMED_WAITING
  Blocked count: 0
  Waited count: 4
  Stack:
    sun.misc.Unsafe.park(Native Method)
    java.util.concurrent.locks.LockSupport.parkNanos(LockSupport.java:215)
    java.util.concurrent.SynchronousQueue$TransferStack.awaitFulfill(SynchronousQueue.java:460)
    java.util.concurrent.SynchronousQueue$TransferStack.transfer(SynchronousQueue.java:362)
    java.util.concurrent.SynchronousQueue.poll(SynchronousQueue.java:941)
    org.eclipse.jetty.util.thread.ReservedThreadExecutor$ReservedThread.reservedWait(ReservedThreadExecutor.java:324)
    org.eclipse.jetty.util.thread.ReservedThreadExecutor$ReservedThread.run(ReservedThreadExecutor.java:399)
    org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedThreadPool.java:883)
    org.eclipse.jetty.util.thread.QueuedThreadPool$Runner.run(QueuedThreadPool.java:1034)
    java.lang.Thread.run(Thread.java:750)
```

Excerpt from the Stacks page listing

Log levels

The ***Log levels*** page allows you to view and set the logging levels of classes used by Ozone.

To view the logging level of a class, paste its name in the first ***Class Name*** field and click ***Get Log Level***.



Log Level

Results

Submitted Class Name: **org.apache.hadoop.ozone.om.TrashPolicyOzone**
Log Class: **org.apache.commons.logging.impl.Log4JLogger**
Effective Level: **INFO**

Get / Set

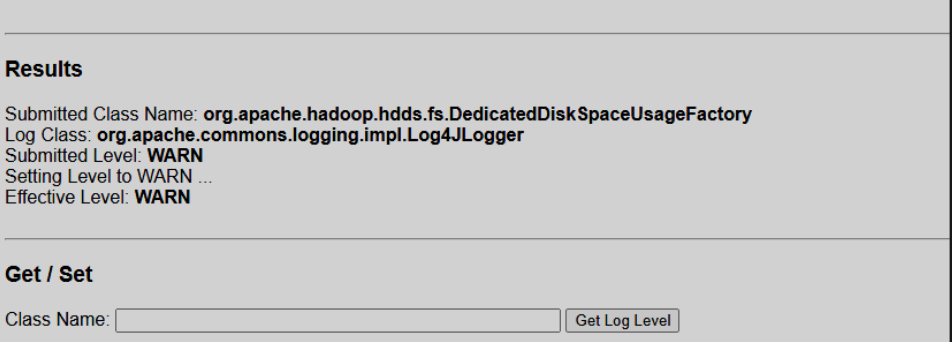
Class Name: [Get Log Level]

Class Name: Level: [Set Log Level]

Hadoop, 2024.

View logging level of a class

To set a logging level for a class, paste its name in the second ***Class Name*** field, then specify the logging level value in the ***Level*** field, and click ***Set Log Level***.



Log Level

Results

Submitted Class Name: **org.apache.hadoop.hdds.fs.DedicatedDiskSpaceUsageFactory**
Log Class: **org.apache.commons.logging.impl.Log4JLogger**
Submitted Level: **WARN**
Setting Level to WARN ...
Effective Level: **WARN**

Get / Set

Class Name: [Get Log Level]

Class Name: Level: [Set Log Level]

Hadoop, 2024.

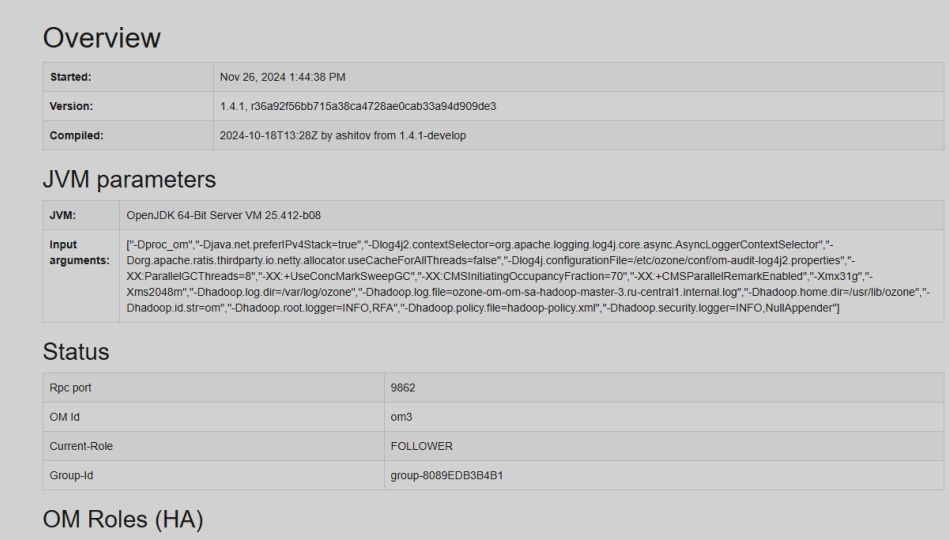
Set logging level of a class

Ozone Manager UI

Overview

Start page of the Ozone Manager UI is the ***Overview*** page. It contains the following information:

- Service details:
  - time started;
  - version and checksum;
  - time compiled, user, and source branch.
- Java virtual machine details: version and input arguments string.
- Status of the current Ozone Manager node:
  - number of the RPC port;
  - ID of the current Ozone Manager node;
  - role of the current Ozone Manager node (leader or follower);
  - ID of the group.
- Ozone Manager node roles in the high availability mode. Current node is highlighted green.
- Metadata volume information: home directories for Ratis and RocksDB.



Started:	Nov 26, 2024 1:44:38 PM
Version:	1.4.1, r36a92f56bb715a38ca4728ae0cab33a94d909de3
Compiled:	2024-10-18T13:28Z by ashitov from 1.4.1-develop

Rpc port	9862
OM Id	om3
Current-Role	FOLLOWER
Group-Id	group-8089EDB3B481

Host Name	Node ID	Ratis Port	Role
sa-hadoop-master-1.ru-central1.internal	om1	9872	LEADER
sa-hadoop-master-2.ru-central1.internal	om2	9872	FOLLOWER
sa-hadoop-master-3.ru-central1.internal	om3	9872	FOLLOWER

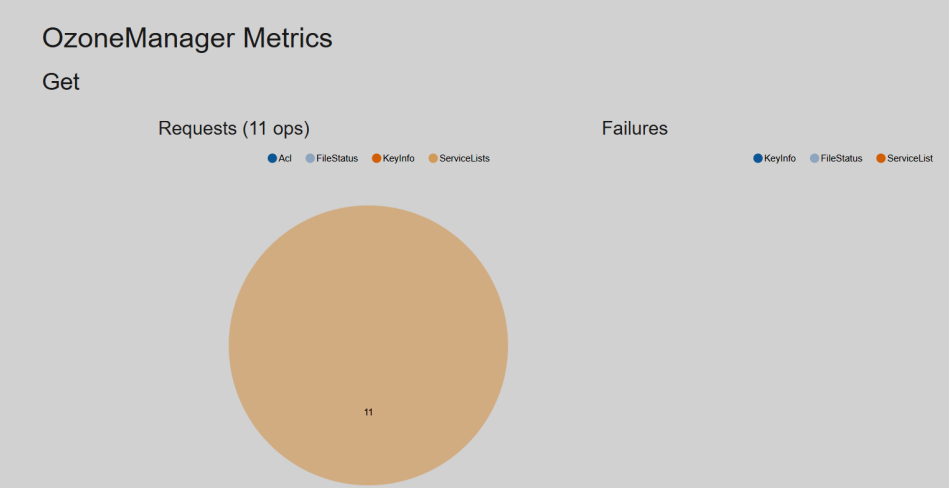
The Ozone Manager UI Overview page

Metrics

The ***Metrics*** group contains the ***Rpc metrics*** and the ***OM metrics*** pages.

OM metrics

This page contains pie charts of requests and failures statistics for various JMX properties, as well as a table of some other properties at the bottom of the page.

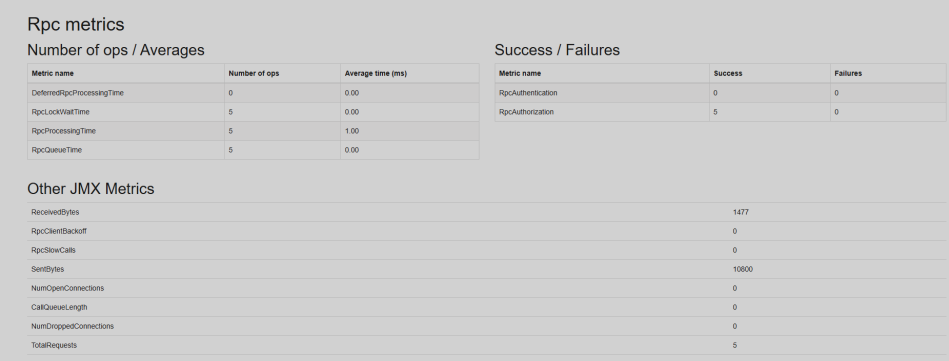


The OM metrics page

Rpc metrics

This page contains the following information:

- Number of executed operations and their average times.
- Success and failure statistics on RPC authentication and authorization metrics.
- Other JMX metrics.



The Rpc metrics page

Configuration

The ***Configuration*** page is identical to the Configuration page of the Ozone Datanode UI.

Documentation

The ***Documentation*** page is identical to the Documentation page of the Ozone Datanode UI.

Common tools

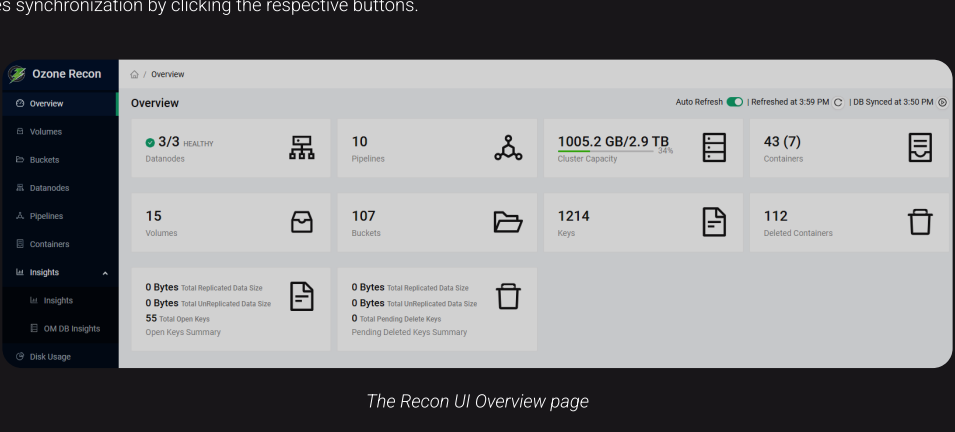
The ***Common tools*** group pages are identical to the corresponding pages of the Common tools group of the Ozone Datanode UI.

Ozone Recon UI

Overview

Start page of the Ozone Recon UI is the ***Overview*** page. It contains ten graphical blocks with some quick statistics. Some of the blocks are clickable and lead to the corresponding pages of the UI. The ***Cluster Capacity*** block has a pop-up legend appearing on mouse hover. The ***Container*** block also has a pop-up tip.

This page auto-refreshes by default. You can turn the auto-refresh off by clicking on its switch in the top right area of the page. Also, in the same area, you can refresh the page manually or invoke the databases synchronization by clicking the respective buttons.



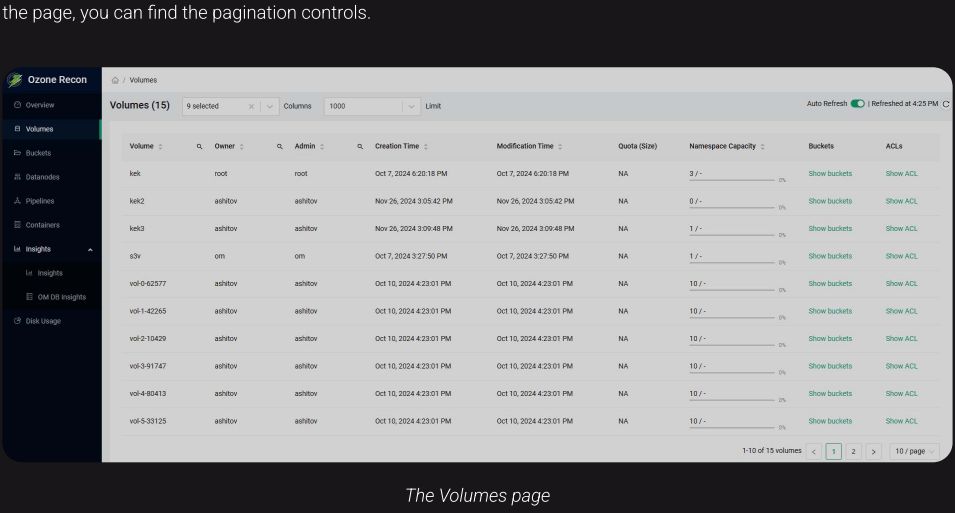
The Recon UI Overview page

Volumes

The ***Volumes*** page contains details of the volumes existing in the Ozone storage. The information is presented as a table that has the following columns:

- ***Volume*** — name of the volume.
- ***Owner*** — username of the volume owner.
- ***Admin*** — username of the volume administrator.
- ***Creation Time*** — time when the volume was created.
- ***Modification Time*** — time when the volume was last modified.
- ***Quota*** — quota value for the volume or buckets, if specified.
- ***Namespace Capacity*** — statistics of namespace usage by the volume: used, remaining, and the percentage of used among total.
- ***Buckets*** — link to the ***Buckets*** page with the filter of this volume applied to the list.
- ***ACLs*** — button that allows viewing the access control list for the volume.

The table is sortable by all columns except ***Quota***, ***Buckets***, and ***ACLs***. You can arbitrarily select the columns to be displayed in the ***Columns*** drop-down list in the top area of the page. You can also specify the limit of rows in the ***Limit*** drop-down list in the same area. Moreover, you can toggle auto-refresh and refresh the page manually the same way as in the ***Overview*** page. In the bottom of the page, you can find the pagination controls.



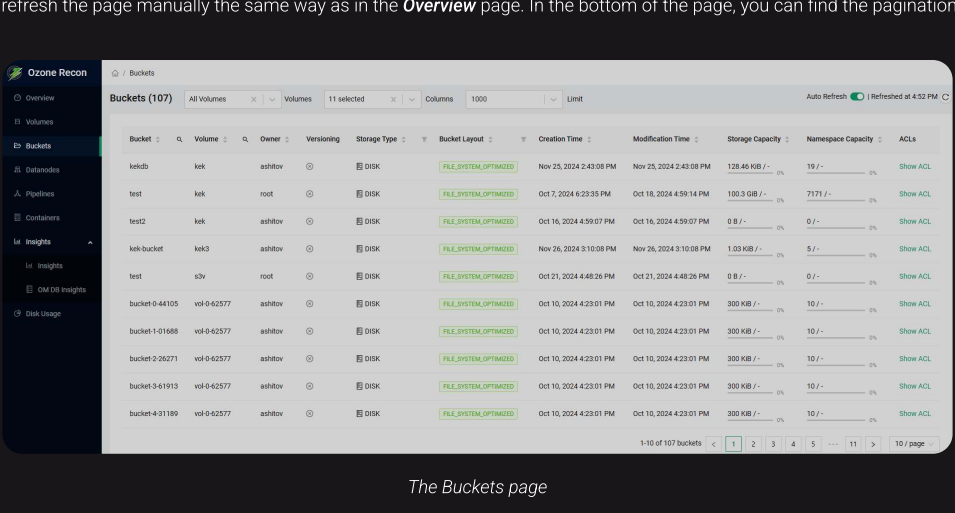
The Volumes page

Buckets

The ***Buckets*** page contains details of the buckets stored in the volumes. The information is presented as a table that has the following columns:

- ***Bucket*** — name of the bucket.
- ***Volume*** — name of the volume that the bucket belongs to.
- ***Owner*** — username of the bucket owner.
- ***Versioning*** — whether versioning is used for the bucket.
- ***Storage Type*** — type of storage where the bucket is located. Possible values: `SSD`, `DISK`, `ARCHIVE`, or `RAM_DISK`.
- ***Bucket Layout*** — type of the bucket layout. Possible values: `FILE_SYSTEM_OPTIMIZED`, `OBJECT_STORE`, or `LEGACY`.
- ***Creation Time*** — time when the bucket was created.
- ***Modification Time*** — time when the bucket was last modified.
- ***Storage Capacity*** — storage size occupied by the bucket, remaining storage size, and the percentage of used among total.
- ***Namespace Capacity*** — statistics of namespace usage by the bucket: used, remaining, and the percentage of used among total.
- ***Source Volume*** — volume that the bucket originates from.
- ***Source Bucket*** — bucket that the bucket originates from.
- ***ACLs*** — button that allows viewing the access control list for the bucket.

The table is sortable by all columns except ***Versioning***, ***Source Volume***, ***Source Bucket***, and ***ACLs***. You can arbitrarily select the columns to be displayed in the ***Columns*** drop-down list in the top area of the page, and filter the table by volumes using the ***Volumes*** drop-down list. You can also specify the limit of rows in the ***Limit*** drop-down list in the same area. Moreover, you can toggle auto-refresh and refresh the page manually the same way as in the ***Overview*** page. In the bottom of the page, you can find the pagination controls.



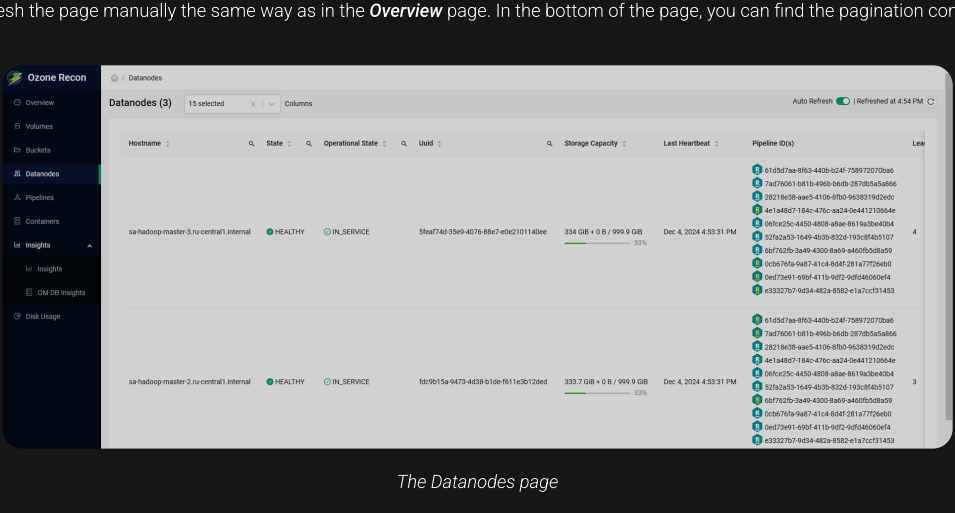
The Buckets page

Datanodes

The ***Datanodes*** page contains details of the datanodes in the Ozone cluster. The information is presented as a table that has the following columns:

- ***Hostname*** — FQDN of the datanode.
- ***State*** — health state of the datanode. Possible values: `HEALTHY`, `HEALTHY_READONLY`, `STALE`, or `DEAD`.
- ***Operational State*** — operational state of the datanode. Possible values: `IN_SERVICE`, `DECOMMISSIONING`, or `DECOMMISSIONED`.
- ***Uuid*** — UUID of the datanode.
- ***Storage Capacity*** — statistics of storage usage on the datanode: used space, remaining space, and the percentage of used among total.
- ***Last Heartbeat*** — time of the last hearthbeat obtained from the datanode.
- ***Pipeline ID(s)*** — list of IDs of the pipelines in which the datanode participates.
- ***Leader Count*** — number of pipelines in which the datanode is elected as a leader.
- ***Containers*** — number of containers on the datanode.
- ***Open Containers*** — number of open containers per pipeline.
- ***Version*** — Ozone version number installed on the datanode.
- ***Setup Time*** — time when the datanode was started.
- ***Revision*** — checksum of the Ozone distribution installed on the datanode.
- ***Build Date*** — time when the Ozone distribution installed on the datanode was built.
- ***Network Location*** — network location of the datanode.

The table is sortable by all columns except ***Pipeline ID(s)***. You can arbitrarily select the columns to be displayed in the ***Columns*** drop-down list in the top area of the page. Moreover, you can toggle auto-refresh and refresh the page manually the same way as in the ***Overview*** page. In the bottom of the page, you can find the pagination controls.



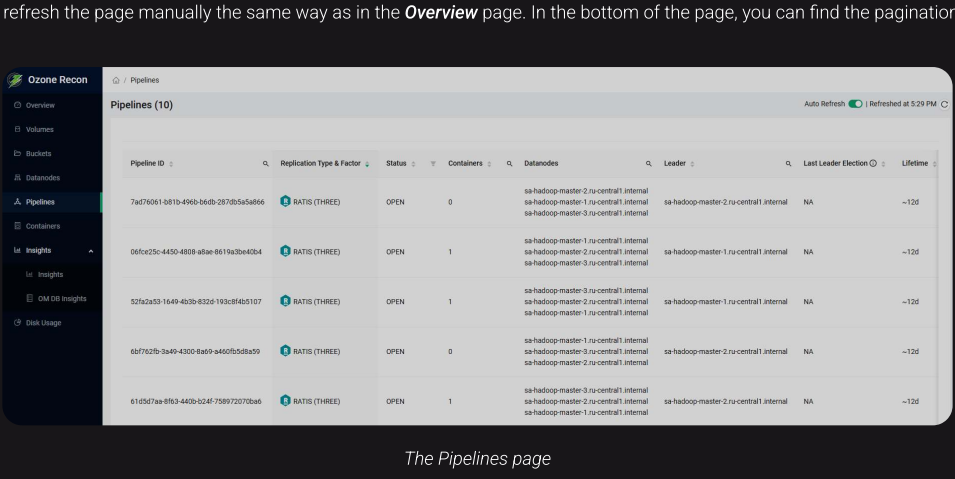
The Datanodes page

Pipelines

The ***Pipelines*** page contains details of the pipelines in the Ozone cluster. The information is presented as a table that has the following columns:

- ***Pipeline ID*** — ID of the pipeline.
- ***Replication Type & Factor*** — type of replication (`RATIS` or `STANDALONE`) and the value of replication factor.
- ***Status*** — pipeline status. Possible values: `OPEN` or `CLOSED`.
- ***Containers*** — number of containers in the pipeline.
- ***Datanodes*** — list of datanodes participating in the pipeline.
- ***Leader*** — datanode having the Ratis `LEADER` role in the pipeline.
- ***Last Leader Election*** — last time when a datanode was assigned the Ratis `LEADER` role in the pipeline.
- ***Lifetime*** — duration of the pipeline open state.
- ***No. of Elections*** — number of times when datanodes were assigned the Ratis `LEADER` role in the pipeline.

You can toggle auto-refresh and refresh the page manually the same way as in the ***Overview*** page. In the bottom of the page, you can find the pagination controls.



The Pipelines page

Containers

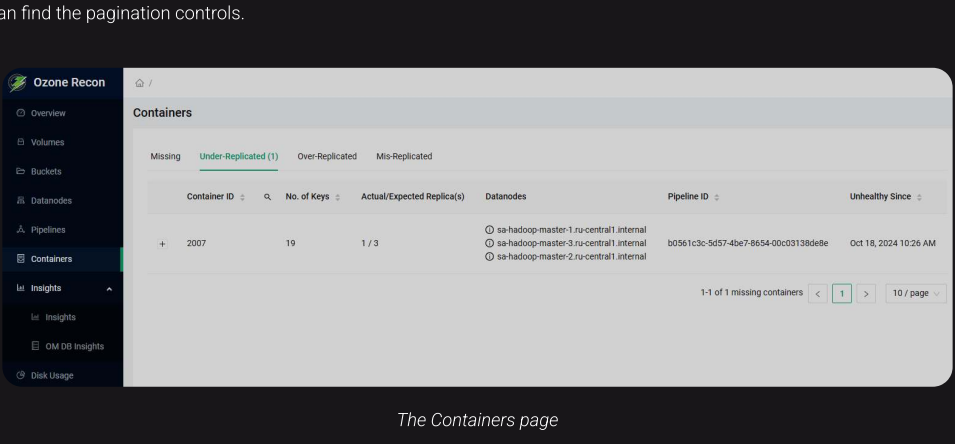
The ***Containers*** page contains details of the containers stored in the Ozone cluster. It has four tabs that contain the following information:

- ***Missing*** — containers that are not reported by any datanode (e.g. containers that were not deleted, but all datanodes that had them are down).
- ***Under-Replicated*** — containers that have less replicas then expected.
- ***Over-Replicated*** — containers that have more replicas then expected.
- ***Mis-Replicated*** — containers that have replicas not compliant with the replication policy (e.g. stored on the same datanode).

The information in each of these tabs is presented as a table that has the following columns:

- ***Container ID*** — ID of the container.
- ***No. of Keys*** — number of keys stored in the container.
- ***Actual/Expected Replica(s)*** — actual and expected numbers of replicas of the container.
- ***Datanodes*** — list of datanodes where the container replicas are expected to be stored.
- ***Pipeline ID*** — ID of the pipeline the container belongs to.
- ***Unhealthy Since*** — time when the container became missing, underreplicated, overreplicated, or misreplicated, depending on the tab.

In the bottom of the page, you can find the pagination controls.



Container ID	No. of Keys	Actual/Expected Replica(s)	Datanodes	Pipeline ID	Unhealthy Since
2007	19	1 / 3	sa-hadoop-master-1.ru-central1.internal sa-hadoop-master-3.ru-central1.internal sa-hadoop-master-2.ru-central1.internal	505613c-5057-4be7-8634-05c03138de9e	Oct 18, 2024 10:26 AM

1-1 of 1 missing containers

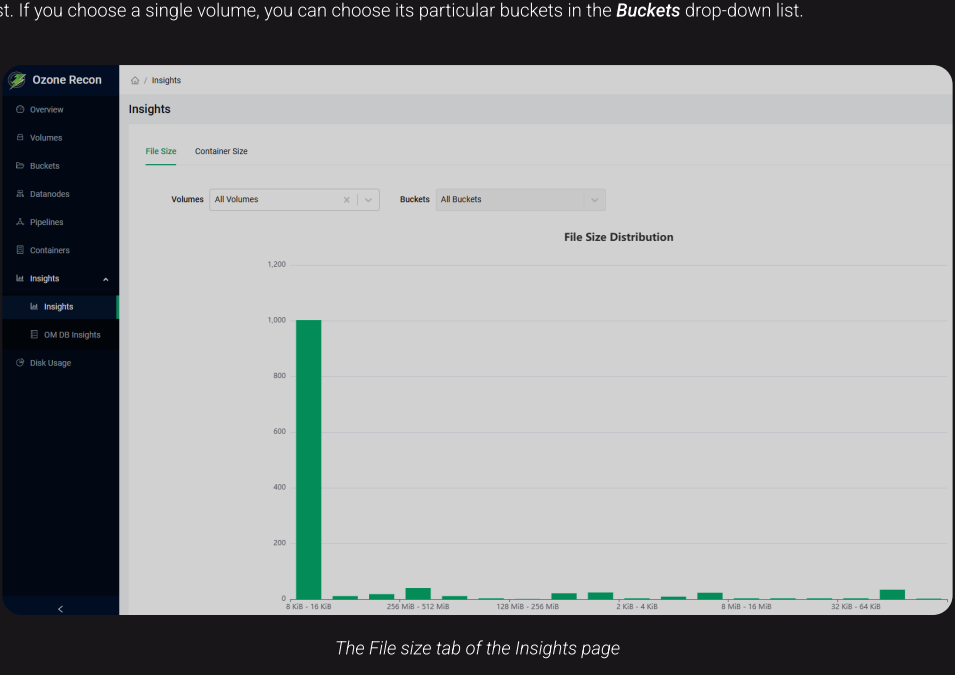
The Containers page

Insights

The ***Insights*** group contains the ***Insights*** page and the ***OM DB Insights*** page.

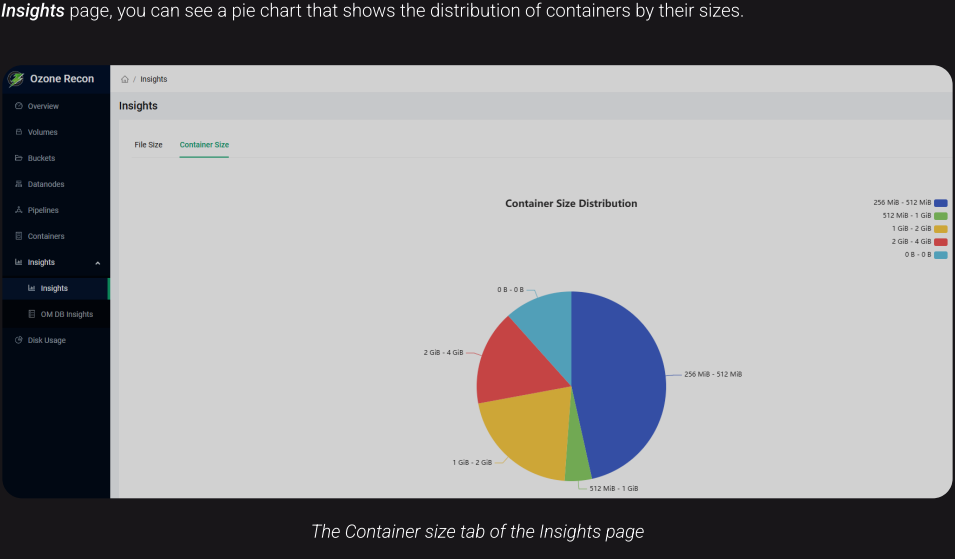
Insights

On the ***File Size*** tab of the ***Insights*** page, you can see a histogram that shows the distribution of files by their sizes. You can choose which volumes to include in the representation using the ***Volumes*** drop-down list. If you choose a single volume, you can choose its particular buckets in the ***Buckets*** drop-down list.



The File size tab of the Insights page

On the ***Container Size*** tab of the ***Insights*** page, you can see a pie chart that shows the distribution of containers by their sizes.



The Container size tab of the Insights page

OM DB Insights

The ***OM DB Insights*** page contains four tabs:

Container Mismatch Info

Containers are supposed to be present in both Ozone Manager and Storage Container Manager databases. This tab contains information on containers that are present only in one of them (mismatched containers). Information is presented as a table that has the following columns:

- ***Container ID*** — ID of the mismatched container.
- ***Count Of Keys*** — number of keys stored in the mismatched container.
- ***Pipelines*** — IDs of pipelines associated with the mismatched container.
- ***Exists*** — database where the mismatched container is present. Possible values: `OM` or `SCM`.

Open Keys

This tab contains information on the open keys in the Ozone Manager database. Information is presented as a table that has the following columns:

- ***Key*** — ID of the key.
- ***Amount of data*** — size occupied by the key contents.
- ***Path*** — path to the key location.
- ***In state since*** — time since the key has been opened.
- ***Replication Factor*** — value of the key replication factor.
- ***Replication Type*** — type of replication of the key. Possible values: `RATIS` or `STANDALONE`.
- ***Type*** — type of the key.

Keys Pending for Deletion

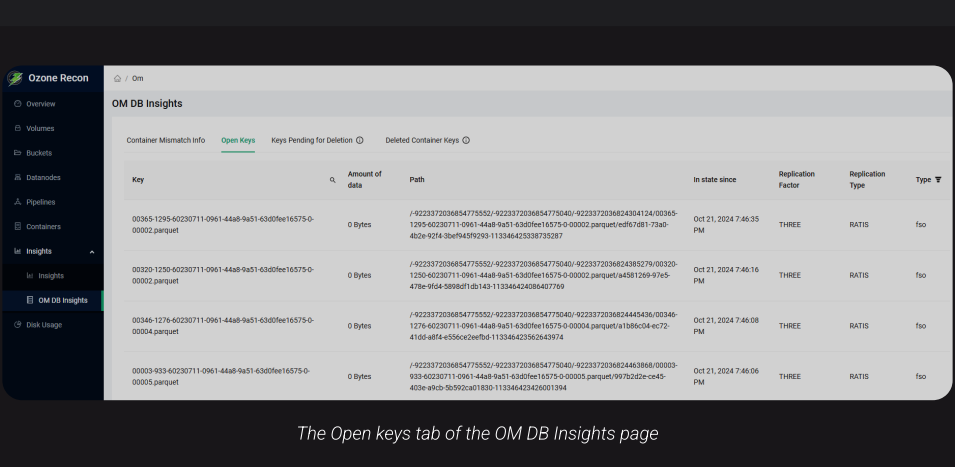
This tab contains information on the keys that are awaiting to be deleted. Information is presented as a table that has the following columns:

- ***Key Name*** — name of the key to be deleted.
- ***Path*** — path to the key location.
- ***Total Data Size*** — size occupied by the key data.
- ***Total Key Count*** — number of keys to be deleted.

Deleted Container Keys

This tab contains information on the keys mapped to containers that have the `DELETED` state in the Storage Container Manager database. Information is presented as a table that has the following columns:

- ***Container ID*** — ID of the deleted container.
- ***Count Of Keys*** — number of keys mapped to the container.
- ***Pipelines*** — list of pipelines associated with the container.



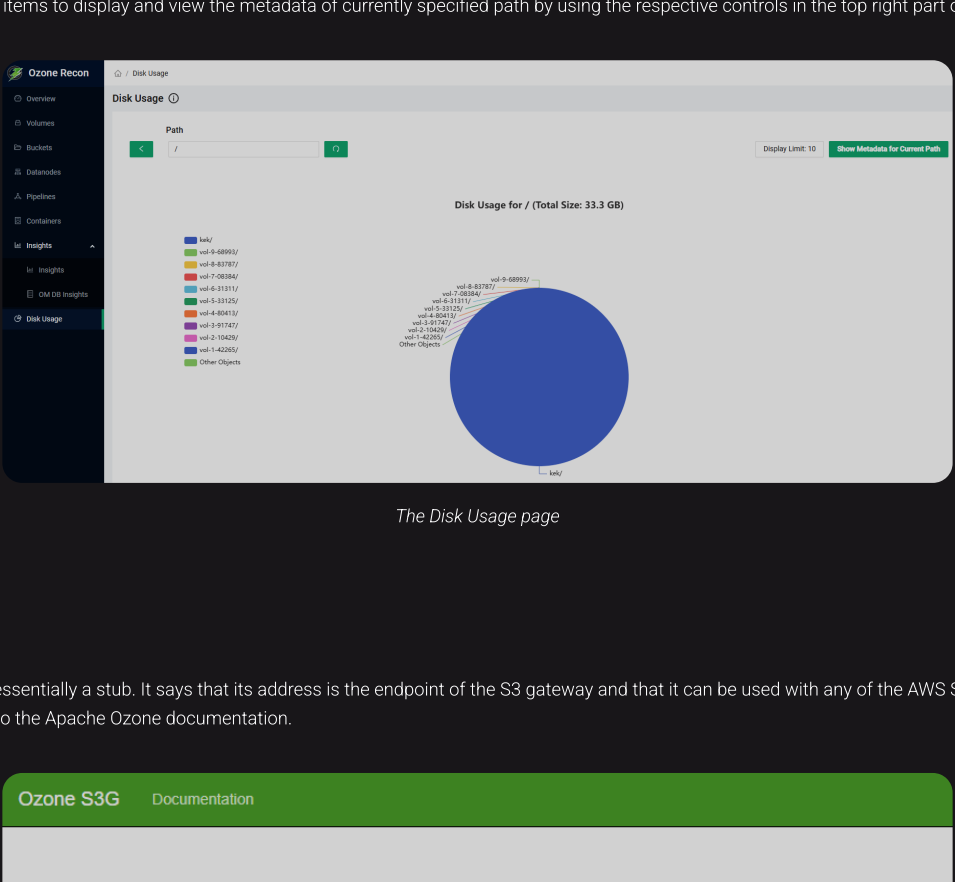
The Open keys tab of the OM DB Insights page

Disk Usage

The ***Disk Usage*** page contains the pie chart of disk usage by buckets stored in the Ozone cluster. You can filter the data in the following ways:

- You can enter a path in the corresponding field to view the statistics for the data contained in the specified directory and all its subdirectories.
- You can click on any item of the color legend in the left part of the page or on the pie chart sector (or its callout) to view its own statistics. Notice that in this case, the respective path appears in the ***Path*** field. This can be done several times until the lowest possible nesting level is reached.
- You can click on the **<** button to the left of the ***Path*** field to move one directory up.

Also you can choose the limit of items to display and view the metadata of currently specified path by using the respective controls in the top right part of the page.

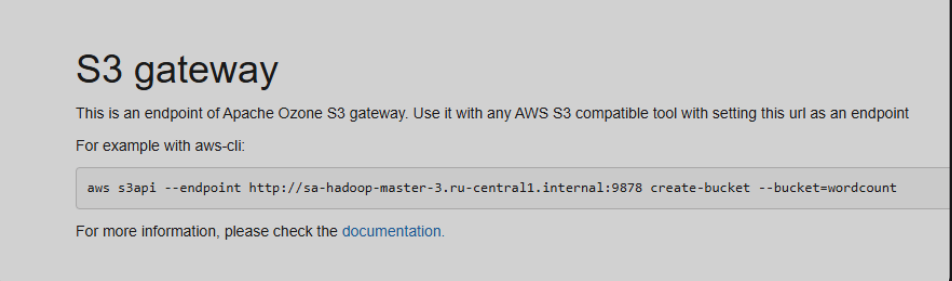


Ozone Recon
Disk Usage
Path
Display Limit: 10
Show Metadata for Current Path
Disk Usage for / (Total Size: 33.3 GB)

The Disk Usage page

Ozone S3G UI

The Ozone S3G UI only page is essentially a stub. It says that its address is the endpoint of the S3 gateway and that it can be used with any of the AWS S3 compatible tools to compose requests. It also contains a link to the Apache Ozone documentation.



S3 gateway

This is an endpoint of Apache Ozone S3 gateway. Use it with any AWS S3 compatible tool with setting this url as an endpoint

For example with aws-cli:

```
aws s3api --endpoint http://sa-hadoop-master-3.ru-central1.internal:9878 create-bucket --bucket=wordcount
```

For more information, please check the documentation.

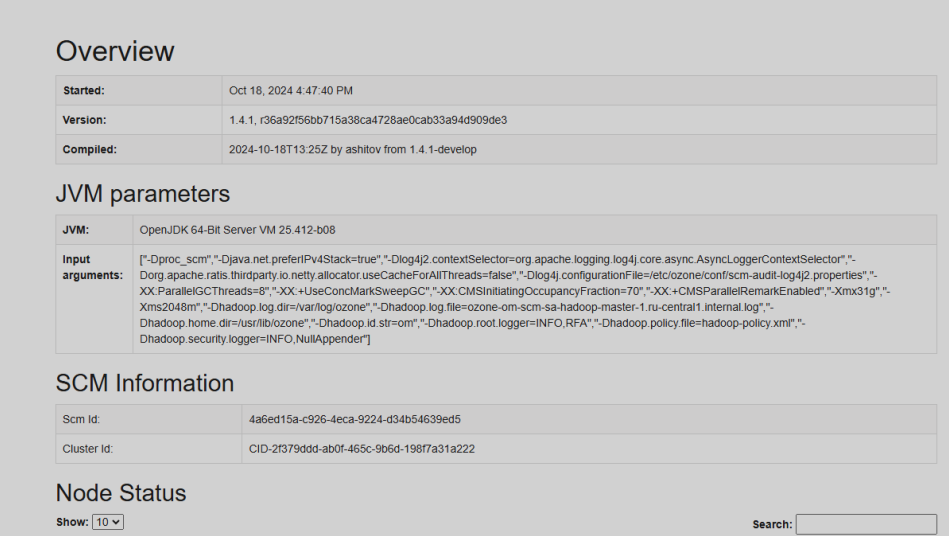
The Ozone S3G UI

Ozone Storage Container Manager UI

Overview

Start page of the Ozone Storage Container Manager (SCM) UI is the ***Overview*** page. It contains the following information:

- Service details:
  - time started;
  - version and checksum;
  - time compiled, user, and source branch.
- Java virtual machine details: version and input arguments string.
- SCM information: IDs of the SCM and cluster.
- Statuses of the nodes participating in the SCM cluster.
- Status of the high availability mode:
  - client and datanode RPC ports;
  - state of the node manager safe mode;
  - roles of the SCM nodes;
  - name of the primordial SCM node.
- Metadata volume information: home directories for Ratis and RocksDB.
- Safemode rules statuses: whether the rules intended to ensure safety passed the check.



Started:	Oct 18, 2024 4:47:20 PM
Version:	1.4.1, r36a92f56bb715a38ca4728ae0cab33a94d909de3
Compiled:	2024-10-18T13:25Z by ashitov from 1.4.1-develop

Scm Id:	4a6ed15a-c926-4eca-9224-d34b54639ed5
Cluster Id:	CID-2f379ddd-ab0f-465c-9b6d-198f7a31a222

HostName	Operational State	Commisioned State
sa-hadoop-master-1.ru-central1.internal	IN_SERVICE	HEALTHY
sa-hadoop-master-2.ru-central1.internal	IN_SERVICE	HEALTHY
sa-hadoop-master-3.ru-central1.internal	IN_SERVICE	HEALTHY

Page: 1 of 1. Showing 1 to 3 of the total 3 entries.

The Storage Container Manager UI Overview page

Metrics

The ***Metrics*** page is identical to the Metrics page of the Ozone Datanode UI.

Configuration

The ***Configuration*** page is identical to the Configuration page of the Ozone Datanode UI.

Documentation

The ***Documentation*** page is identical to the Documentation page of the Ozone Datanode UI.

Common tools

The ***Common tools*** group pages are identical to the corresponding pages of the Common tools group of the Ozone Datanode UI.

Ozone integration with cluster services

Konstantin Alpashkin

Contents

To Table of Contents

- Test cluster
- Configure ADH services to use Ozone
  - Step 1. Prepare Ozone file system
  - Step 2. Configure Spark3
  - Step 3. Configure Hive
  - Step 4. Configure YARN
- Use Ozone and HDFS
  - Hive
  - Spark3

This article describes a use case of integrating the Ozone service with an up-and-running ADH cluster. The scenario shows how to configure ADH services like Spark3, Hive, Impala, and YARN to use the Ozone storage rather than HDFS.

After switching to Ozone, the HDFS storage remains accessible and can be used by services along with Ozone.

Test cluster

The use case assumes the following environment:

- An ADH cluster 3.3.6.2 or higher is installed and is kerberized. The cluster includes the following services:
  - Core configuration
  - ZooKeeper
  - HDFS
  - YARN
  - ADPG
  - Hive
  - Impala
  - Spark3
- The Ozone service is added to the running ADH cluster. After Ozone is installed, run the ***Update Core configuration*** action of the Core configuration service and restart the following services:
  - YARN
  - Hive
  - Impala
  - Spark3

  This is required to pull in necessary Ozone dependencies to the classpath.
- The HDFS/Ozone services use the namespaces as described in the table.

Service/Component	Configuration property	Value
HDFS	dfs.internal.nameservices	adh
Ozone	ozone.service.id	adho
Ozone Manager	ozone.om.service.ids	adho
Ozone Storage Container Manager	ozone.scm.service.ids	adho

Configure ADH services to use Ozone

Step 1. Prepare Ozone file system

1. Using Ozone CLI, create a volume and a bucket where ADH services will store their data:

```
$ ozone sh volume create ozone
$ ozone sh bucket create /ozone/adh
```

Verify the new bucket:

```
$ hdfs dfs -ls ofs://adho/ozone
```

The output:

```
drwxrwxrwx   - k_alpashkin_krb1 k_alpashkin_krb1          0 2025-02-05 15:41 ofs://adho/ozone/adh
```

2. In the Core configuration service, set Ozone as the default file system using the following property.

Configuration section	Configuration property	Value
core-site.xml	fs.defaultFS	ofs://adho

3. Run the ***Update Core configuration*** action of the Core configuration service.

Step 2. Configure Spark3

1. Copy the necessary Spark3 archives from HDFS to Ozone and create log directories for Spark3 operation:

```
$ hdfs dfs -get hdfs://adh/apps/spark/spark3-yarn-archive.tgz
$ hdfs dfs -mkdir -p ofs://adho/ozone/adh/apps/spark
$ hdfs dfs -put spark3-yarn-archive.tgz ofs://adho/ozone/adh/apps/spark/spark3-yarn-archive.tgz
$ hdfs dfs -mkdir -p ofs://adho/ozone/adh/var/log/spark3/apps
```

2. Configure the Spark3 service to work with Ozone. Namely, update the properties listed in the table.

Configuration section	Configuration property	Value
spark-defaults.conf	spark.yarn.archive	ofs://adho/ozone/adh/apps/spark/spark3-yarn-archive.tgz
	spark.eventLog.dir	ofs://adho/ozone/adh/var/log/spark3/apps
	spark.history.fs.logDirectory	

3. Restart the Spark3 service.

Step 3. Configure Hive

1. Copy the necessary Hive files from HDFS to Ozone and create a new warehouse directory:

```
$ hdfs dfs -get hdfs://adh/apps/tez/tez-0.10.3.tar.gz
$ hdfs dfs -mkdir -p ofs://adho/ozone/adh/apps/tez
$ hdfs dfs -put tez-0.10.3.tar.gz ofs://adho/ozone/adh/apps/tez/tez-0.10.3.tar.gz
$ hdfs dfs -mkdir -p ofs://adho/ozone/adh/apps/hive/warehouse
```

2. Configure the Hive service to use the new warehouse location in Ozone.

Configuration section	Configuration property	Value
hive-site.xml	hive.metastore.warehouse.dir	ofs://adho/ozone/adh/apps/hive/warehouse

3. (Optional) Change the storage location of the `default` Hive database.

Change database files location

1. On a host with the Hive client installed, open the `beeline` shell:

```
$ beeline
```

2. In the `beeline` shell, connect to Hive using the JDBC string for the kerberized environment, for example:

```
!connect jdbc:hive2://ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.
internal:2181/;principal=hive/ka-adh-2.ru-central1.internal@AD.RANGER-
TEST;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=arenadata/cluster/17/hiveserver2
```

TIP

You can find the up-to-date JDBC string on the Hive service ***Info*** page in ADCM (***Clusters*** → ***<cluster_name>*** → ***Services*** → ***Hive*** → ***Info***).

3. Change the database location by running:

```
ALTER DATABASE default SET LOCATION 'ofs://adho/ozone/adh/apps/hive/warehouse';
```

4. Verify the new database location:

```
DESCRIBE DATABASE EXTENDED default;
```

The output:

```
+----------+------------------------+---------------------------------------------+-----------------------------------------
--+-------------+-------------+-----------------+----------------+-------------+
| db_name  |        comment         |                  location                   |              managedlocation
| owner_name  | owner_type  | connector_name  | remote_dbname  | parameters  |
+----------+------------------------+---------------------------------------------+-----------------------------------------
--+-------------+-------------+-----------------+----------------+-------------+
| default  | Default Hive database  | ofs://adho/ozone/adh/apps/hive/warehouse    | ofs://adho/ozone/adh/apps/hive/warehouse
| public      | ROLE        |                 |                |             |
+----------+------------------------+---------------------------------------------+-----------------------------------------
--+-------------+-------------+-----------------+----------------+-------------+
```

4. Restart the Hive service.

Step 4. Configure YARN

1. Create the necessary directories for the YARN service:

```
$ hdfs dfs -mkdir -p ofs://adho/ozone/adh/system/yarn/node-labels
```

2. Configure YARN to work with Ozone. Namely, specify the properties listed in the table.

Configuration section	Configuration property	Value
Custom yarn-site.xml	yarn.node-labels.fs-store.root-dir	ofs://adho/ozone/adh/system/yarn/node-labels
	yarn.nodemanager.remote-app-log-dir	ofs://adho/ozone/adh/logs

3. Restart YARN.

After completing the steps above, Hive, Spark, and Impala services will by default store data in Ozone rather than HDFS. Also, temporary and user files generated by YARN during application execution will also be stored in Ozone.

Use Ozone and HDFS

Hive

At the same time, the above services configured to work with Ozone, can still interact with HDFS. For example, all the Hive tables originally created in HDFS before the transition to Ozone, remain accessible. To create new Hive objects in HDFS, specify the location explicitly. For example:

```
CREATE DATABASE IF NOT EXISTS db_for_hdfs
COMMENT 'This is a database in HDFS.'
LOCATION 'hdfs://adh/apps/hive/warehouse/db_for_hdfs.db';
```

NOTE

All tables created in the `db_for_hdfs` database will be stored in HDFS by default.

Check the new database location:

```
DESCRIBE DATABASE EXTENDED db_for_hdfs;
```

The output:

```
+--------------+---------------------------+------------------------------------------------+-----------------------------------------
----------------+-------------+-------------+-----------------+----------------+-------------+
|   db_name    |          comment          |                    location                    |                  managedlocation
| owner_name  | owner_type  | connector_name  | remote_dbname  | parameters  |
+--------------+---------------------------+------------------------------------------------+-----------------------------------------
----------------+-------------+-------------+-----------------+----------------+-------------+
| db_for_hdfs  | This is database in hdfs  | hdfs://adh/apps/hive/warehouse/db_for_hdfs.db  |
ofs://adho/ozone/adh/apps/hive/warehouse/db_for_hdfs.db | hive   | USER        |                 |                |             |
+--------------+---------------------------+------------------------------------------------+-----------------------------------------
----------------+-------------+-------------+-----------------+----------------+-------------+
```

Spark3

Having configured your kerberized Spark cluster to work with Ozone, you can still run Spark jobs that interact with HDFS. However, for this, you have to explicitly instruct Spark to pass Kerberos authentication to access required HDFS directories. You can do this by using the `spark.kerberos.access.hadoopFileSystems` parameter, which accepts a comma-separated list of kerberized Hadoop filesystems your Spark application is going to access. For example:

```
/bin/spark3-submit \
    --deploy-mode cluster \
    --master yarn \
    --conf spark.kerberos.access.hadoopFileSystems=hdfs://adh
    test.py
```

NOTE

When using Ozone, Spark3 writes staging data to the */user/{username}* Ozone bucket. Keep in mind that in a kerberized environment, `{username}` is resolved to the name of the Kerberos principal. Thus, if the user name used to run a Spark application does not meet bucket naming requirements (e.g. contains `_` characters), the Spark job fails. To write Spark staging data to an Ozone bucket created in advance, use the `spark.yarn.stagingDir` parameter. This parameter can be specified in ADCM settings (***Custom spark-defaults.conf***) or by using the `--conf` notation, for example:

```
/bin/spark3-submit \
    --deploy-mode cluster \
    --master yarn \
    --conf spark.yarn.stagingDir=ofs://adho/ozone/adh/var \
    test.py
```

Ozone rack awareness

Vladimir Adamenkov

To Table of Contents

Rack awareness in Ozone is a feature that takes into account the physical network topology when placing data. It is crucial for data locality, fault tolerance, and overall performance, particularly in a geographically distributed cluster. If rack awareness is on, Ozone will place each key replica on a host in a different rack. This insures availability of data in case of a network failure or other unavailability issues.

To configure rack awareness for Ozone via ADCM, perform the following steps:

1. Go to the ADCM UI and select your cluster on the ***Clusters*** page.
2. Go to the ***Services*** tab and select ***Ozone***.
3. Switch on the ***Show advanced*** toggle and locate the ***Topology script*** and ***Topology data*** parameters.
4. Paste your network topology script as the value of the ***Topology script*** parameter.

Example topology script

```
#!/bin/bash

# Adjust/Add the property "net.topology.script.file.name"
# to core-site.xml with the "absolute" path the this
# file.  ENSURE the file is "executable".

# Supply appropriate rack prefix
RACK_PREFIX=default

# To test, supply a hostname as script input:
if [ $# -gt 0 ]; then

CTL_FILE=${CTL_FILE:-"topology.data"}

HADOOP_CONF=${HADOOP_CONF:-"/etc/hadoop/conf"}

if [ ! -f ${HADOOP_CONF}/${CTL_FILE} ]; then
  echo -n "/$RACK_PREFIX/rack "
  exit 0
fi

while [ $# -gt 0 ] ; do
  nodeArg=$1
  exec< ${HADOOP_CONF}/${CTL_FILE}
  result=""
  while read line ; do
    ar=( $line )
    if [ "${ar[0]}" = "$nodeArg" ] ; then
      result="${ar[1]}"
    fi
  done
  shift
  if [ -z "$result" ] ; then
    echo -n "/$RACK_PREFIX/rack "
  else
    echo -n "/$RACK_PREFIX/rack_$result "
  fi
done

else
  echo -n "/$RACK_PREFIX/rack "
fi
```

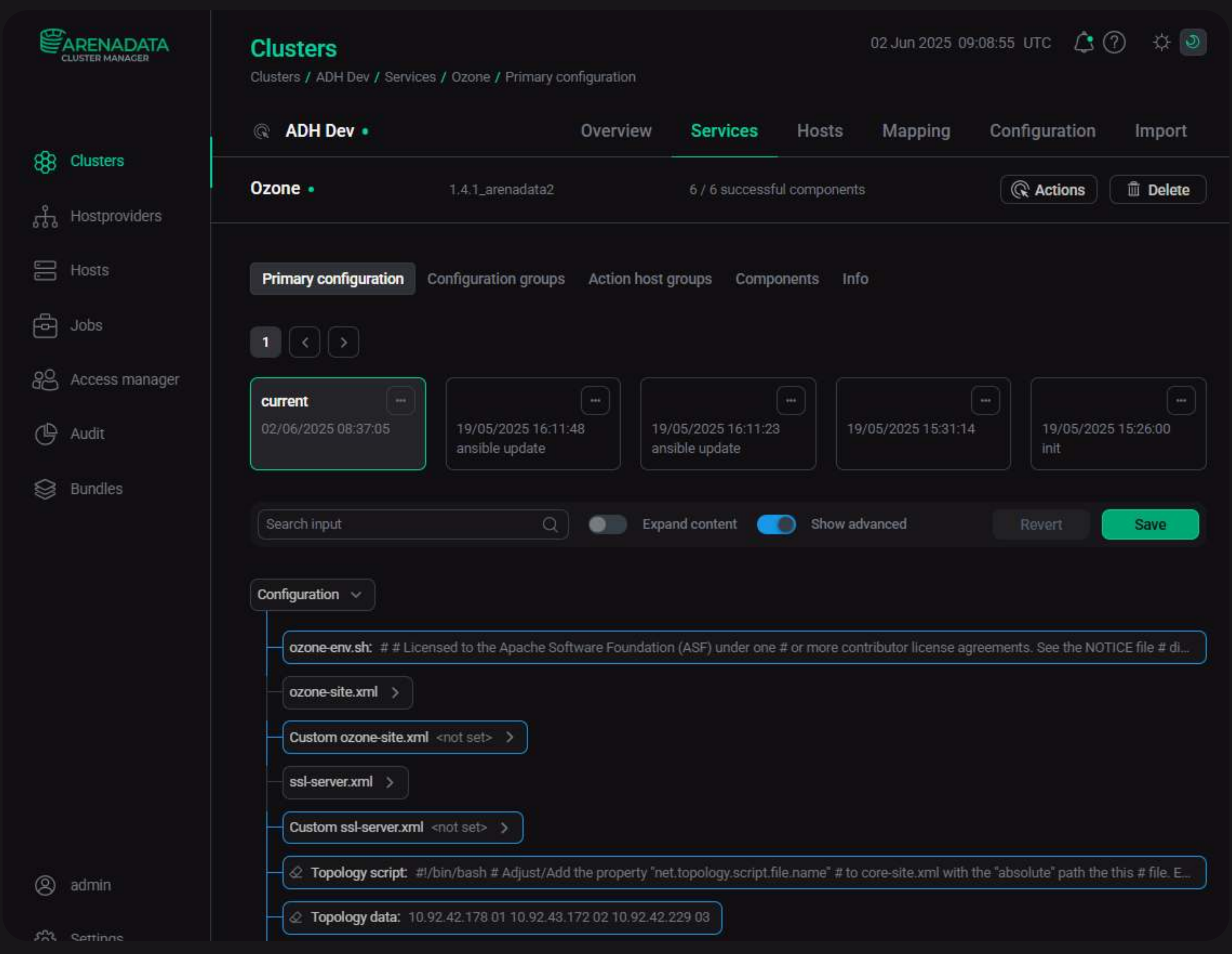
You can find additional script examples in the Rack Awareness article.

5. As value of the ***Topology data*** parameter, list the racks IPs and their corresponding IDs as shown in the following example:

Example topology data

```
# This file should be:
#  - Placed in the /etc/hadoop/conf directory
#    - On the Namenode (and backups IE: HA, Failover, etc)
#    - On the Job Tracker OR Resource Manager (and any Failover JT's/RM's)
# This file should be placed in the /etc/hadoop/conf directory.

# Add Hostnames to this file. Format <host ip> <rack_location>
10.92.42.178 01
10.92.43.172 02
10.92.42.229 03
```

6. Click ***Save***, then ***Create***.
7. In the ***Actions*** drop-down menu, select ***Restart***.
8. Make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Clusters
Clusters / ADH Dev / Services / Ozone / Primary configuration
02 Jun 2025 09:08:55 UTC
ADH Dev
Overview
Services
Hosts
Mapping
Configuration
Import
Ozone
1.4.1_arenadata2
6 / 6 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
1
current
02/06/2025 08:37:05
19/05/2025 16:11:48
ansible update
19/05/2025 16:11:23
ansible update
19/05/2025 15:31:14
19/05/2025 15:26:00
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
ozone-site.xml
Custom ozone-site.xml <not set>
ssl-server.xml
Custom ssl-server.xml <not set>

ozone-env.sh: # # Licensed to the Apache Software Foundation (ASF) under one # or more contributor license agreements. See the NOTICE file # di...

Topology script: #!/bin/bash # Adjust/Add the property "net.topology.script.file.name" # to core-site.xml with the "absolute" path the this # file. E...

Topology data: 10.92.42.178 01 10.92.43.172 02 10.92.42.229 03

Topology script and data fields in ADCM

To check if rack awareness has been successfully configured, you can use the following command:

```
$ ozone admin datanode list
```

The output should look like this:

```
Datanode 10.92.42.178:9866 - Rack: /rack01 - Status: UP
Datanode 10.92.43.172:9866 - Rack: /rack02 - Status: UP
Datanode 10.92.42.229:9866 - Rack: /rack03 - Status: UP
```

Ozone service management via ADCM

Vladimir Adamenkov

Contents

To Table of Contents

Overview

Overview

The ADCM UI provides actions to manage the Ozone service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Ozone service are listed in the table below.

Ozone service actions

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Ozone components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM
Manage Ranger plugin	Enables or disables Ranger plugin for Ozone

The Ozone has five components:

- Ozone Datanode
- Ozone Manager
- Ozone Recon
- Ozone S3G
- Ozone Storage Container Manager

All of these components have the same set of available actions listed in the table below.

Ozone components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Connect to Phoenix

Daria Barysheva

Contents

To Table of Contents

SQLLine
JDBC
GUI
  - Step 1. Add a driver
  - Step 2. Add a connection
  - Step 3. Work with Phoenix tables

There are several ways to work with Phoenix. You can use the most suitable work mode for you.

SQLLine

The simplest way to begin working with Phoenix is to use SQLLine. It is a Java console, available on each node of the HBase cluster after its installation. To start working with SQLLine, run the following commands:

```
$ cd /usr/lib/phoenix/bin
$ ./sqlline.py
```

The SQLLine prompt ends with a `>` character. All subsequent commands should be written after it.

The SQLLine prompt

```
Setting property: [incremental, false]
Setting property: [isolation, TRANSACTION_READ_COMMITTED]
issuing: !connect jdbc:phoenix: none none org.apache.phoenix.jdbc.PhoenixDriver
Connecting to jdbc:phoenix:
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
21/11/30 13:51:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
Connected to: Phoenix (version 5.0)
Driver: PhoenixEmbeddedDriver (version 5.0)
Autocommit status: true
Transaction isolation: TRANSACTION_READ_COMMITTED
Building list of tables and columns for tab-completion (set fastconnect to true to skip)...
142/142 (100%) Done
Done
sqlline version 1.2.0
0: jdbc:phoenix:>
```

To logout from SQLLine, use the following command:

```
!quit
```

The basic data operations available in Phoenix via SQLLine are described in Use SQLLine to work with Phoenix.

JDBC

There are two ways to connect to Phoenix via JDBC:

- **Phoenix Thick Driver**. It is used to connect via ZooKeeper. The full JDBC URL looks like this:

```
jdbc:phoenix:[comma-separated ZooKeeper Quorum Hosts [: ZK port [:hbase root znode [:kerberos_principal [:path to kerberos
keytab]]]]
```

The simplest example:

```
jdbc:phoenix:bds-adh-1.ru-central1.internal,bds-adh-2.ru-central1.internal,bds-adh-3.ru-central1.internal:2181
```

- **Phoenix Thin Driver**. It is designed to implement a standalone client avoiding excessive number of dependencies. It uses special Phoenix Query Server services. The JDBC URL looks like this:

```
jdbc:phoenix:thin:[key=value[;key=value...]]
```

The example:

```
jdbc:phoenix:thin:url=http://10.92.6.9:8765;serialization=PROTOBUF
```

NOTE

You can find more information about Phoenix JDBC drivers in Phoenix documentation.

In practice, both drivers are used. The choice of a specific driver is determined by the cluster topology, the number of Phoenix Query Servers, and the ability to place dependencies into the client application. When using batches, the performance is approximately the same for both drivers, without batches — the Thick Driver usually shows a performance of about 30% higher.

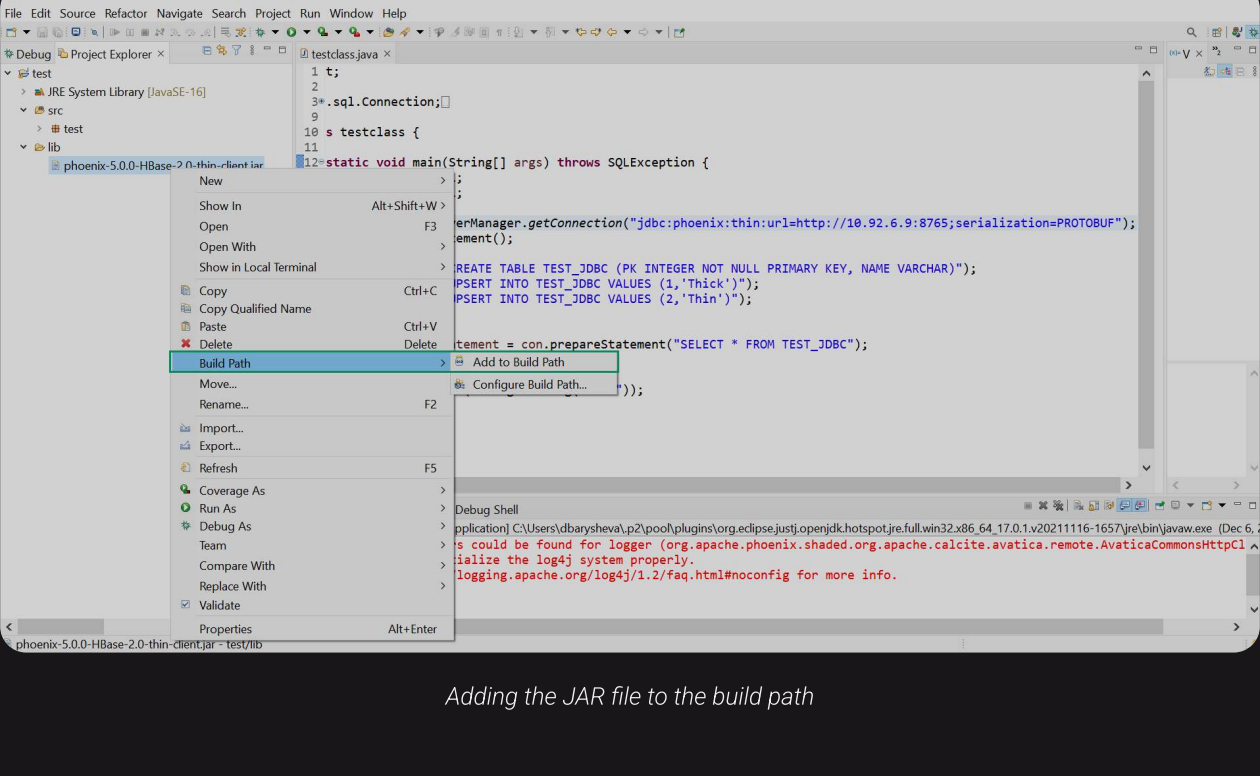
Let's consider a simple Java application, using the Phoenix Thin Driver:

1. Create a Java class with the following code (using Eclipse IDE, for example). Replace `<IP>` with the IP-address of your Phoenix Query Server. This code creates a new table `TEST_JDBC` with two columns, inserts some values into it, and then selects these values from the table.

```
package test;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.Statement;

public class testclass {

        public static void main(String[] args) throws SQLException {
                Statement stmt = null;
                ResultSet rset = null;

                Connection con = DriverManager.getConnection("jdbc:phoenix:thin:url=http://<IP>:8765;serialization=PROTOBUF");
                stmt = con.createStatement();

                stmt.executeUpdate("CREATE TABLE TEST_JDBC (PK INTEGER NOT NULL PRIMARY KEY, NAME VARCHAR)");
                stmt.executeUpdate("UPSERT INTO TEST_JDBC VALUES (1,'Thick')");
                stmt.executeUpdate("UPSERT INTO TEST_JDBC VALUES (2,'Thin')");
                con.commit();

                PreparedStatement statement = con.prepareStatement("SELECT * FROM TEST_JDBC");
                rset = statement.executeQuery();
                while (rset.next()) {
                        System.out.println(rset.getString("NAME"));
                }
                statement.close();
                con.close();
        }
}
```

2. Get the corresponding version of the JAR file *phoenix-<phoenix_version>-HBase-<hbase_version>-thin-client.jar*. For example, for HBase 2.0 and Phoenix 5.0.0, this file is called *phoenix-5.0.0-HBase-2.0-thin-client.jar*. You can copy it from any host of your HBase cluster (from the */usr/lib/phoenix* directory). The other way is to download it from Phoenix repository.

3. Add the uploaded file to the build path of your Java application. If you use Eclipse, you can add the file to the */lib* directory of your workspace and then use the context menu item ***Build Path → Add to Build Path***.



eclipse-workspace - test/src/test/testclass.java - Eclipse IDE
Build Path
Add to Build Path
Configure Build Path...

Adding the JAR file to the build path

4. Run your class as Java application. The result should be similar to:

```
log4j:WARN No appenders could be found for logger (org.apache.phoenix.shaded.org.apache.calcite.avatica.remote.
AvaticaCommonsHttpClientImpl).
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
Thick
Thin
```

CAUTION

If there is no Kerberos authentication in your ADH cluster, the JDBC connection will use the login name `phoenix` by default. This user should have the sufficient rights in HBase. For example, if you want only to create tables and write data to them, run the following command in HBase shell:

```
grant 'phoenix', 'WC'
```

Otherwise, you will get the following errors in your Java application:

```
Insufficient permissions (user=phoenix, scope=default, params=
[namespace=default,table=default:TEST_JDBC,family=0],action=CREATE)
Insufficient permissions (user=phoenix, scope=default:SYSTEM.CATALOG, family=0:USE_STATS_FOR_PARALLELIZATION,
params=[table=default:SYSTEM.CATALOG,family=0:USE_STATS_FOR_PARALLELIZATION],action=WRITE)
```

For more information about the previously used command, see the grant.

5. Check the existence of the `TEST_JDBC` table in Phoenix:

```
!tables
```

The output:

```
+------------+--------------+--------------+---------------+----------+--------+
| TABLE_CAT  | TABLE_SCHEM  |  TABLE_NAME  |  TABLE_TYPE   | REMARKS  | TYPE_N |
+------------+--------------+--------------+---------------+----------+--------+
|            | SYSTEM       | CATALOG      | SYSTEM TABLE  |          |        |
|            | SYSTEM       | FUNCTION     | SYSTEM TABLE  |          |        |
|            | SYSTEM       | LOG          | SYSTEM TABLE  |          |        |
|            | SYSTEM       | SEQUENCE     | SYSTEM TABLE  |          |        |
|            | SYSTEM       | STATS        | SYSTEM TABLE  |          |        |
|            |              | BEST_BOOKS   | TABLE         |          |        |
|            |              | TEST_JDBC    | TABLE         |          |        |
|            |              | employee     | TABLE         |          |        |
|            |              | people_ages  | TABLE         |          |        |
+------------+--------------+--------------+---------------+----------+--------+
0: jdbc:phoenix:> select * from TEST_JDBC;
+-----+--------+
| PK  |  NAME  |
+-----+--------+
| 1   | Thick  |
| 2   | Thin   |
+-----+--------+
2 rows selected (0.058 seconds)
```

GUI

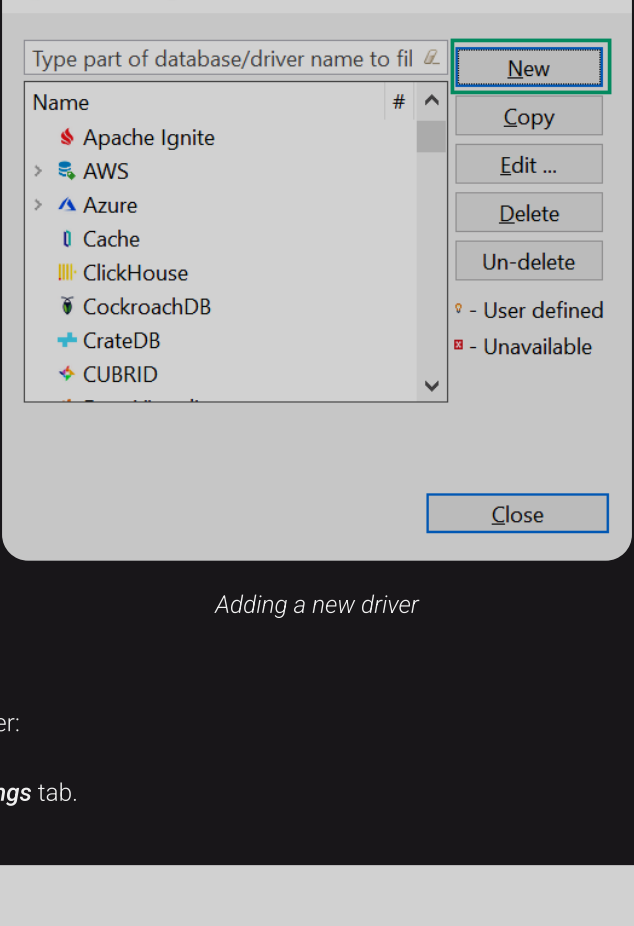
You can also use JDBC drivers for connecting to Phoenix via GUI. Let's consider the example of using a thin JDBC driver in DBeaver.

NOTE

We give this subsection only as an example. If GUI access is required, we strongly recommend to take a look at Zeppelin, which is included into the ADH package and specifically designed for Big Data analytics. You can find information on connecting from Zeppelin to Phoenix in Add a custom interpreter to a group.

Step 1. Add a driver

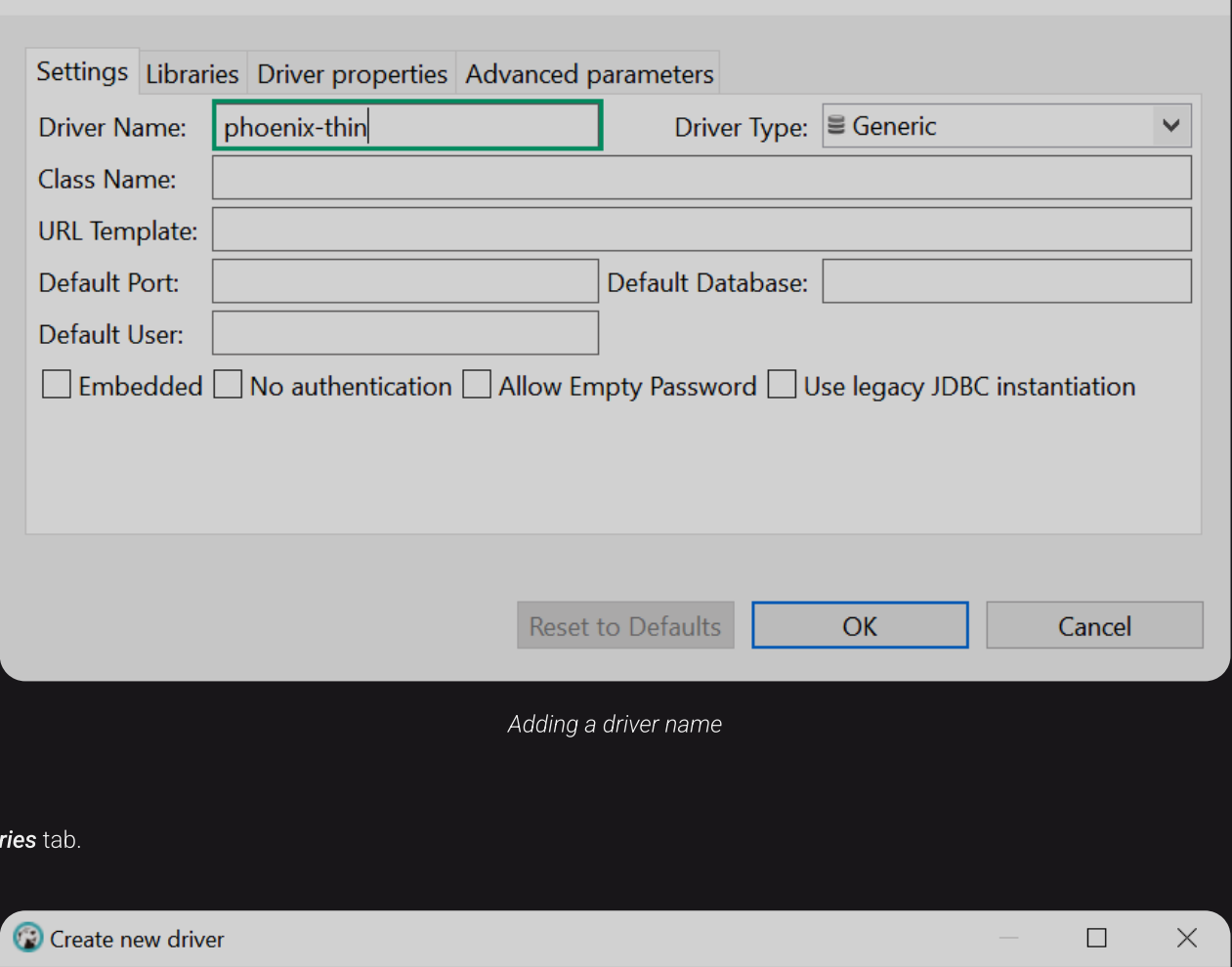
1. Install DBeaver on your local computer.
2. Get the corresponding version of the JAR file *phoenix-<phoenix_version>-HBase-<hbase_version>-thin-client.jar*. For example, for HBase 2.0 and Phoenix 5.0.0, this file is called *phoenix-5.0.0-HBase-2.0-thin-client.jar*. You can copy it from any host of your HBase cluster (from the */usr/lib/phoenix* directory). The other way is to download it from Phoenix repository.
3. Put the uploaded file into the *plugins* folder inside of the DBeaver work directory. For example, in Windows, the full path should look like this: *C:/Program Files/DBeaver/plugins/*.
4. Select ***Database → Driver Manager*** in the main menu of DBeaver.
5. In the opened form, click ***New***.



Driver Manager
New
Copy
Edit ...
Delete
Un-delete
Apache Ignite
AWS
Azure
Cache
ClickHouse
CockroachDB
CrateDB
CUBRID
Close

Adding a new driver

6. In the opened window, add information about a new driver:
   - Enter the driver name `phoenix-thin` at the ***Settings*** tab.

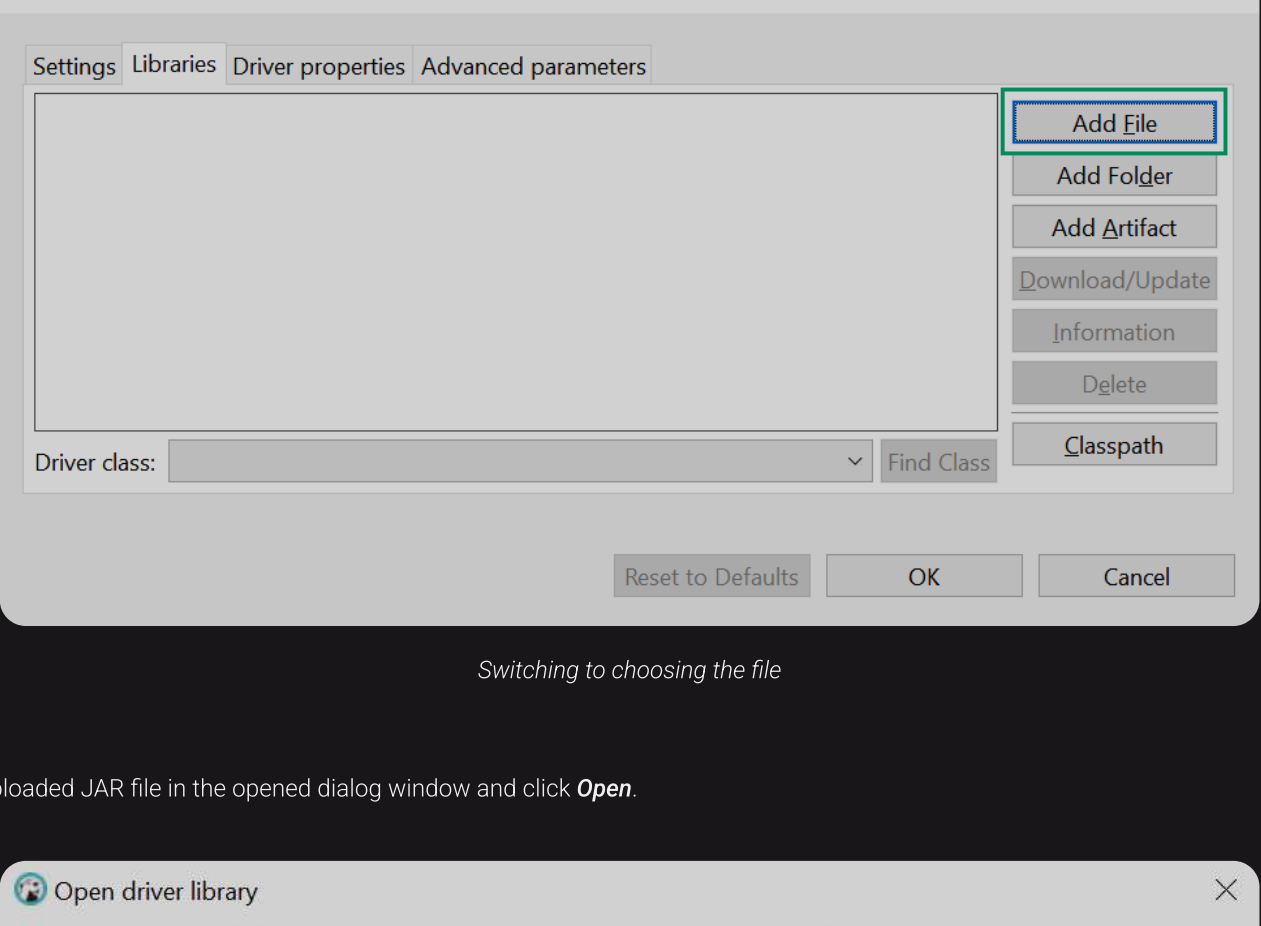


Create new driver
Settings
Libraries
Driver properties
Advanced parameters
Reset to Defaults
OK
Cancel

Driver Name: phoenix-thin
Driver Type: Generic

Adding a driver name

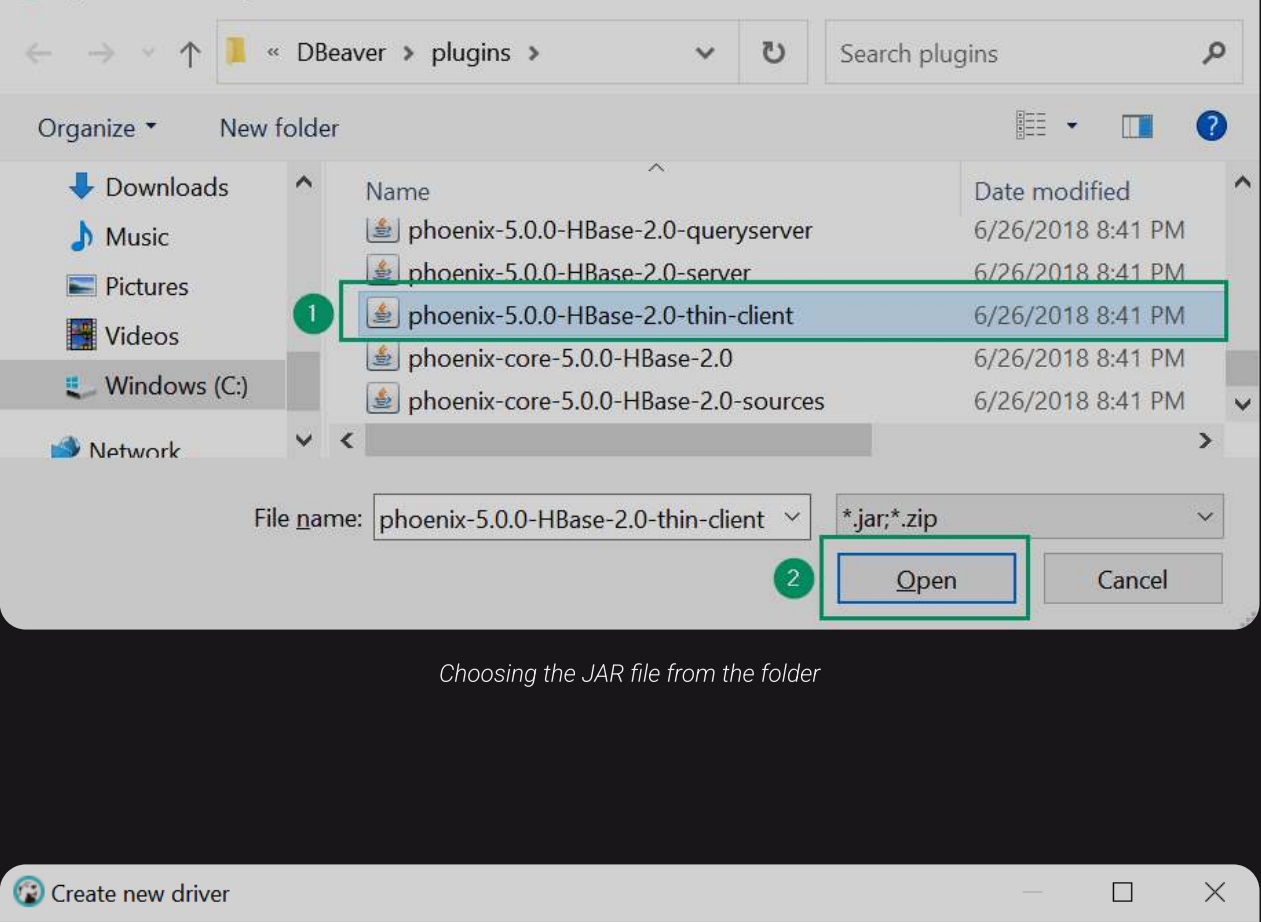
   - Click ***Add file*** at the ***Libraries*** tab.



Create new driver
Add File
Add Folder
Add Artifact
Download/Update
Information
Delete
Classpath
Find Class

Switching to choosing the file

   - Choose the previously uploaded JAR file in the opened dialog window and click ***Open***.

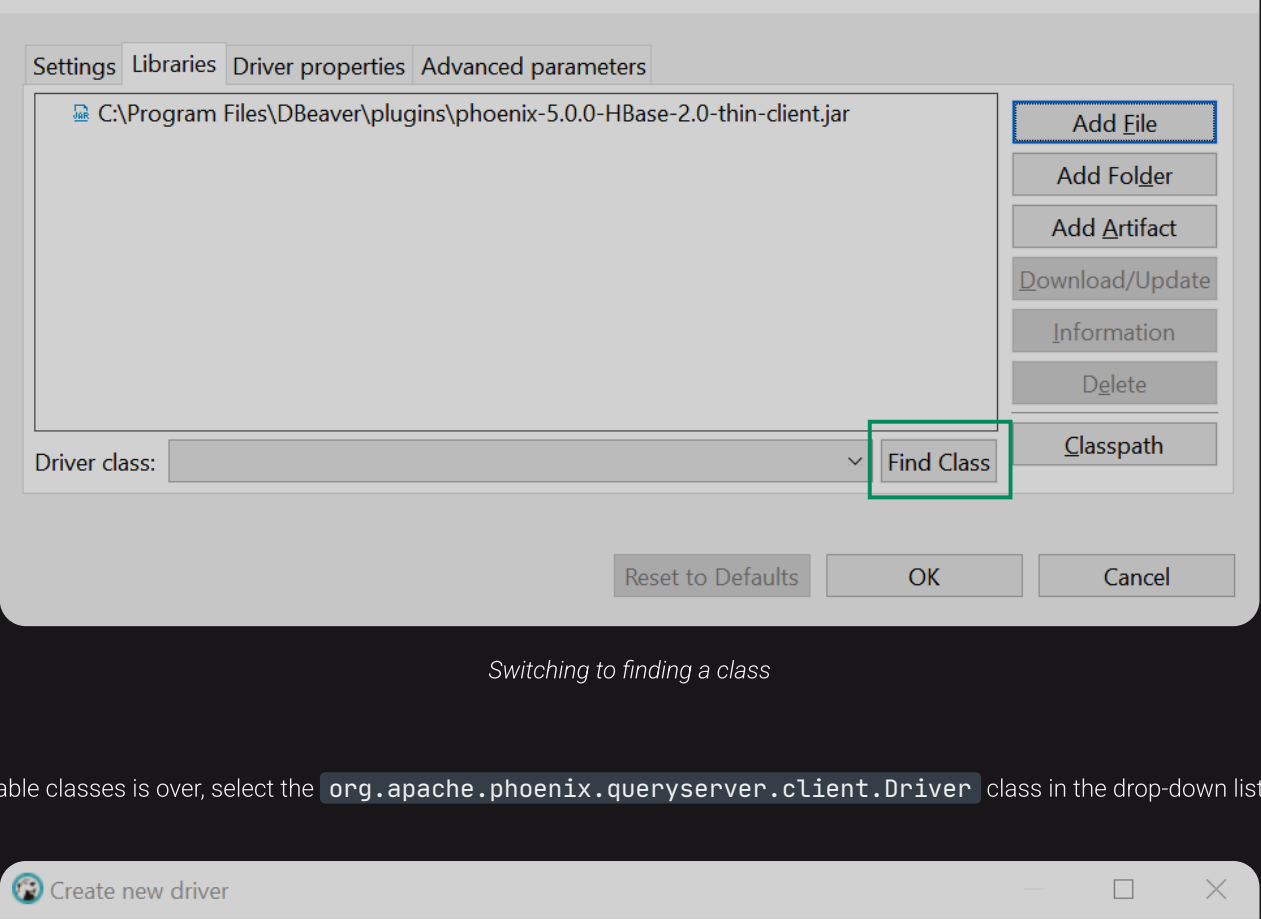


Open driver library
DBeaver › plugins ›
phoenix-5.0.0-HBase-2.0-queryserver
phoenix-5.0.0-HBase-2.0-server
phoenix-5.0.0-HBase-2.0-thin-client
phoenix-core-5.0.0-HBase-2.0
phoenix-core-5.0.0-HBase-2.0-sources
1
2
Open
Cancel

File name: phoenix-5.0.0-HBase-2.0-thin-client

Choosing the JAR file from the folder

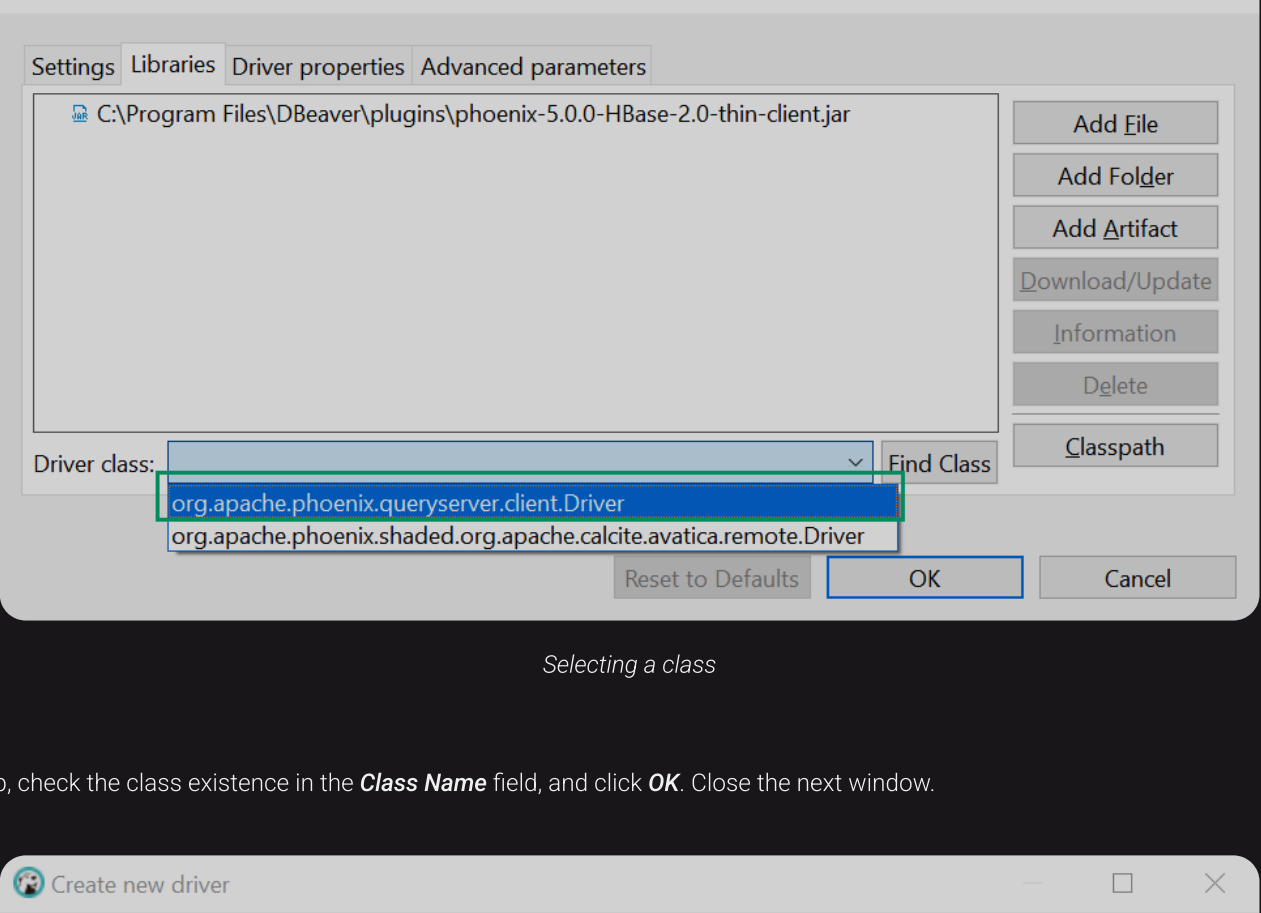
   - Click ***Find Class***.



C:\Program Files\DBeaver\plugins\phoenix-5.0.0-HBase-2.0-thin-client.jar

Switching to finding a class

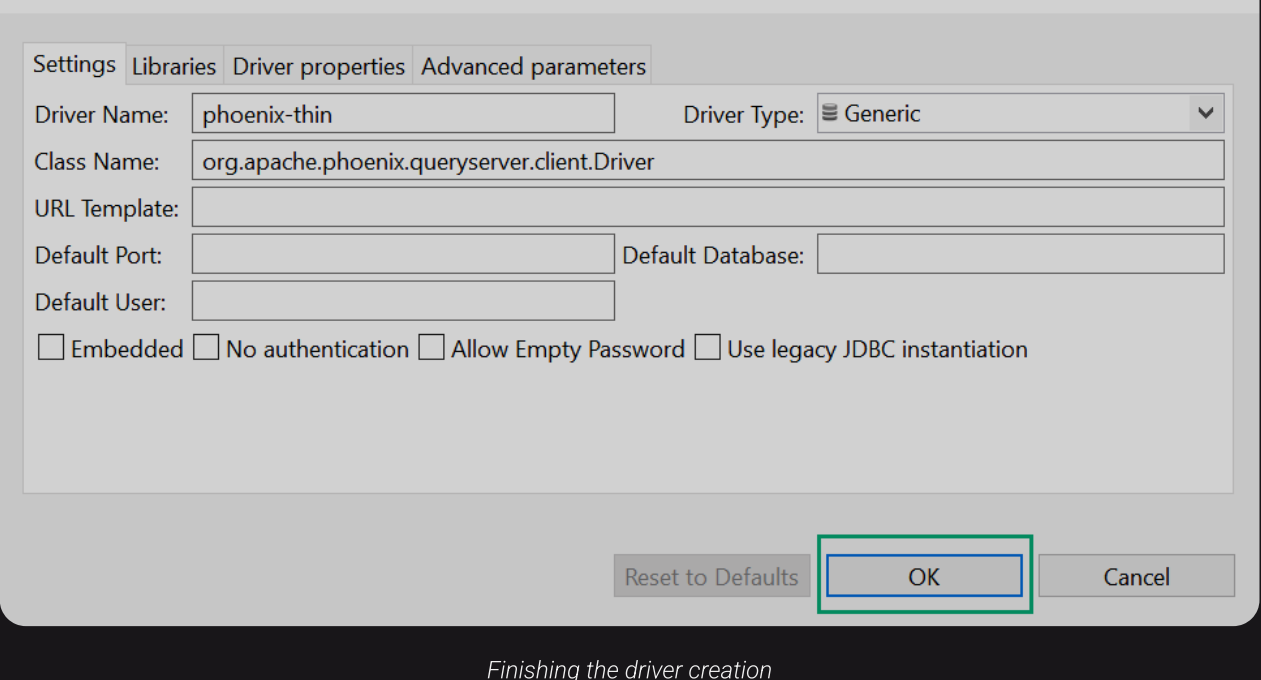
   - When the search for suitable classes is over, select the `org.apache.phoenix.queryserver.client.Driver` class in the drop-down list.



Driver class:
org.apache.phoenix.queryserver.client.Driver
org.apache.phoenix.shaded.org.apache.calcite.avatica.remote.Driver

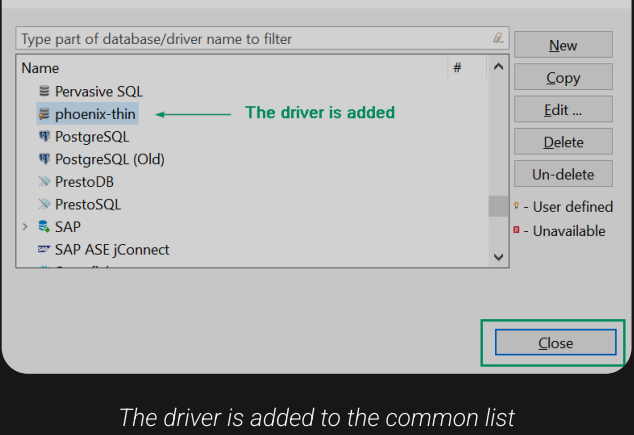
Selecting a class

   - Return to the ***Settings*** tab, check the class existence in the ***Class Name*** field, and click ***OK***. Close the next window.



Driver Name: phoenix-thin
Driver Type: Generic
Class Name: org.apache.phoenix.queryserver.client.Driver

Finishing the driver creation



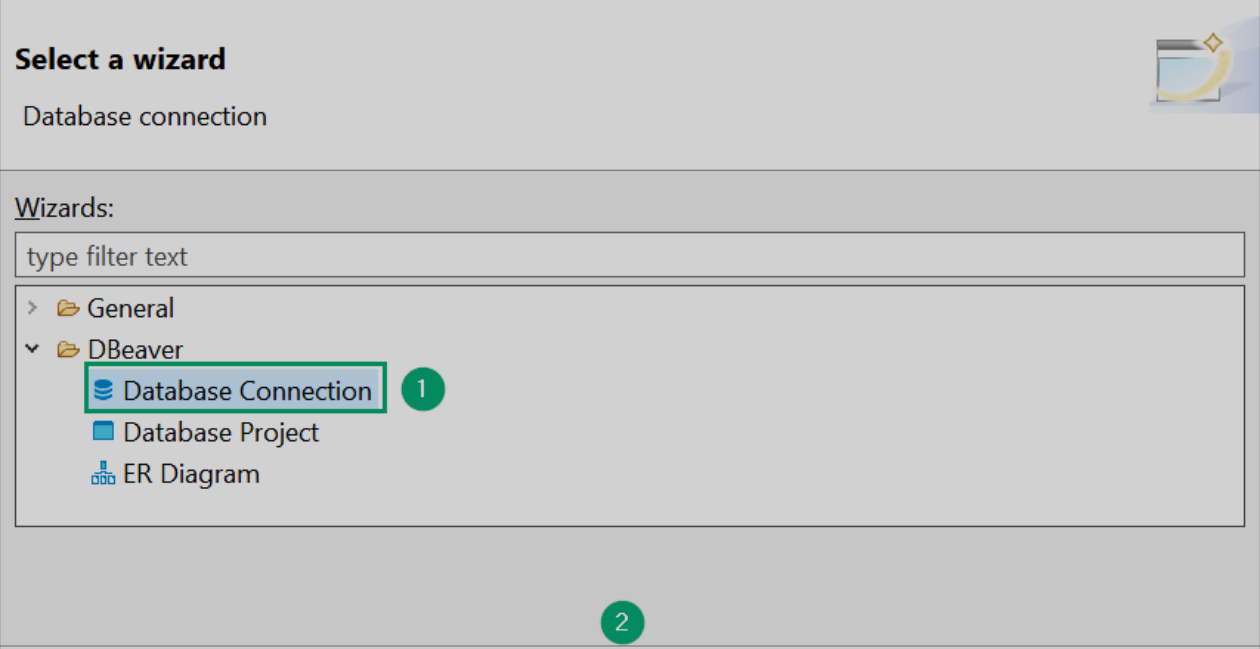
Driver Manager
phoenix-thin
The driver is added
Close

The driver is added to the common list

The new driver is ready.

Step 2. Add a connection

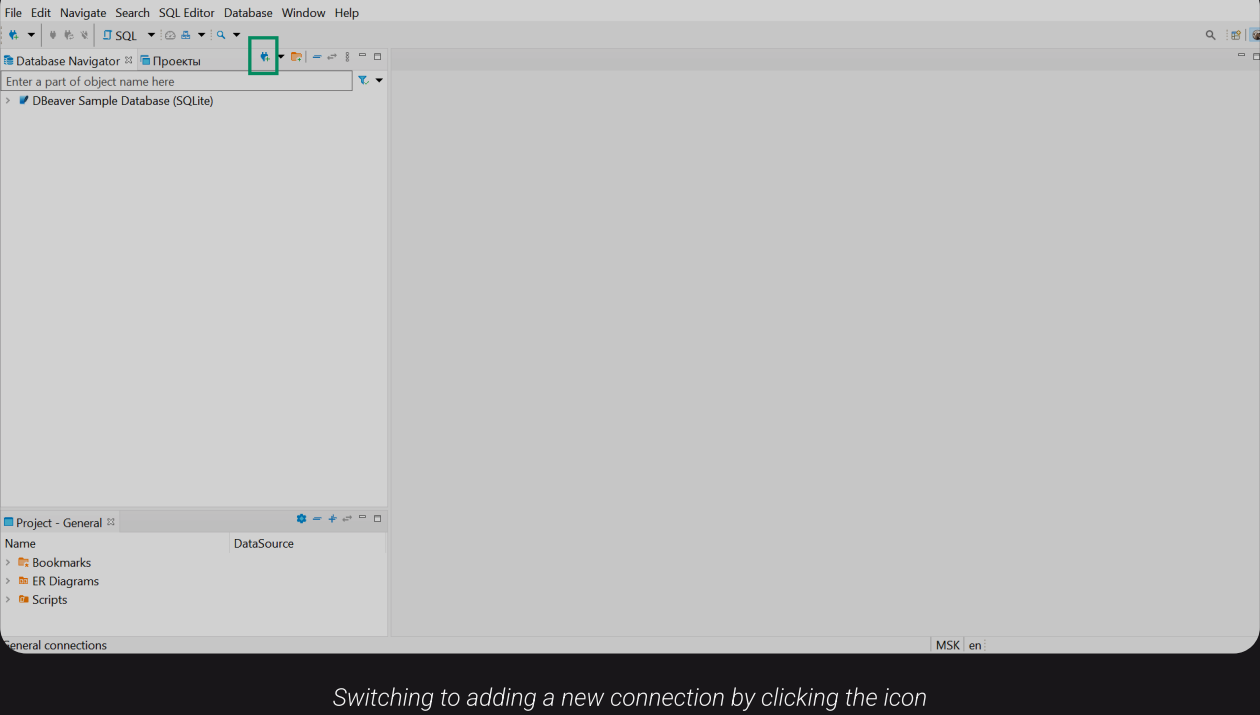
1. Open the menu item ***File → New***. In the opened window, select the dropdown menu item ***DBeaver → Database Connection*** and click ***Next***.



Select a wizard
Database connection
General
DBeaver
Database Connection
Database Project
ER Diagram
1
2
Next >
Cancel

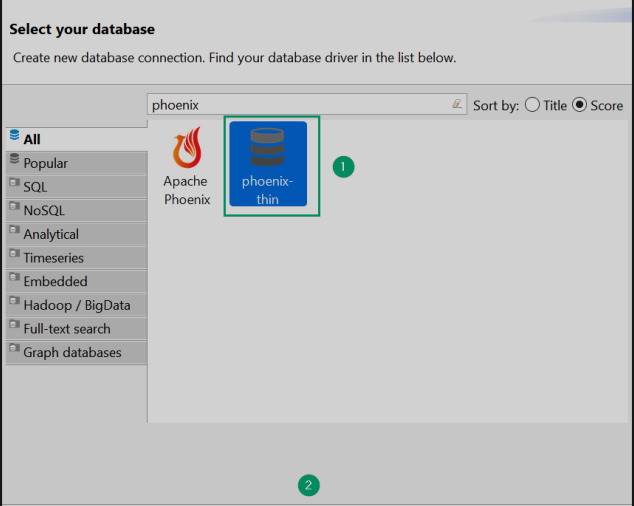
Switching to adding a new connection from the main menu

Another way is to click the corresponding icon in the main window.



Switching to adding a new connection by clicking the icon

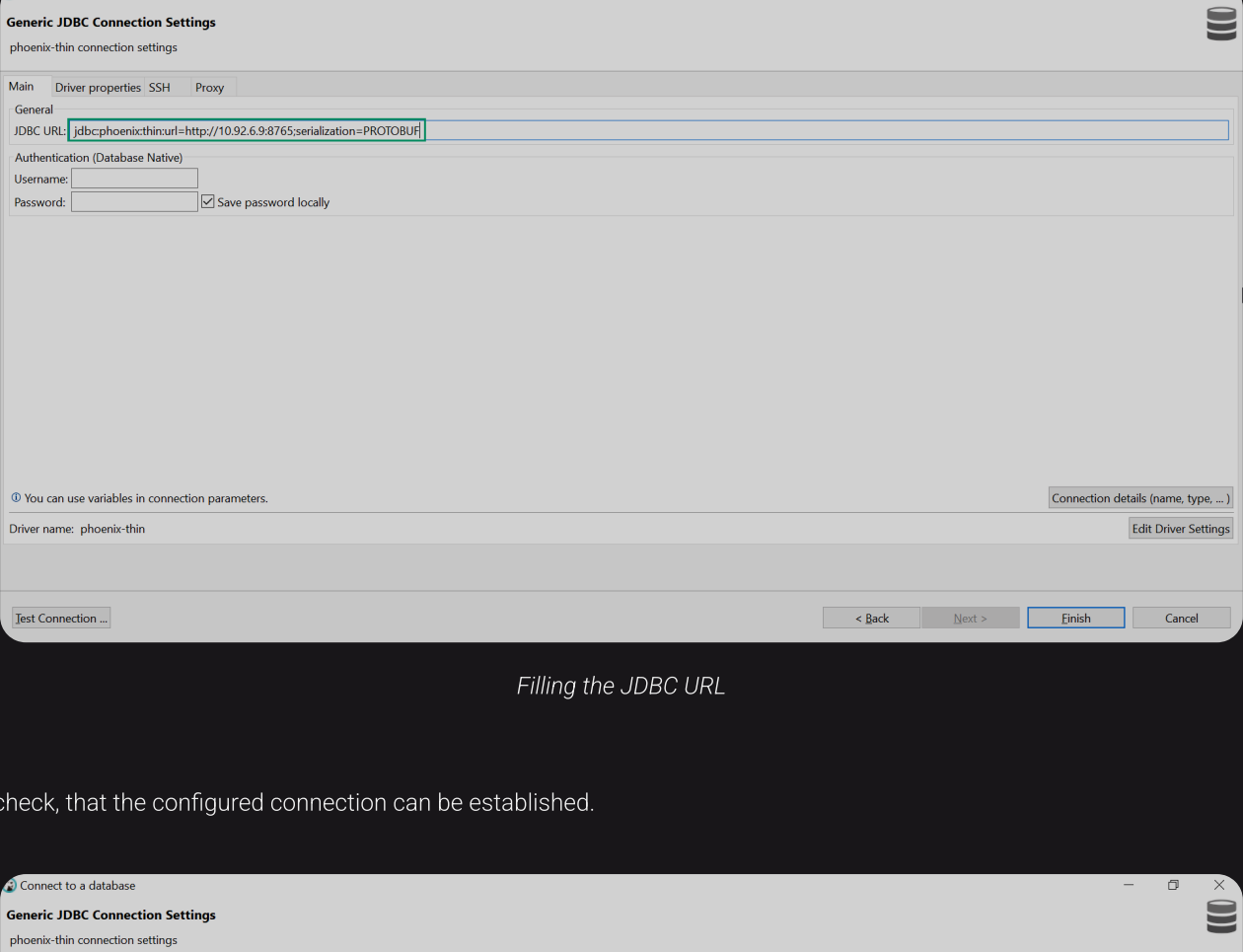
2. Find the previously created driver `phoenix-thin` and click ***Next***.



Connect to a database
Select your database
Apache Phoenix
phoenix-thin
1
2
Next >

Selecting the driver

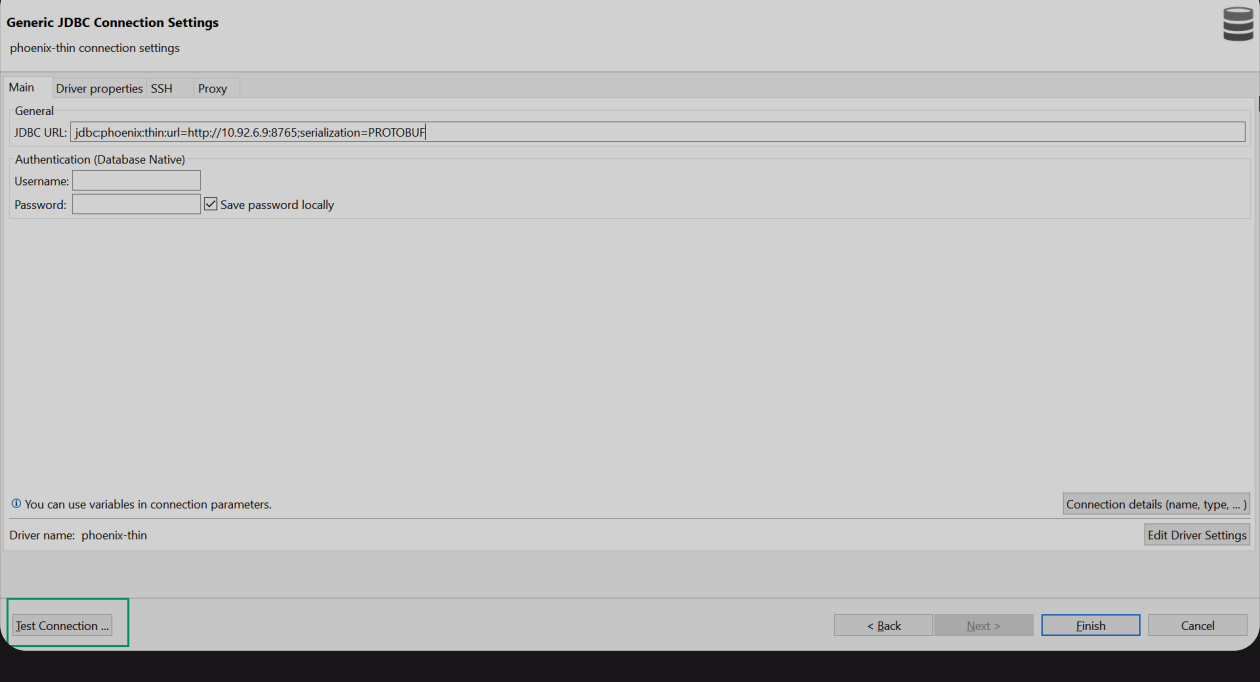
3. Enter the JDBC URL in the field of the same name. Use the following syntax: `jdbc:phoenix:thin:url=http://<IP>:8765;serialization=PROTOBUF`, where `<IP>` means the IP-address of your Phoenix Query Server.



JDBC URL: jdbc:phoenix:thin:url=http://10.92.6.9:8765;serialization=PROTOBUF
Driver name: phoenix-thin

Filling the JDBC URL

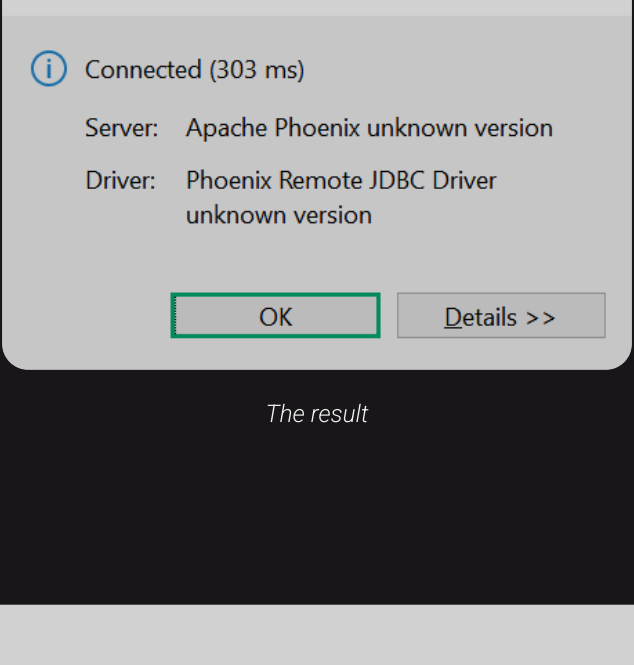
4. Click ***Test Connection...*** to check, that the configured connection can be established.



Test Connection ...

Testing the connection

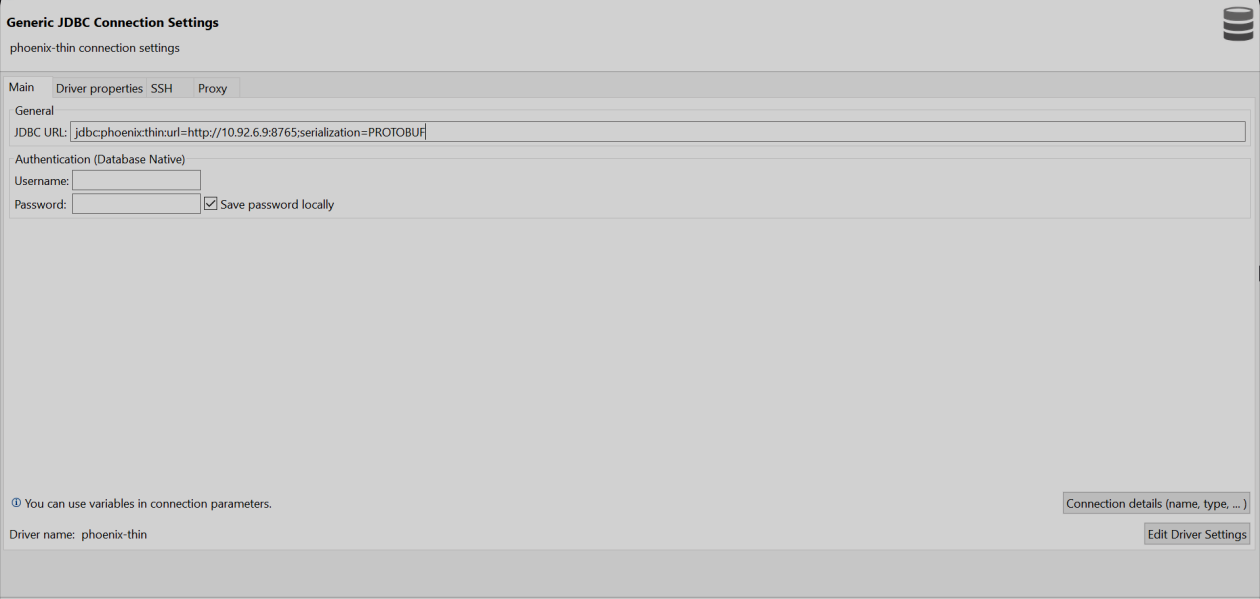
The result is similar to:



Connection test
Connected (303 ms)
Server: Apache Phoenix unknown version
Driver: Phoenix Remote JDBC Driver unknown version

The result

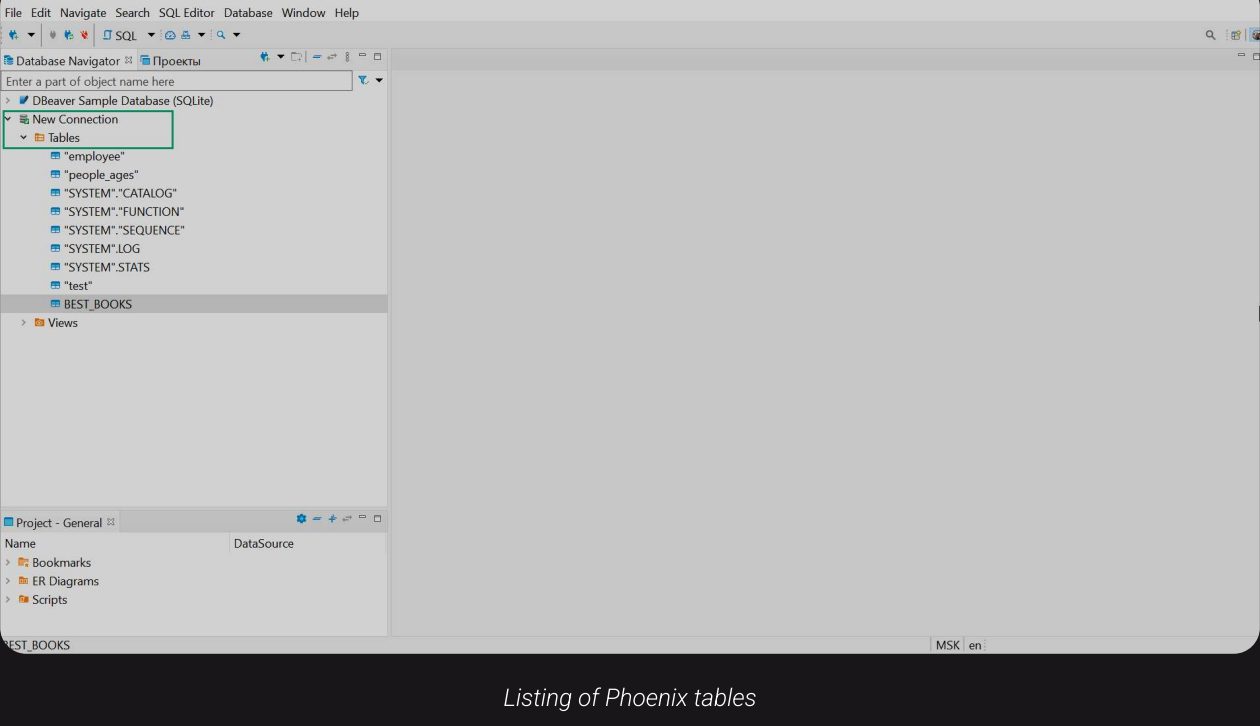
5. Click ***Finish***. The connection is added.



Finishing the connection adding

Step 3. Work with Phoenix tables

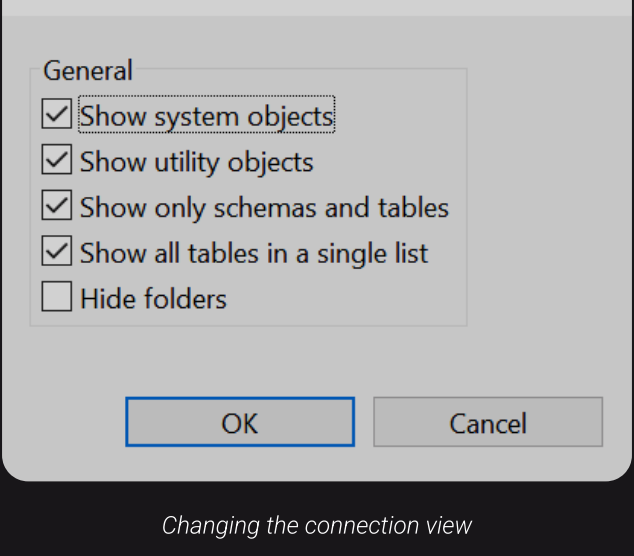
After the connection is established, you can see it in the current connections list at the left side of the main window. If you expand the list item ***Tables***, you will get all the tables in Phoenix.



New Connection
Tables
"employee"
"people_ages"
"SYSTEM"."CATALOG"
"SYSTEM"."FUNCTION"
"SYSTEM"."SEQUENCE"
"SYSTEM".LOG
"SYSTEM".STATS
"test"
BEST_BOOKS
Views

Listing of Phoenix tables

If you do not see the tables, and no errors are displayed, try to change the connection view to ***Custom*** by selecting the context menu item ***Connection View → Custom...*** and setting the necessary flags.



Navigator settings
General
☑ Show system objects
☑ Show utility objects
☑ Show only schemas and tables
☑ Show all tables in a single list
☐ Hide folders

Changing the connection view

CAUTION

If there is no Kerberos authentication in your ADH cluster, the JDBC connection will use the login name `phoenix` by default. This user should have sufficient rights in HBase. For example, if you want only to read data via DBeaver, run the following command in HBase shell:

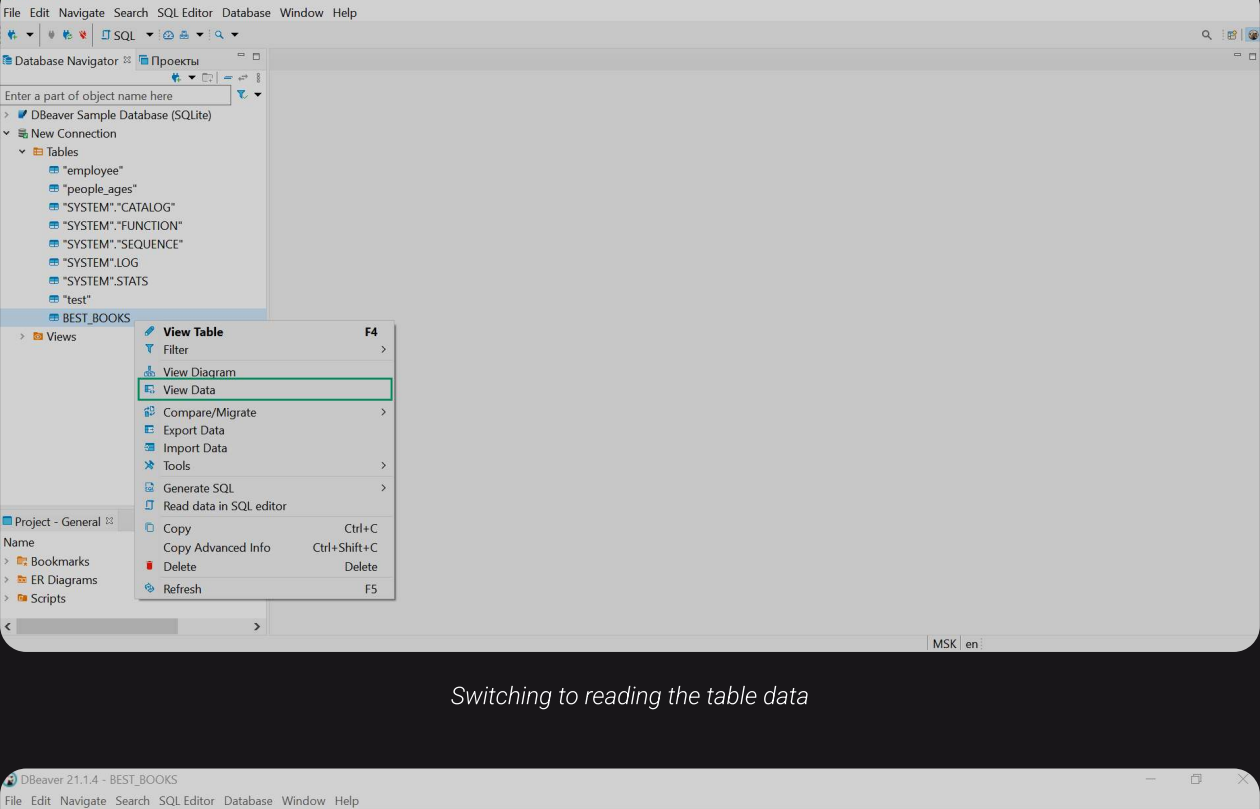
```
grant 'phoenix', 'R'
```

Otherwise, you will get the following error in DBeaver when reading tables:

```
Insufficient permissions for user 'phoenix' (table=SYSTEM.CATALOG, action=READ).
```

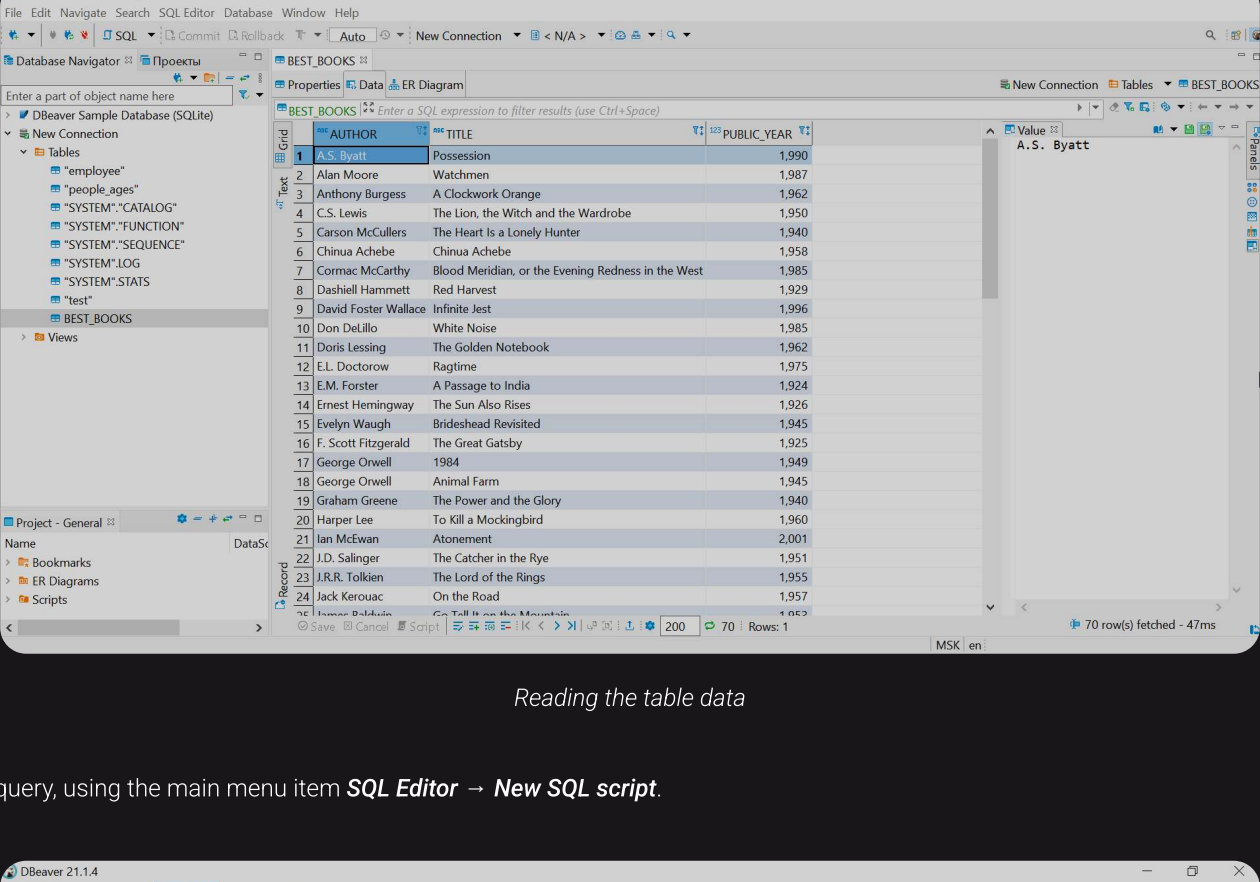
For more information about the `grant` command, see grant in References.

To get the content of a table, select the ***View Data*** item in its context menu.



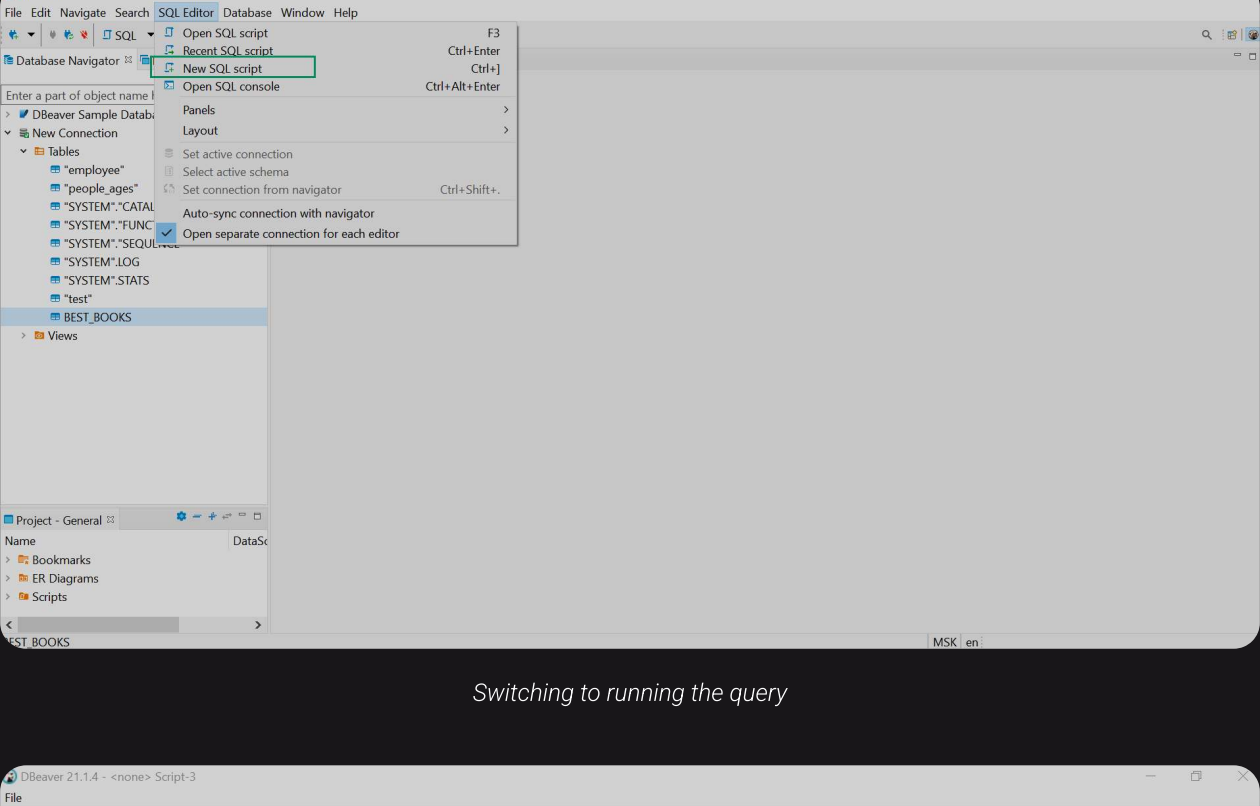
View Table
Filter
View Diagram
View Data
Compare/Migrate
Export Data
Import Data
Tools
Generate SQL
Read data in SQL editor
Copy
Copy Advanced Info
Delete
Refresh

Switching to reading the table data



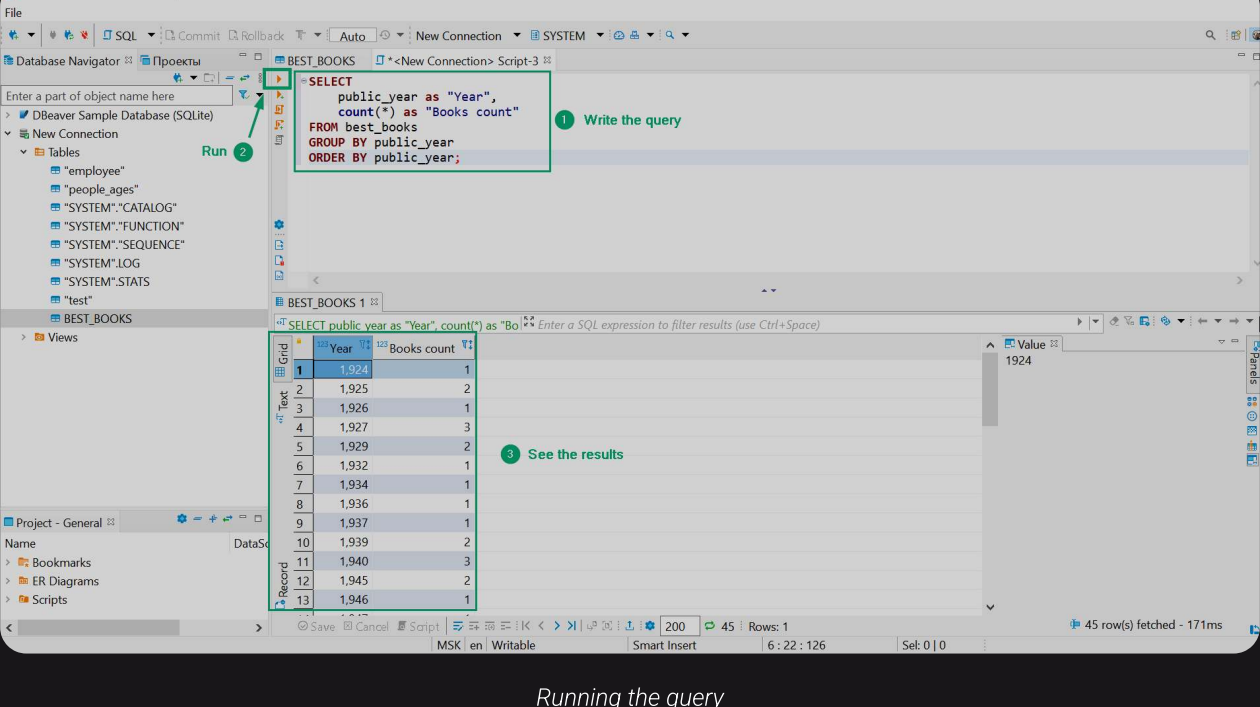
Reading the table data

You can also run any other SQL query, using the main menu item ***SQL Editor → New SQL script***.



SQL Editor
Open SQL script
Recent SQL script
New SQL script
Open SQL console

Switching to running the query



Run
2
1
Write the query
3
See the results

```
SELECT
    public_year as "Year",
    count(*) as "Books count"
FROM best_books
GROUP BY public_year
ORDER BY public_year;
```

Running the query

Use SQLLine to work with Phoenix

Daria Barysheva

Contents

To Table of Contents

Step 1. Create a table
Step 2. Get information about the table
Step 3. Insert new data into the table
Step 4. Select data from the table
Step 5. Update data in the table
Step 6. Delete data from the table
Step 7. Alter the table
Step 8. Drop the table

The SQLLine section describes how to connect to Phoenix via SQLLine. The main data operations available in Phoenix, while working with SQLLine, are listed below. They are based on the example described in the HBase data model section. We use this example to show how to work with data both in HBase and Phoenix in order to understand the similarities and differences between them.

The full list of these commands as well as other grammar specifications you can find in Phoenix documentation.

NOTE

Before executing these commands, we recommend you to read about HBase data model and basic data operations in HBase.

Step 1. Create a table

Most syntax constructions in Phoenix correspond to SQL. To create a new table, use the CREATE TABLE keywords, then define a table name in the double quotes, and after it describe all the columns with their types in the round brackets. If you want to distribute columns between different column families in HBase, you should separate column family names from column qualifiers by dots `.`, as shown below. The default column family name is `0`. After column descriptions, you can also set some table-scope parameters.

The following command creates the `articles` table with the primary key `row_id`, which will be stored in HBase as a row key. Besides, the command defines two columns inside of the column family `basic` (`author`, `header`) and four columns in the column family `tags` (`arch`, `concepts`, `tutorials`, `ref`). We also define that all columns will store five versions of each data value.

TIP

Use double quotes `"` for table names, column family names, and column qualifiers in this and all subsequent operations. Otherwise, Phoenix will convert all characters in the query to the uppercase by default. For example, `ARTICLES` instead of `articles`.

```
CREATE TABLE "articles" ("row_id" VARCHAR PRIMARY KEY, "basic"."author" VARCHAR, "basic"."header" VARCHAR, "tags"."arch" BOOLEAN,
"tags"."concepts" BOOLEAN, "tags"."tutorials" BOOLEAN, "tags"."ref" BOOLEAN) VERSIONS=5;
```

The output is similar to the following:

```
No rows affected (1.313 seconds)
```

Step 2. Get information about the table

To check existence of the table and to get its description, you can use one of the following SQLLine commands:

- !tables — returns the list of all tables in Phoenix:

```
!tables
```

The output:

```
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------------+---------------+---------------+-----------------+------------+-------------+----------------+-
---------------------+---------------------+-----------------------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  |  TABLE_TYPE   | REMARKS  | TYPE_NAME  | SELF_REFERENCING_COL_NAME  | REF_GENERATION
| INDEX_STATE  | IMMUTABLE_ROWS  | SALT_BUCKETS  | MULTI_TENANT  | VIEW_STATEMENT  | VIEW_TYPE  | INDEX_TYPE  | TRANSACTIONAL  |
IS_NAMESPACE_MAPPED  | GUIDE_POSTS_WIDTH  | TRANSACTION_PROVIDER  |
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------------+---------------+---------------+-----------------+------------+-------------+----------------+-
---------------------+---------------------+-----------------------+
|            | SYSTEM       | CATALOG     | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | FUNCTION    | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | LOG         | SYSTEM TABLE  |          |            |                            |
|              | true            | 32            | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | SEQUENCE    | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | STATS       | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            |              | articles    | TABLE         |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------------+---------------+---------------+-----------------+------------+-------------+----------------+-
---------------------+---------------------+-----------------------+
```

- !describe — returns a description for the defined table containing all column names, data types, and other attributes:

```
!describe "articles"
```

The output:

```
+------------+--------------+-------------+--------------+------------+------------+--------------+----------------+-------------
----+-----------------+-----------+----------+------------+----------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  | COLUMN_NAME  | DATA_TYPE  | TYPE_NAME  | COLUMN_SIZE  | BUFFER_LENGTH  |
DECIMAL_DIGITS  | NUM_PREC_RADIX  | NULLABLE  | REMARKS  | COLUMN_DEF  | SQL_DATA |
+------------+--------------+-------------+--------------+------------+------------+--------------+----------------+-------------
----+-----------------+-----------+----------+------------+----------+
|            |              | articles    | row_id       | 12         | VARCHAR    | null         | null           | null
| null            | 0         |          |            | null     |
|            |              | articles    | author       | 12         | VARCHAR    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | header       | 12         | VARCHAR    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | arch         | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | concepts     | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | tutorials    | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | ref          | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
+------------+--------------+-------------+--------------+------------+------------+--------------+----------------+-------------
----+-----------------+-----------+----------+------------+----------+
```

If the command output does not fit in the console window, you can log out from SQLLine (using the `!quit` command) and run the following command to expand the output width:

```
$ stty cols 400
```

Another way is to run the following command for line-by-line data output in SQLLine (to return back, use `horizontal`):

```
!outputformat vertical
```

In addition to using SQLLine, you can also check the created table in HBase shell using the `exists` and `describe` commands. Check that the table exists and all defined column families are created:

```
exists 'articles'
```

The output:

```
Table articles does exist
Took 0.0453 seconds
=> true
```

```
describe 'articles'
```

The output:

```
Table articles is ENABLED
articles, {TABLE_ATTRIBUTES => {coprocessor$1 => '|org.apache.phoenix.coprocessor.ScanRegionObserver|805306366|', coprocessor$2 =>
'|org.apache.phoenix.coprocessor.UngroupedAggregateRegionObserver|805306366|', coprocessor$3 => '|org.apache.phoenix.coprocessor.
GroupedAggregateRegionObserver|805306366|', coprocessor$4 => '|org.apache.phoenix.coprocessor.
ServerCachingEndpointImpl|805306366|', coprocessor$5 => '|org.apache.phoenix.hbase.index.Indexer|805306366|index.builder=org.
apache.phoenix.index.PhoenixIndexBuilder,org.apache.hadoop.hbase.index.codec.class=org.apache.phoenix.index.PhoenixIndexCodec'}
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'FAST_DIFF', TTL => 'FOREVER', MIN_VERSIONS => '0',
REPLICATION_SCOPE => '0', BLOOMFILTER => 'NONE', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE =>
'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'tags', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'FAST_DIFF', TTL => 'FOREVER', MIN_VERSIONS => '0',
REPLICATION_SCOPE => '0', BLOOMFILTER => 'NONE', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE =>
'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
2 row(s)
Took 0.0356 seconds
```

Step 3. Insert new data into the table

To add new data into the created table, you should use the UPSERT command. It combines two SQL commands: `INSERT` and `UPDATE`. If there is a row with the specified primary key, the data is updated, otherwise it is inserted.

The following command adds new data for the row key `article1`. If column names are not defined after the table name (in the round brackets), the values are inserted in the order defined during the table creation. The last column `ref` is not defined, so it will not be stored in HBase.

TIP

To define `CHAR` and `VARCHAR` values, use single quotes `'`, not double quotes.

```
UPSERT INTO "articles" VALUES('article1', 'Test author', 'Test article', true, true, true);
```

The following command adds new data for the row key `article2`. Column names are explicitly specified after the table name; the missed columns will not be stored in HBase:

```
UPSERT INTO "articles"("row_id", "basic"."author", "basic"."header", "tags"."ref") VALUES('article2', 'Test author2', 'Test
article2', true);
```

The output of each command is similar to the following:

```
1 row affected (0.018 seconds)
```

Step 4. Select data from the table

To select all data from the created table, use the SELECT command:

```
SELECT * FROM "articles";
```

The output:

```
+-----------+---------------+----------------+-------+-----------+------------+-------+
|  row_id   |    author     |     header     | arch  | concepts  | tutorials  |  ref  |
+-----------+---------------+----------------+-------+-----------+------------+-------+
| article1  | Test author   | Test article   | true  | true      | true       |       |
| article2  | Test author2  | Test article2  |       |           |            | true  |
+-----------+---------------+----------------+-------+-----------+------------+-------+
2 rows selected (0.06 seconds)
```

You can use different SQL constructions for filtering, sorting, and grouping the output results: `WHERE`, `GROUP BY`, `ORDER BY`, and other. Subqueries are also allowed.

The following query selects the data with the specified primary key value:

```
SELECT * FROM "articles" WHERE "row_id" = 'article1';
```

The result should look like this:

```
+-----------+--------------+---------------+-------+-----------+------------+------+
|  row_id   |    author    |    header     | arch  | concepts  | tutorials  | ref  |
+-----------+--------------+---------------+-------+-----------+------------+------+
| article1  | Test author  | Test article  | true  | true      | true       |      |
+-----------+--------------+---------------+-------+-----------+------------+------+
1 row selected (0.038 seconds)
```

You can also check the table content in HBase shell by running the `scan` command:

```
scan 'articles'
```

In the command output, you can see all the added values. In addition to the user-defined columns, Phoenix also adds one service column for each row key: in our example, with the encoded name `\x00\x00\x00\x00` and the encoded value `x`. This column is used for performance improvements during scan operations — do not change it.

TIP

Notice that data added by Phoenix is encoded. That is why it is not recommended to use HBase API to work with such data as it requires decoding. Use Phoenix instead of it.

```
ROW                                COLUMN+CELL
 article1                          column=basic:\x00\x00\x00\x00, timestamp=1638285442663, value=x
 article1                          column=basic:\x80\x0B, timestamp=1638285442663, value=Test author
 article1                          column=basic:\x80\x0C, timestamp=1638285442663, value=Test article
 article1                          column=tags:\x80\x0D, timestamp=1638285442663, value=\x01
 article1                          column=tags:\x80\x0E, timestamp=1638285442663, value=\x01
 article1                          column=tags:\x80\x0F, timestamp=1638285442663, value=\x01
 article2                          column=basic:\x00\x00\x00\x00, timestamp=1638285450913, value=x
 article2                          column=basic:\x80\x0B, timestamp=1638285450913, value=Test author2
 article2                          column=basic:\x80\x0C, timestamp=1638285450913, value=Test article2
 article2                          column=tags:\x80\x10, timestamp=1638285450913, value=\x01
2 row(s)
Took 0.0214 seconds
```

Step 5. Update data in the table

To update some data values in the table, use the same command as for inserting new data, that is, UPSERT. The following command updates the value of the `header` column in the row with the `article1` key:

```
UPSERT INTO "articles"("row_id", "basic"."header") VALUES('article1', 'Test article. Version 2');
```

The output:

```
1 row affected (0.015 seconds)
```

Now, if you apply the `SELECT` command to the table, you will see the updated data:

```
SELECT * FROM "articles";
```

The output:

```
+-----------+---------------+--------------------------+-------+-----------+------------+-------+
|  row_id   |    author     |          header          | arch  | concepts  | tutorials  |  ref  |
+-----------+---------------+--------------------------+-------+-----------+------------+-------+
| article1  | Test author   | Test article. Version 2  | true  | true      | true       |       |
| article2  | Test author2  | Test article2            |       |           |            | true  |
+-----------+---------------+--------------------------+-------+-----------+------------+-------+
2 rows selected (0.041 seconds)
```

Check it also in HBase shell by executing the following command:

```
scan 'articles'
```

Its result is similar:

```
ROW                                COLUMN+CELL
 article1                          column=basic:\x00\x00\x00\x00, timestamp=1638285553403, value=x
 article1                          column=basic:\x80\x0B, timestamp=1638285442663, value=Test author
 article1                          column=basic:\x80\x0C, timestamp=1638285553403, value=Test article. Version 2
 article1                          column=tags:\x80\x0D, timestamp=1638285442663, value=\x01
 article1                          column=tags:\x80\x0E, timestamp=1638285442663, value=\x01
 article1                          column=tags:\x80\x0F, timestamp=1638285442663, value=\x01
 article2                          column=basic:\x00\x00\x00\x00, timestamp=1638285450913, value=x
 article2                          column=basic:\x80\x0B, timestamp=1638285450913, value=Test author2
 article2                          column=basic:\x80\x0C, timestamp=1638285450913, value=Test article2
 article2                          column=tags:\x80\x10, timestamp=1638285450913, value=\x01
2 row(s)
Took 0.0125 seconds
```

However, as we specified the need to store five versions of data values when creating the table, HBase stores not only the last version, but also the previous one. To see this, it is necessary to run in HBase shell the `scan` command with the `TIMERANGE` argument:

```
----ma
scan 'articles', {TIMERANGE => [1638285442663, 1638285450914]}
----
```

The output represents value versions within the specified timestamp range:

```
ROW                                COLUMN+CELL
 article1                          column=basic:\x00\x00\x00\x00, timestamp=1638285442663, value=x
 article1                          column=basic:\x80\x0B, timestamp=1638285442663, value=Test author
 article1                          column=basic:\x80\x0C, timestamp=1638285442663, value=Test article
 article1                          column=tags:\x80\x0D, timestamp=1638285442663, value=\x01
 article1                          column=tags:\x80\x0E, timestamp=1638285442663, value=\x01
 article1                          column=tags:\x80\x0F, timestamp=1638285442663, value=\x01
 article2                          column=basic:\x00\x00\x00\x00, timestamp=1638285450913, value=x
 article2                          column=basic:\x80\x0B, timestamp=1638285450913, value=Test author2
 article2                          column=basic:\x80\x0C, timestamp=1638285450913, value=Test article2
 article2                          column=tags:\x80\x10, timestamp=1638285450913, value=\x01
2 row(s)
Took 0.0112 seconds
```

Step 6. Delete data from the table

To delete data values from the table, use the DELETE command. The following command deletes all the table rows, where the value of the `header` column is equal to `Test article. Version 2`.

```
DELETE FROM "articles" WHERE "basic"."header" = 'Test article. Version 2';
```

The output:

```
1 row affected (0.018 seconds)
```

CAUTION

Notice that unlike HBase, when you run `DELETE` in Phoenix, not only the latest value version is deleted, but also all versions. This works even if the need to store multiple values versions was specified when creating the table.

If you apply `SELECT` to the table now, you will see only one row:

```
SELECT * FROM "articles";
```

The output shows that the first row has been deleted:

```
+-----------+---------------+----------------+-------+-----------+-------------+
|  row_id   |    author     |     header     | arch  | concepts  | tutorials   |
+-----------+---------------+----------------+-------+-----------+-------------+
| article2  | Test author2  | Test article2  |       |           |             |
+-----------+---------------+----------------+-------+-----------+-------------+
1 row selected (0.042 seconds)
```

HBase shell returns a result similar to the previous one:

```
scan 'articles'
```

The output:

```
ROW                                COLUMN+CELL
 article2                          column=basic:\x00\x00\x00\x00, timestamp=1638283068620, value=x
 article2                          column=basic:\x80\x0B, timestamp=1638283068620, value=Test author2
 article2                          column=basic:\x80\x0C, timestamp=1638283068620, value=Test article2
 article2                          column=tags:\x80\x10, timestamp=1638283068620, value=\x01
1 row(s)
Took 0.0101 seconds
```

The following command deletes the row of the table with the specified primary key value:

```
DELETE FROM "articles" WHERE "row_id" = 'article2';
```

After this operation, the table becomes empty:

```
SELECT * FROM "articles";
```

The output:

```
+---------+---------+---------+-------+-----------+-----------+------+
| row_id  | author  | header  | arch  | concepts  | tutorials  | ref  |
+---------+---------+---------+-------+-----------+-----------+------+
+---------+---------+---------+-------+-----------+-----------+------+
No rows selected (0.027 seconds)
```

Step 7. Alter the table

To change the table structure, use the ALTER TABLE command. The following command adds a new column family `temp` and the `review` column in it:

```
ALTER TABLE "articles" ADD "temp"."review" VARCHAR VERSIONS=3;
```

The result is:

```
No rows affected (6.604 seconds)
```

Check the updated structure using the `!describe` command in SQLLine:

```
!describe "articles"
```

The output contains the added column `review`:

```
+------------+--------------+-------------+--------------+------------+------------+--------------+----------------+-------------
----+-----------------+-----------+----------+------------+----------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  | COLUMN_NAME  | DATA_TYPE  | TYPE_NAME  | COLUMN_SIZE  | BUFFER_LENGTH  |
DECIMAL_DIGITS  | NUM_PREC_RADIX  | NULLABLE  | REMARKS  | COLUMN_DEF  | SQL_DATA |
+------------+--------------+-------------+--------------+------------+------------+--------------+----------------+-------------
----+-----------------+-----------+----------+------------+----------+
|            |              | articles    | row_id       | 12         | VARCHAR    | null         | null           | null
| null            | 0         |          |            | null     |
|            |              | articles    | author       | 12         | VARCHAR    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | header       | 12         | VARCHAR    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | arch         | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | concepts     | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | tutorials    | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | ref          | 16         | BOOLEAN    | null         | null           | null
| null            | 1         |          |            | null     |
|            |              | articles    | review       | 12         | VARCHAR    | null         | null           | null
| null            | 1         |          |            | null     |
+------------+--------------+-------------+--------------+------------+------------+--------------+----------------+-------------
----+-----------------+-----------+----------+------------+----------+
```

In HBase shell, use the `describe` command to see the added column family:

```
describe 'articles'
```

The output:

```
Table articles is ENABLED
articles, {TABLE_ATTRIBUTES => {coprocessor$1 => '|org.apache.phoenix.coprocessor.ScanRegionObserver|805306366|', coprocesso
r$2 => '|org.apache.phoenix.coprocessor.UngroupedAggregateRegionObserver|805306366|', coprocessor$3 => '|org.apache.phoenix.
coprocessor.GroupedAggregateRegionObserver|805306366|', coprocessor$4 => '|org.apache.phoenix.coprocessor.ServerCachingEndpo
intImpl|805306366|', coprocessor$5 => '|org.apache.phoenix.hbase.index.Indexer|805306366|index.builder=org.apache.phoenix.in
dex.PhoenixIndexBuilder,org.apache.hadoop.hbase.index.codec.class=org.apache.phoenix.index.PhoenixIndexCodec'}
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'FAST_DIFF', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICAT
ION_SCOPE => '0', BLOOMFILTER => 'NONE', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'fa
lse', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'tags', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => '
FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'FAST_DIFF', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATI
ON_SCOPE => '0', BLOOMFILTER => 'NONE', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'fal
se', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'temp', VERSIONS => '3', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS => '
FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'FAST_DIFF', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATI
ON_SCOPE => '0', BLOOMFILTER => 'NONE', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'fal
se', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
3 row(s)
Took 0.0222 seconds
```

Step 8. Drop the table

To delete the table permanently, use the DROP TABLE command:

```
DROP TABLE "articles";
```

The output:

```
No rows affected (1.246 seconds)
```

Now, if you run the `!tables` command, you will see that the `articles` table does not exist anymore:

```
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------------+---------------+---------------+-----------------+------------+-------------+----------------+-
---------------------+---------------------+-----------------------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  |  TABLE_TYPE   | REMARKS  | TYPE_NAME  | SELF_REFERENCING_COL_NAME  | REF_GENERATION
| INDEX_STATE  | IMMUTABLE_ROWS  | SALT_BUCKETS  | MULTI_TENANT  | VIEW_STATEMENT  | VIEW_TYPE  | INDEX_TYPE  | TRANSACTIONAL  |
IS_NAMESPACE_MAPPED  | GUIDE_POSTS_WIDTH  | TRANSACTION_PROVIDER  |
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------------+---------------+---------------+-----------------+------------+-------------+----------------+-
---------------------+---------------------+-----------------------+
|            | SYSTEM       | CATALOG     | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | FUNCTION    | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | LOG         | SYSTEM TABLE  |          |            |                            |
|              | true            | 32            | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | SEQUENCE    | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
|            | SYSTEM       | STATS       | SYSTEM TABLE  |          |            |                            |
|              | false           | null          | false         |                 |            |             | false          |
false                | null                |                       |
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------------+---------------+---------------+-----------------+------------+-------------+----------------+-
---------------------+---------------------+-----------------------+
```

JOIN clause in Phoenix

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
- JOIN types
  - INNER JOIN
  - LEFT/RIGHT OUTER JOIN
  - FULL OUTER JOIN
  - SELF JOIN
  - CROSS JOIN
  - GROUP JOIN
- Exclusion of INNER part from OUTER JOIN
- JOIN with indices
- Hints
  - Hash and sort-merge joins
  - Star join optimization

Overview

Phoenix fully supports the standard SQL `JOIN` syntax. It allows combining records from different tables if they have common values.

In this article, three example tables with the following columns are used.

Example tables

Table	Column	Description
states	name	State name
	abbr	State abbreviation
	capitol	State capital
	largest	State largest city
enterprises	name	Enterprise name
	country	Country name where the enterprise is located
	state	State abbreviation (in USA) or region name where the enterprise is located
	town	Town name where the enterprise is located
people	name	Name of the person
	age	Age of the person
	town	Town name where the person lives
	state	State abbreviation (in USA) or region name where the person lives
	country	Country name where the person lives

To be able to follow examples given in this article, you need to create those tables in your Phoenix environment. Do the following:

1. Download the CSV files with sample data: people.csv, enterprises.csv, and states.csv.
2. Put the downloaded files on the filesystem of the host to which the ***Phoenix Query Server*** component is mapped, to the directory of your choice.
3. Connect to this host and load the Phoenix shell by executing the following command:

```
$ /usr/lib/phoenix/bin/sqlline.py
```

4. Create the empty tables by executing the following commands:

```
CREATE TABLE enterprises (
name VARCHAR(50) NOT NULL,
country VARCHAR(50),
state VARCHAR(50),
town VARCHAR(50)
CONSTRAINT pk PRIMARY KEY (name));
```

```
CREATE TABLE people (
name VARCHAR(50) NOT NULL,
age SMALLINT,
town VARCHAR(50),
state VARCHAR(50),
country VARCHAR(50)
CONSTRAINT pk PRIMARY KEY (name));
```

```
CREATE TABLE states (
name VARCHAR(50) NOT NULL,
abbr VARCHAR(50),
capitol VARCHAR(50),
largest VARCHAR(50)
CONSTRAINT pk PRIMARY KEY (name));
```

5. Exit the shell by pressing `Ctrl+D` and load the data from the CSV files into the tables by executing the following commands:

```
$ /usr/lib/phoenix/bin/psql.py /<csv_dir>/people.csv
$ /usr/lib/phoenix/bin/psql.py /<csv_dir>/enterprises.csv
$ /usr/lib/phoenix/bin/psql.py /<csv_dir>/states.csv
```

where `<csv_dir>` is the path to the directory where the downloaded CSV files are located.

JOIN types

INNER JOIN

The `INNER JOIN` takes each row in the first table and searches for a row in the second table that matches the specified condition. If such a row is found, the `INNER JOIN` composes a row with values of the columns specified in the query. This row is added to the result and `INNER JOIN` proceeds.

This joining mode is used by default, so you may omit the `INNER` modifier in the query text.

For example, to find all american enterprises along with their hometowns and the respective state capitols, you need the following query:

```
SELECT e.name, e.town, s.capitol
FROM states AS s
INNER JOIN enterprises AS e
ON s.abbr=e.state;
```

Result:

```
+------------------+---------------+-------------+
|      E.NAME      |    E.TOWN     |  S.CAPITOL  |
+------------------+---------------+-------------+
| Rockstar         | New York City | Albany      |
| Bethesda         | Rockville     | Annapolis   |
| DigitalAnvil     | Austin        | Austin      |
| Shaw Group       | Baton Rouge   | Baton Rouge |
| Microsoft        | Redmond       | Olympia     |
| Avnet            | Phoenix       | Phoenix     |
| Apple            | Cupertino     | Sacramento  |
| Obsidian         | Irvine        | Sacramento  |
| Sierra           | Los Angeles   | Sacramento  |
| Intel            | Sacramento    | Sacramento  |
| Northrop Grumman | Albuquerque   | Santa Fe    |
+------------------+---------------+-------------+
```

LEFT/RIGHT OUTER JOIN

The `LEFT OUTER JOIN` works like `INNER JOIN`, but adds values from the first table anyway, regardless of whether the specified condition is satisfied.

For example, to list all the enterprises along with their hometowns, and also the state capitols for those located in the USA, you need the following query:

```
SELECT e.name, e.town, s.capitol
FROM enterprises AS e
LEFT OUTER JOIN states AS s
ON e.state=s.abbr;
```

Result:

```
+------------------+---------------+-------------+
|      E.NAME      |    E.TOWN     |  S.CAPITOL  |
+------------------+---------------+-------------+
| Northrop Grumman | Albuquerque   | Santa Fe    |
| DigitalAnvil     | Austin        | Austin      |
| Shaw Group       | Baton Rouge   | Baton Rouge |
| Apple            | Cupertino     | Sacramento  |
| Obsidian         | Irvine        | Sacramento  |
| Eidos            | London        |             |
| Sierra           | Los Angeles   | Sacramento  |
| Alluxi           | Monterrey     |             |
| Rockstar         | New York City | Albany      |
| Empresa          | Nogales       |             |
| Avnet            | Phoenix       | Phoenix     |
| Microsoft        | Redmond       | Olympia     |
| Bethesda         | Rockville     | Annapolis   |
| Intel            | Sacramento    | Sacramento  |
| Ubisoft          | Saint-Mande   |             |
+------------------+---------------+-------------+
```

The `RIGHT OUTER JOIN` does the same, but in regard to the second table instead of the first.

For example, to list all the USA states along with their capitols, and also the enterprises that may be located in those states along with their hometowns, you need the following query:

```
SELECT s.name, s.capitol, e.name, e.town
FROM enterprises AS e
RIGHT OUTER JOIN states AS s
ON e.state=s.abbr
ORDER BY s.name
LIMIT 20;
```

Result:

```
+-------------+--------------+----------+-------------+
|   S.NAME    |  S.CAPITOL   |  E.NAME  |   E.TOWN    |
+-------------+--------------+----------+-------------+
| Alabama     | Montgomery   |          |             |
| Alaska      | Juneau       |          |             |
| Arizona     | Phoenix      | Avnet    | Phoenix     |
| Arkansas    | Little Rock  |          |             |
| California  | Sacramento   | Apple    | Cupertino   |
| California  | Sacramento   | Obsidian | Irvine      |
| California  | Sacramento   | Sierra   | Los Angeles |
| California  | Sacramento   | Intel    | Sacramento  |
| Colorado    | Denver       |          |             |
| Connecticut | Hartford     |          |             |
| Delaware    | Dover        |          |             |
| Florida     | Tallahassee  |          |             |
| Georgia     | Atlanta      |          |             |
| Hawaii      | Honolulu     |          |             |
| Idaho       | Boise        |          |             |
| Illinois    | Springfield  |          |             |
| Indiana     | Indianapolis |          |             |
| Iowa        | Des Moines   |          |             |
| Kansas      | Topeka       |          |             |
| Kentucky    | Frankfort    |          |             |
+-------------+--------------+----------+-------------+
```

The `OUTER` modifier is implied with the `LEFT` and `RIGHT` modifiers, so it can be omitted.

FULL OUTER JOIN

The `FULL OUTER JOIN` unites the `LEFT JOIN` and the `RIGHT JOIN`: it combines the rows that satisfy the specified condition with the rows that do not from both tables.

Example:

```
SELECT e.name, e.town, s.capitol
FROM enterprises AS e
FULL OUTER JOIN states AS s
ON e.state=s.abbr
LIMIT 20;
```

Result:

```
+------------------+---------------+----------------+
|      E.NAME      |    E.TOWN     |   S.CAPITOL    |
+------------------+---------------+----------------+
|                  |               | Juneau         |
|                  |               | Montgomery     |
|                  |               | Little Rock    |
| Avnet            | Phoenix       | Phoenix        |
| Apple            | Cupertino     | Sacramento     |
| Obsidian         | Irvine        | Sacramento     |
| Intel            | Sacramento    | Sacramento     |
| Sierra           | Los Angeles   | Sacramento     |
|                  |               | Denver         |
|                  |               | Hartford       |
|                  |               | Dover          |
| Eidos            | London        |                |
|                  |               | Tallahassee    |
|                  |               | Atlanta        |
|                  |               | Honolulu       |
|                  |               | Des Moines     |
|                  |               | Boise          |
|                  |               | Springfield    |
|                  |               | Indianapolis   |
| Ubisoft          | Saint-Mande   |                |
+------------------+---------------+----------------+
```

The `OUTER` modifier is implied with the `FULL` modifier, so it can be omitted.

SELF JOIN

You can join a table with itself (e.g., to filter it). For example, to list the states the capitols of which are also their largest cities, you need the following query:

```
SELECT s.name, s.capitol
FROM states AS s
JOIN states AS ss
ON s.capitol=ss.largest;
```

Result:

```
+---------------+----------------+
|    S.NAME     |   S.CAPITOL    |
+---------------+----------------+
| Arizona       | Phoenix        |
| Arkansas      | Little Rock    |
| Colorado      | Denver         |
| Georgia       | Atlanta        |
| Hawaii        | Honolulu       |
| Idaho         | Boise          |
| Indiana       | Indianapolis   |
| Iowa          | Des Moines     |
| Massachusetts | Boston         |
| Mississippi   | Jackson        |
| Ohio          | Columbus       |
| Oklahoma      | Oklahoma City  |
| Rhode Island  | Providence     |
| Tennessee     | Nashville      |
| Utah          | Salt Lake City |
| West Virginia | Charleston     |
| West Virginia | Charleston     |
| Wyoming       | Cheyenne       |
+---------------+----------------+
```

CROSS JOIN

Normally, there is not much use for `CROSS JOIN` in Phoenix as the tables are usually very big. For that reason, Phoenix ignores the `CROSS` modifier. However, cross joins are still possible in Phoenix and employ the following syntax:

```
SELECT <col_1>, <col_2>, ... <col_n>
FROM t1, t2;
```

GROUP JOIN

Phoenix supports complex `JOIN` syntax. The result of joining two tables can be joined with another table using parenthesis. Also, instead of any table in a `JOIN` clause, you can use a subquery (derived table), which in turn can also contain its own `JOIN` clause.

For example, to list the people who live in the capitols of the states where there is also an enterprise located, you need the following query:

```
SELECT p.name, e.name, s.capitol
FROM people AS p
JOIN
    (enterprises AS e
    JOIN states AS s
    ON e.town=s.capitol)
ON p.town=e.town
LIMIT 20;
```

Result:

```
+------------------+--------------+-------------+
|      P.NAME      |    E.NAME    |  S.CAPITOL  |
+------------------+--------------+-------------+
| Abbott Howard    | Shaw Group   | Baton Rouge |
| Alvarado Melvin  | DigitalAnvil | Austin      |
| Baker Daniel     | Shaw Group   | Baton Rouge |
| Barnes Mollie    | Shaw Group   | Baton Rouge |
| Becker Samuel    | Shaw Group   | Baton Rouge |
| Bush Cameron     | Shaw Group   | Baton Rouge |
| Carpenter Carrie | Avnet        | Phoenix     |
| Clarke Joe       | Avnet        | Phoenix     |
| Cox Dennis       | Intel        | Sacramento  |
| Cox Shawn        | Avnet        | Phoenix     |
| Daniel Ada       | DigitalAnvil | Austin      |
| Day Hattie       | Intel        | Sacramento  |
| Erickson Mayme   | Intel        | Sacramento  |
| Evans Sophie     | Avnet        | Phoenix     |
| Farmer Dean      | Shaw Group   | Baton Rouge |
| Fisher Gordon    | Intel        | Sacramento  |
| Fletcher Julia   | Avnet        | Phoenix     |
| Garner Juan      | Shaw Group   | Baton Rouge |
| Gill Josephine   | Shaw Group   | Baton Rouge |
| Gonzales Edna    | Shaw Group   | Baton Rouge |
+------------------+--------------+-------------+
```

Exclusion of INNER part from OUTER JOIN

Sometimes you might want to exclude the rows that satisfy the specified condition from the `JOIN` result. To that end, you need to add a `WHERE` clause to the query.

For example, to list all the enterprises outside of USA, you need the following query:

```
SELECT e.name, e.country, e.state, e.town
FROM enterprises AS e
LEFT JOIN states AS s
ON e.state=s.abbr
WHERE s.abbr IS NULL;
```

Result:

```
+---------+-----------+---------------+-------------+
| E.NAME  | E.COUNTRY |    E.STATE    |   E.TOWN    |
+---------+-----------+---------------+-------------+
| Alluxi  | Mexico    | Nuevo Leon    | Monterrey   |
| Eidos   | UK        | England       | London      |
| Empresa | Mexico    | Sonora        | Nogales     |
| Ubisoft | France    | Ile-de-France | Saint-Mande |
+---------+-----------+---------------+-------------+
```

JOIN with indices

You can create secondary indices for your tables which will be used automatically for `JOIN` queries. Example:

```
CREATE INDEX ipeople ON people (town) INCLUDE (state);
CREATE INDEX istates ON states (capitol) INCLUDE (name, abbr);
CREATE INDEX ienterprises ON enterprises (town) INCLUDE (name, state);
```

These indices take precedence when running certain queries. Example:

```
EXPLAIN SELECT p.name, e.name, s.capitol
FROM people AS p
JOIN
    (enterprises AS e
    JOIN states AS s
    ON e.town=s.capitol)
ON p.town=e.town;
```

Result:

```
+----------------------------------------------------------------------------------+----------------+---------------+------------
-+
|                                       PLAN                                       | EST_BYTES_READ | EST_ROWS_READ | EST_INFO_TS
|
+----------------------------------------------------------------------------------+----------------+---------------+------------
-+
| CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER IPEOPLE                 | null           | null          | null
|
|     SERVER FILTER BY FIRST KEY ONLY                                              | null           | null          | null
|
|     PARALLEL INNER-JOIN TABLE 0                                                  | null           | null          | null
|
|         CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER IENTERPRISES    | null           | null          | null
|
|             SERVER FILTER BY FIRST KEY ONLY                                      | null           | null          | null
|
|             PARALLEL INNER-JOIN TABLE 0                                          | null           | null          | null
|
|                 CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER ISTATES | null           | null          | null
|
|                     SERVER FILTER BY FIRST KEY ONLY                              | null           | null          | null
|
|     DYNAMIC SERVER FILTER BY "P.0:TOWN" IN ("E.LOCATION:TOWN")                  | null           | null          | null
|
+----------------------------------------------------------------------------------+----------------+---------------+------------
-+
```

As you can see, for this particular query Phoenix uses all three previously created indices rather than original tables.

Hints

Phoenix uses hints to switch between different `JOIN` ways of operation.

Hash and sort-merge joins

Phoenix employs two types of `JOIN` algorithms: **hash** and **sort-merge**. Hash `JOIN` is the fastest algorithm, so it is used by default. However, it cannot be used if the total size of the tables is too large and exceeds the available memory. In this case, you have to use the sort-merge `JOIN` algorithm by means of the `/*+ USE_SORT_MERGE_JOIN */` hint in queries:

```
SELECT /*+ USE_SORT_MERGE_JOIN */ <values>
FROM <table>
JOIN <clause_condition>
```

Star join optimization

Phoenix uses star join optimization for queries that have multiple `JOIN` clauses. For example, your query has the following structure:

```
SELECT p.name, s.abbr, e.name
FROM people AS p
JOIN states AS s
ON p.state=s.abbr
JOIN enterprises AS e
ON p.state=e.state;
```

When processing such query, Phoenix will load all the tables involved into the RAM and perform both `JOIN` clauses simultaneously. This can be seen in the query plan, which you can get by using the `EXPLAIN` operator:

```
EXPLAIN SELECT p.name, s.abbr, e.name
FROM people AS p
JOIN states AS s
ON p.state=s.abbr
JOIN enterprises AS e
ON p.state=e.state;
```

Result:

```
+----------------------------------------------------------------------------------------------------------+----------------+----
-----------+--------+
|                                                   PLAN                                                   | EST_BYTES_READ |
EST_ROWS_READ | EST_IN |
+----------------------------------------------------------------------------------------------------------+----------------+----
-----------+--------+
| CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER PEOPLE                                          | null           |
null          | null   |
|     PARALLEL INNER-JOIN TABLE 0                                                                          | null           |
null          | null   |
|         CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER STATES                                  | null           |
null          | null   |
|     PARALLEL INNER-JOIN TABLE 1                                                                          | null           |
null          | null   |
|         CLIENT 1-CHUNK 976934 ROWS 314572800 BYTES PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER ENTERPRISES | null           |
null          | null   |
+----------------------------------------------------------------------------------------------------------+----------------+----
-----------+--------+
```

Sometimes, the tables can be very big to be loaded into the RAM at the same time. In such cases, you need to turn the star join optimization off by using the `/+ NO_STAR_JOIN/` hint in the query:

```
EXPLAIN SELECT /*+ NO_STAR_JOIN*/ p.name, s.abbr, e.name
FROM people AS p
JOIN states AS s
ON p.state=s.abbr
JOIN enterprises AS e
ON p.state=e.state;
```

Result:

```
+--------------------------------------------------------------------------------------------------+----------------+------------
---+-------------+
|                                               PLAN                                               | EST_BYTES_READ |
EST_ROWS_READ | EST_INFO_TS |
+--------------------------------------------------------------------------------------------------+----------------+------------
---+-------------+
| CLIENT 1-CHUNK 976934 ROWS 314572800 BYTES PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER ENTERPRISES | null           | null
| null        |
|     PARALLEL INNER-JOIN TABLE 0                                                                  | null           | null
| null        |
|         CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER PEOPLE                          | null           | null
| null        |
|             PARALLEL INNER-JOIN TABLE 0                                                          | null           | null
| null        |
|                 CLIENT 1-CHUNK PARALLEL 1-WAY ROUND ROBIN FULL SCAN OVER STATES                  | null           | null
| null        |
+--------------------------------------------------------------------------------------------------+----------------+------------
---+-------------+
```

As can be seen, in this case, the `JOIN` operations are performed one by one.

Subqueries in Phoenix

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
- IN
- EXISTS
- Comparison operators
  - Unmodified
  - Modified
- Correlated subqueries
- Branches and nesting
  - AND and OR branches
  - Nesting
- Derived tables
- Limitations

Overview

Phoenix supports subqueries contained in `WHERE` and `FROM` clauses. You can use logical operators and comparison operators, both modified and unmodified, when composing subqueries.

To be able to follow examples given in this article, you need to create the example tables in your Phoenix environment as described in the JOIN clause in Phoenix article.

IN

The `IN` operator allows you to search the result of a subquery. The subquery in the parenthesis returns a table (possibly empty), and the `IN` operator treats this table as the set of values among which to search for matches according to the specified condition.

For example, to list the people who live in american states where there are also enterprises located, you need the following query:

```
SELECT p.name, p.state, p.town
FROM people AS p
WHERE p.state IN
    (SELECT e.state
    FROM enterprises AS e
    WHERE e.country='USA')
LIMIT 20;
```

Result:

```
+-------------------+---------+---------------+
|      P.NAME       | P.STATE |    P.TOWN     |
+-------------------+---------+---------------+
| Abbott Delia      | TX      | Dallas        |
| Abbott Howard     | LA      | Baton Rouge   |
| Adams Clyde       | LA      | Lafayette     |
| Aguilar Terry     | CA      | Fresno        |
| Alexander Derrick | TX      | Dallas        |
| Alexander Gregory | LA      | Lafayette     |
| Alexander Leon    | AZ      | Kingman       |
| Alvarado Melvin   | TX      | Austin        |
| Alvarado Timothy  | CA      | San Francisco |
| Alvarez Bruce     | CA      | Modesto       |
| Andrews Lucy      | AZ      | Kingman       |
| Armstrong Marie   | NM      | Santa Fe      |
| Atkins Gene       | TX      | Dallas        |
| Austin Eugene     | LA      | Lafayette     |
| Austin Travis     | TX      | El Paso       |
| Bailey Cameron    | TX      | Lubbock       |
| Baker Daniel      | LA      | Baton Rouge   |
| Ball Nelle        | NM      | Albuquerque   |
| Barber Minerva    | TX      | Beaumont      |
| Barnes Mollie     | LA      | Baton Rouge   |
+-------------------+---------+---------------+
```

You can also search for multiple column values. The number of columns in the `WHERE` clause must be the same as the number of columns in the `SELECT` clause of the subquery. If it is more than one, then the columns in the `WHERE` clause must be enclosed in parenthesis.

For example, to list the enterprises which are located in state capitols also being the largest city, you need the following query:

```
SELECT e.name, e.state, e.town
FROM enterprises AS e
WHERE (e.state, e.town) IN
    (SELECT s.abbr, s.capitol
    FROM states AS s
    WHERE s.capitol=s.largest);
```

Result:

```
+--------+---------+---------+
| E.NAME | E.STATE | E.TOWN  |
+--------+---------+---------+
| Avnet  | AZ      | Phoenix |
+--------+---------+---------+
```

The `NOT` keyword inverts the `IN` clause. Using the `NOT IN` operator will search among the values that are not the results of the subquery.

For example, to list the people who live in the states where there are no 44 years old people, you need the following query:

```
SELECT p.name, p.state, p.age
FROM people AS p
WHERE p.state NOT IN
    (SELECT pp.state
    FROM people AS pp
    WHERE pp.age=44)
LIMIT 20;
```

Result:

```
+-------------------+---------+-------+
|      P.NAME       | P.STATE | P.AGE |
+-------------------+---------+-------+
| Alexander Leon    | AZ      | 42    |
| Andrews Lucy      | AZ      | 33    |
| Becker Marie      | AZ      | 19    |
| Bowman Randall    | AZ      | 46    |
| Bryant Marcus     | OR      | 30    |
| Buchanan Todd     | OR      | 25    |
| Burton Luella     | AZ      | 62    |
| Byrd Nannie       | OR      | 59    |
| Carpenter Carrie  | AZ      | 24    |
| Chambers Hettie   | AZ      | 52    |
| Clarke Joe        | AZ      | 26    |
| Cox Shawn         | AZ      | 25    |
| Cunningham Sallie | AZ      | 56    |
| Duncan Ollie      | AZ      | 45    |
| Elliott Myrtie    | OR      | 59    |
| Evans Sophie      | AZ      | 58    |
| Ferguson Josie    | AZ      | 28    |
| Fletcher Julia    | AZ      | 38    |
| Fowler Hettie     | OR      | 36    |
| Fowler May        | OR      | 64    |
+-------------------+---------+-------+
```

EXISTS

The `EXISTS` operator is logical. It returns `true` if the subquery result is not empty, and `false` otherwise. As the query is processed, the subquery in the parenthesis is executed for every iteration of the outer query. Also, for an `EXISTS` subquery to have any sense, it needs to be correlated to the outer query, i.e. to have some reference to its conditions. If there is no correlation between the queries, then the `EXISTS` clause will always resolve to the same value regardless of the outer query.

For example, to list american states where people of age 65 live, you need the following query:

```
SELECT s.name
FROM states AS s
WHERE EXISTS
    (SELECT p.state
    FROM people AS p
    WHERE p.age=65 AND p.state=s.abbr);
```

As you can see, in this example, the subquery contains the reference to the `states` table, as does the outer query.

Result:

```
+------------+
|   S.NAME   |
+------------+
| Arizona    |
| California |
| Louisiana  |
| Nevada     |
| Oregon     |
| Texas      |
+------------+
```

The `NOT` keyword inverts the logical operator. The `NOT EXISTS` operator will return `true` if the subquery result is empty, and `false` otherwise.

For example, to list all the states outside USA from the `people` table, you need the following query:

```
SELECT p.state
FROM people AS p
WHERE NOT EXISTS
    (SELECT s.abbr
    FROM states AS s
    WHERE s.abbr=p.state)
GROUP BY p.state;
```

Result:

```
+-----------------+
|     P.STATE     |
+-----------------+
| California Baja |
| Chihuahua       |
| Quintana Roo    |
| Sonora          |
| Yucatan         |
+-----------------+
```

Comparison operators

You can use comparison operators in the `WHERE` clause with the subquery being the right operand. Phoenix uses the following comparison operators:

- `=` — equal to;
- `!=` or `<>` — not equal to;
- `>` — greater than;
- `<` — less than;
- `>=` or `!<` — greater than or equal to;
- `<=` or `!>` — less than or equal to.

Unmodified

Unmodified comparison operators are used as they are in ordinary comparison operations, without any modificators.

For example, to list the distribution of ages of people older than average, you need the following query:

```
SELECT p.age, count(p.age)
FROM people AS p
WHERE p.age >
    (SELECT avg(p.age)
    FROM people AS p)
GROUP BY p.age;
```

Result:

```
+-----+------------+
| AGE | COUNT(AGE) |
+-----+------------+
| 41  | 19         |
| 42  | 19         |
| 43  | 22         |
| 44  | 16         |
| 45  | 19         |
| 46  | 17         |
| 47  | 23         |
| 48  | 23         |
| 49  | 14         |
| 50  | 12         |
| 51  | 17         |
| 52  | 26         |
| 53  | 21         |
| 54  | 21         |
| 55  | 10         |
| 56  | 26         |
| 57  | 20         |
| 58  | 26         |
| 59  | 22         |
| 60  | 22         |
| 61  | 23         |
| 62  | 21         |
| 63  | 18         |
| 64  | 21         |
| 65  | 17         |
+-----+------------+
```

Modified

Modified comparison operators are logical and are used with one of the following modifiers:

- `ANY` or `SOME` — the operator will return `true` if at least one of the subquery results matches the comparison condition, and `false` otherwise;
- `ALL` — the operator will return `true` if all of the subquery results match the comparison condition, and `false` otherwise.

The subquery must return values from a single column.

For example, to list the american states where there are enterprises located along with their capitols, you need the following query:

```
SELECT s.name, s.capitol
FROM states AS s
WHERE s.abbr = ANY
    (SELECT e.state
    FROM enterprises AS e
    JOIN
    states AS s
    ON s.abbr=e.state);
```

Result:

```
+------------+-------------+
|    NAME    |   CAPITOL   |
+------------+-------------+
| Arizona    | Phoenix     |
| California | Sacramento  |
| Louisiana  | Baton Rouge |
| Maryland   | Annapolis   |
| New Mexico | Santa Fe    |
| New York   | Albany      |
| Texas      | Austin      |
| Washington | Olympia     |
+------------+-------------+
```

To list the people whose age is not found among the residents of Nevada, along with their state and age, you need the following query:

```
SELECT p.name, p.state, p.age
FROM people AS p
WHERE p.age <> ALL
    (SELECT p.age
    FROM people AS p
    WHERE p.state='NV')
LIMIT 20;
```

Result:

```
+--------------------+-----------------+-----+
|        NAME        |      STATE      | AGE |
+--------------------+-----------------+-----+
| Abbott Jack        | Chihuahua       | 29  |
| Adams Clyde        | LA              | 29  |
| Alvarado Dominic   | California Baja | 63  |
| Alvarez Jacob      | Yucatan         | 49  |
| Anderson Lucile    | Sonora          | 33  |
| Andrews Lucy       | AZ              | 33  |
| Andrews Noah       | California Baja | 63  |
| Armstrong Isabella | Sonora          | 49  |
| Atkins Gene        | TX              | 37  |
| Bailey Richard     | Yucatan         | 39  |
| Benson Hallie      | California Baja | 37  |
| Bowers Norman      | Chihuahua       | 29  |
| Boyd Travis        | Yucatan         | 49  |
| Brock Carlos       | Quintana Roo    | 49  |
| Burgess Nell       | TX              | 33  |
| Byrd Nell          | Quintana Roo    | 39  |
| Castillo Edna      | Quintana Roo    | 63  |
| Chapman Lilly      | Sonora          | 63  |
| Christensen Gavin  | Quintana Roo    | 63  |
| Clark Joshua       | Sonora          | 63  |
+--------------------+-----------------+-----+
```

Correlated subqueries

A subquery is called **correlated** or **synchronized** when it references the same tables the outer query does. A correlated subquery is executed each time the outer query processes a next row according to its own clauses. This distinguishes the correlated subquery from an independent one, which is unrelated to the outer query and is only executed once.

For example, to list the american states where there are enterprises located in their largest cities, along with their capitols, you need the following query:

```
SELECT s.name, s.capitol
FROM states AS s
WHERE s.abbr IN
    (SELECT e.state
    FROM enterprises AS e
    WHERE e.town=s.largest);
```

Result:

```
+------------+------------+
|   S.NAME   | S.CAPITOL  |
+------------+------------+
| Arizona    | Phoenix    |
| California | Sacramento |
| New Mexico | Santa Fe   |
| New York   | Albany     |
+------------+------------+
```

Branches and nesting

You can unite subqueries using the `AND` and `OR` operators. Also, you can enclose the subqueries in one another however deep you want.

AND and OR branches

The `AND` operator is logical and means both operands of the expression need to be true for the whole expression to be true. The `OR` operator is logical and means any of the operands of the expression needs to be true for the whole expression to be true.

For example, to list the american people whose age is 65, and people anywhere whose age is 18, you need the following query:

```
SELECT p.name, p.age, p.state
FROM people AS p
WHERE p.state IN
    (SELECT s.abbr
    FROM states AS s)
AND p.age>64
OR p.age IN
    (SELECT p.age
    FROM people AS p
    WHERE p.age=18);
```

Result:

```
+--------------------+-------+-----------------+
|       P.NAME       | P.AGE |     P.STATE     |
+--------------------+-------+-----------------+
| Aguilar Terry      | 65    | CA              |
| Arnold Bettie      | 18    | NV              |
| Bailey Cameron     | 18    | TX              |
| Banks Catherine    | 18    | Sonora          |
| Campbell Todd      | 18    | NV              |
| Casey Louise       | 18    | California Baja |
| Davis Alan         | 18    | CA              |
| Farmer Dean        | 65    | LA              |
| Fitzgerald Timothy | 18    | Chihuahua       |
| Griffin Alexander  | 65    | NV              |
| Harper Alvin       | 65    | AZ              |
| Hill Katharine     | 18    | Quintana Roo    |
| Hodges Abbie       | 18    | Yucatan         |
| James Loretta      | 18    | TX              |
| Lopez Katharine    | 65    | AZ              |
| Love Dylan         | 65    | TX              |
| McCarthy Eunice    | 65    | LA              |
| Miller Francis     | 18    | Sonora          |
| Munoz Ricardo      | 18    | CA              |
| Myers Ricky        | 18    | LA              |
| Norton Bertha      | 18    | TX              |
| Phelps Rena        | 18    | Quintana Roo    |
| Riley Mollie       | 65    | OR              |
| Stokes Mittie      | 18    | OR              |
| Strickland Helena  | 65    | TX              |
| Thompson Warren    | 18    | AZ              |
| Torres Elsie       | 18    | LA              |
| Vaughn Belle       | 18    | CA              |
+--------------------+-------+-----------------+
```

Nesting

You can use subqueries inside subqueries. This is called **nesting**.

For example, to list the enterprises in the states that are listed from `A` to `C` for the people from the `people` table, you need the following query:

```
SELECT e.name
FROM enterprises AS e
WHERE e.state IN
    (SELECT p.state
    FROM people AS p
    WHERE p.state IN
        (SELECT s.abbr
        FROM states AS s
        WHERE s.name < 'D'));
```

Result:

```
+----------+
|  E.NAME  |
+----------+
| Apple    |
| Avnet    |
| Intel    |
| Obsidian |
| Sierra   |
+----------+
```

Derived tables

You can specify a subquery in the `FROM` clause. The result of such subquery will make up a table which can be searched for values.

For example, to list the people outside the USA along with their hometowns and ages, who are also older than 63, you need the following query:

```
SELECT pp.name, pp.town, pp.age
FROM
    (SELECT p.name, p.town, p.age
    FROM people AS p
    WHERE p.country != 'USA')
AS pp
WHERE pp.age>63;
```

Result:

```
+------------------+-----------+-----+
|       NAME       |   TOWN    | AGE |
+------------------+-----------+-----+
| Bryant Addie     | Izamal    | 64  |
| Chavez Don       | Chetumal  | 65  |
| Cohen Victor     | Chetumal  | 64  |
| Cox Lou          | Tijuana   | 64  |
| Craig Bruce      | Chihuahua | 65  |
| Elliott Callie   | Juarez    | 65  |
| Flores Lottie    | Uman      | 64  |
| Flores Nannie    | Chihuahua | 64  |
| Frank Chester    | Nogales   | 65  |
| Gill Bessie      | Nogales   | 64  |
| Hunter Curtis    | Mexicali  | 65  |
| Keller Loretta   | Tijuana   | 65  |
| Little Antonio   | Juarez    | 64  |
| Massey Hettie    | Tijuana   | 65  |
| McGee Lulu       | Cancun    | 64  |
| Mitchell Gary    | Uman      | 64  |
| Mitchell Jason   | Ensenada  | 65  |
| Riley Terry      | Juarez    | 64  |
| Torres Katharine | Juarez    | 64  |
| Wheeler Oscar    | Merida    | 64  |
| Wise Andre       | Cancun    | 64  |
+------------------+-----------+-----+
```

Limitations

There are two limitations to the Phoenix usage of subqueries as of now:

- You cannot use a subquery as an item in a `SELECT` clause.
- Subqueries cannot have a `HAVING` clause.

Sequences in Phoenix

Vladimir Adamenkov

Contents

To Table of Contents

- Overview
- Create a sequence
- Delete a sequence
- Generate new values
  - UPSERT VALUES statement
  - SELECT statement
- View the last value
- View sequence details
- Sequence specifics

Overview

Sequence is a tool that consecutively generates a monotonically increasing or decreasing row of integers according to specified conditions. Such rows are mostly used to create IDs, but can also be used for other purposes. Sequences may or may not belong to a schema.

Create a sequence

To create a sequence, use the following syntax:

```
CREATE SEQUENCE [IF NOT EXISTS] {[<schema_name>.]<sequence_name>}
[START [WITH] <number>]
[INCREMENT [BY] <number>]
[MINVALUE <number>]
[MAXVALUE <number>]
[CYCLE]
[CACHE <number>];
```

Parameters and keywords of the CREATE SEQUENCE command

Parameter or keyword	Description
IF NOT EXISTS	Without this keyword, attempting to create a sequence with the name of an already existing sequence will result in an error message. Not used by default
schema_name	Name of the schema the sequence belongs to
sequence_name	Name of the sequence
START (WITH)	The number that the sequence begins with. It is returned the first time the sequence is called to generate a new value. This value must be within the range specified by the values of the `MINVALUE` and `MAXVALUE` keywords, ends included. The `WITH` keyword is optional (does not affect anything). Default: `1`
number	Numerical value required by the `START WITH`, `INCREMENT BY`, `MINVALUE`, `MAXVALUE`, and `CACHE` keywords
INCREMENT (BY)	Value of the sequence increment or decrement size. It is added to the current sequence value to obtain the next value. The `BY` keyword is optional (does not affect anything). Default: `1`
MINVALUE	Minimum value of the sequence. Used as a stopping point for descending sequences or as a starting point for ascending sequences that use cycling after reaching the maximum value. Default: `-9223372036854775808`
MAXVALUE	Maximum value of the sequence. Used as a stopping point for ascending sequences or as a starting point for descending sequences that use cycling after reaching the minimum value. Default: `9223372036854775807`
CYCLE	Defines whether to restart the sequence from the minimum value (or from the maximum value if the sequence is descending) when the other end value is exceeded or to throw an exception. Not used by default
CACHE	The number of consecutive values to store on the client side that must be exhausted before another batch can be requested from the server. Multiple clients can use the same sequence. Any time any client requests a new batch, it will obtain the next one. Default: `100`. The default value can be changed manually by adding the `phoenix.sequence.cacheSize` parameter to the ***Custom hbase-site.xml*** section of the HBase service configuration page

NOTE

Sequences in Phoenix can only use the `BIGINT` type for returned values (ranging from `-9223372036854775808` to `9223372036854775807`).

Example:

```
CREATE SEQUENCE IF NOT EXISTS test_schema.test_sequence
START WITH 97 INCREMENT BY 2 MINVALUE 0 MAXVALUE 100 CYCLE CACHE 50;
```

This command creates a sequence that returns the first value of `97`, then the value of `99`, and then will cyclically return even numbers from `0` to `100` in ascending order. The first 50 values will be stored on the client side and returned consecutively during executions of queries before obtaining another 50 values from the server.

Delete a sequence

To delete a sequence, use the following syntax:

```
DROP SEQUENCE [IF EXISTS] <sequence_ref>;
```

where `<sequence_ref>` is the exact name of the sequence as it was specified when the sequence was originally created. If a sequence belongs to a schema, then it needs to be specified in this command as well.

Example:

```
DROP SEQUENCE IF EXISTS test_schema.test_sequence;
```

Generate new values

To make a sequence generate a new value, use the `NEXT VALUE FOR` expression followed by the sequence name in a `SELECT` or an `UPSERT VALUES` statement. A sequence that is called the first time generates the starting value (default or predefined one).

UPSERT VALUES statement

For example, to create a table and fill it with values while using a sequence to generate IDs for the primary key, execute the following commands:

```
CREATE TABLE weapon_types (
id INTEGER NOT NULL PRIMARY KEY,
wpn_type VARCHAR(50),
ammo_type VARCHAR(50));

CREATE SEQUENCE IF NOT EXISTS test_schema.test_sequence START 1 INCREMENT 1;

UPSERT INTO weapon_types(id, wpn_type, ammo_type) VALUES (NEXT VALUE FOR test_schema.test_sequence, 'handgun', 'bullet');
UPSERT INTO weapon_types(id, wpn_type, ammo_type) VALUES (NEXT VALUE FOR test_schema.test_sequence, 'shotgun', 'buckshot');
UPSERT INTO weapon_types(id, wpn_type, ammo_type) VALUES (NEXT VALUE FOR test_schema.test_sequence, 'shotgun', 'slug');
UPSERT INTO weapon_types(id, wpn_type, ammo_type) VALUES (NEXT VALUE FOR test_schema.test_sequence, 'assault rifle', 'bullet');
UPSERT INTO weapon_types(id, wpn_type, ammo_type) VALUES (NEXT VALUE FOR test_schema.test_sequence, 'sniper rifle', 'bullet');
UPSERT INTO weapon_types(id, wpn_type, ammo_type) VALUES (NEXT VALUE FOR test_schema.test_sequence, 'crossbow', 'arrow');

SELECT * FROM weapon_types;
```

Result:

```
+----+---------------+-----------+
| ID |   WPN_TYPE    | AMMO_TYPE |
+----+---------------+-----------+
| 1  | handgun       | bullet    |
| 2  | shotgun       | buckshot  |
| 3  | shotgun       | slug      |
| 4  | assault rifle | bullet    |
| 5  | sniper rifle  | bullet    |
| 6  | crossbow      | arrow     |
+----+---------------+-----------+
```

SELECT statement

You can use the `SELECT` statement to load values into a table from another table using a single `UPSERT` statement. For example, to create a table enumerating the ammo types from the previously created table while using a sequence to generate IDs for the primary key, execute the following commands:

```
CREATE TABLE ammo_types (
id INTEGER NOT NULL PRIMARY KEY,
ammo_type VARCHAR(50));

CREATE SEQUENCE IF NOT EXISTS test_schema.test_sequence2 START 1 INCREMENT 1;

UPSERT INTO ammo_types(id, ammo_type)
SELECT NEXT VALUE FOR test_schema.test_sequence2, ammo_type
FROM weapon_types
GROUP BY ammo_type;

SELECT * FROM ammo_types;
```

Result:

```
+----+-----------+
| ID | AMMO_TYPE |
+----+-----------+
| 1  | arrow     |
| 2  | buckshot  |
| 3  | bullet    |
| 4  | slug      |
+----+-----------+
```

View the last value

To view the last value generated by a sequence, use the `CURRENT VALUE FOR` expression followed by the sequence name in a `SELECT` statement. Example:

```
SELECT CURRENT VALUE FOR test_schema.test_sequence;
```

Result:

```
+---------------------------------------------+
| CURRENT VALUE FOR TEST_SCHEMA.TEST_SEQUENCE |
+---------------------------------------------+
| 4                                           |
+---------------------------------------------+
```

NOTE

If a sequence has never been called to generate a new value, this command will return an error.

If a `CURRENT VALUE FOR` expression is used together with a `NEXT VALUE FOR` expression for the same sequence in a single statement, they will both return the same newly generated value.

View sequence details

To view the details of all sequences, use the following command:

```
SELECT * FROM SYSTEM."SEQUENCE";
```

Result:

```
+-----------+-----------------+---------------+------------+---------------+--------------+------------+----------------------+--
------------------+------------+--------------------+
| TENANT_ID | SEQUENCE_SCHEMA | SEQUENCE_NAME | START_WITH | CURRENT_VALUE | INCREMENT_BY | CACHE_SIZE |      MIN_VALUE       |
MAX_VALUE       | CYCLE_FLAG | LIMIT_REACHED_FLAG |
+-----------+-----------------+---------------+------------+---------------+--------------+------------+----------------------+--
------------------+------------+--------------------+
|           | TEST_SCHEMA     | TEST_SEQUENCE | 1          | 101           | 1            | 100        | -9223372036854775808 |
9223372036854775807 | false      | false              |
+-----------+-----------------+---------------+------------+---------------+--------------+------------+----------------------+--
------------------+------------+--------------------+
```

Notice that the `CURRENT_VALUE` column value for the sequence is `101`. This is because the cache size is 100, and the first 100 values were already sent to the client. If you create another connection session to the Phoenix Query Server, it will be treated as another client. In this case, if you generate the next value for the same sequence in that second session, a value of `101` will be returned, and the `CURRENT_VALUE` column value will change to `201`. When a client exhausts its cache of values, it will obtain the next batch from the server. The starting value of this batch will depend on how many batches were given to other clients before that.

The `SYSTEM.SEQUENCE` table is read-only.

Sequence specifics

When using sequences, consider the following:

- You cannot alter a sequence after it is created.
- Both `NEXT VALUE FOR` and `CURRENT VALUE FOR` expressions can only be used in the `SELECT` and `UPSERT VALUES` statements. Using these expressions in the `WHERE`, `GROUP BY`, `HAVING`, or `ORDER BY` statements will result in an error.
- The `NEXT VALUE FOR` expression generates a new value only once per a single `SELECT` or `UPSERT` statement. For example, you may use the statement of the following kind:

```
UPSERT INTO test_table(c1,c2) VALUES (NEXT VALUE FOR some_sequence, NEXT VALUE FOR some_sequence);
```

In this case, the values in both columns will be the same.

Use salted tables in Phoenix

Vladimir Adamenkov

Contents

To Table of Contents

Overview
Usage
Considerations

Overview

HBase distributes its data among regions hosted by region servers. Regions are determined by row key ranges. In some cases, regions that contain data belonging to a small range of row key space may experience extra high load (write and/or read), while other regions may remain relatively idle. This usually happens when row keys represent a monotonous sequence of some kind (integer IDs, timestamps) and is called **hotspotting**. It leads to lower performance and unnecessary region splits.

To avoid such situations, **salting** is used in Apache Phoenix. Salting means adding a prefix to row keys which is called **salt** — a single byte in case of Apache Phoenix. It is done so that adjacent keys would have much higher chance of going to different regions. This allows for more even data distribution and more parallel writes. Read performance can also improve, but not always (see Considerations).

Usage

To create a salted table, you need to specify a number of row key prefixes (**salt buckets**) to use throughout the table, for example:

```
CREATE TABLE people (
name VARCHAR(50) NOT NULL PRIMARY KEY,
age SMALLINT,
town VARCHAR(50),
state VARCHAR(50),
country VARCHAR(50))
SALT_BUCKETS = 10;
```

This command creates an empty table with 10 salt buckets. Now you can load the data into the table. To be able to follow examples given in this article, you need to import the data from the people.csv sample file as described in the JOIN clause in Phoenix article. Use the table created in the example above.

To see how salting affects the row keys, execute the following query:

```
SELECT * FROM people
WHERE people.name > 'X';
```

This query should return all people whose names begin with `Y` or `Z`. Result:

```
+-----------------+-----+-------------+-----------------+---------+
|      NAME       | AGE |    TOWN     |      STATE      | COUNTRY |
+-----------------+-----+-------------+-----------------+---------+
| Young Della     | 21  | Chetumal    | Quintana Roo    | MEX     |
| Yates Bill      | 48  | Juarez      | Chihuahua       | MEX     |
| Yates Francis   | 28  | Juarez      | Chihuahua       | MEX     |
| Yates Justin    | 47  | New Orleans | LA              | USA     |
| Yates Danny     | 47  | Merida      | Yucatan         | MEX     |
| Yates Douglas   | 35  | Phoenix     | AZ              | USA     |
| Zimmerman Madge | 46  | Nogales     | AZ              | USA     |
| Young Mattie    | 39  | San Jose    | CA              | USA     |
| Yates George    | 39  | Chihuahua   | Chihuahua       | MEX     |
| Yates Lucile    | 25  | Chihuahua   | Chihuahua       | MEX     |
| Zimmerman Gene  | 35  | Tijuana     | California Baja | MEX     |
| Yates Johnny    | 20  | Phoenix     | AZ              | USA     |
| Young Catherine | 60  | Juarez      | Chihuahua       | MEX     |
| Young Josephine | 29  | Redding     | OR              | USA     |
+-----------------+-----+-------------+-----------------+---------+
14 rows selected (0.019 seconds)
```

As you can see, the order of names is not exactly alphabetical, although HBase is supposed to sort the row keys. This happened because the names have prefixes which are not displayed, and the sorting is done according to those prefixes. You can still get the alphabetic order for names by explicitly sorting the column values. Example:

```
SELECT * FROM people
WHERE people.name > 'X'
ORDER BY people.name;
```

Result:

```
+-----------------+-----+-------------+-----------------+---------+
|      NAME       | AGE |    TOWN     |      STATE      | COUNTRY |
+-----------------+-----+-------------+-----------------+---------+
| Yates Bill      | 48  | Juarez      | Chihuahua       | MEX     |
| Yates Danny     | 47  | Merida      | Yucatan         | MEX     |
| Yates Douglas   | 35  | Phoenix     | AZ              | USA     |
| Yates Francis   | 28  | Juarez      | Chihuahua       | MEX     |
| Yates George    | 39  | Chihuahua   | Chihuahua       | MEX     |
| Yates Johnny    | 20  | Phoenix     | AZ              | USA     |
| Yates Justin    | 47  | New Orleans | LA              | USA     |
| Yates Lucile    | 25  | Chihuahua   | Chihuahua       | MEX     |
| Young Catherine | 60  | Juarez      | Chihuahua       | MEX     |
| Young Della     | 21  | Chetumal    | Quintana Roo    | MEX     |
| Young Josephine | 29  | Redding     | OR              | USA     |
| Young Mattie    | 39  | San Jose    | CA              | USA     |
| Zimmerman Gene  | 35  | Tijuana     | California Baja | MEX     |
| Zimmerman Madge | 46  | Nogales     | AZ              | USA     |
+-----------------+-----+-------------+-----------------+---------+
14 rows selected (0.024 seconds)
```

In this case, when the `ORDER BY` clause is used for the row key column, the salt is ignored.

To display the list of salted tables, execute the following query:

```
SELECT table_name, salt_buckets
FROM SYSTEM.CATALOG
WHERE salt_buckets IS NOT NULL;
```

Result example:

```
+---------------+--------------+
|  TABLE_NAME   | SALT_BUCKETS |
+---------------+--------------+
| PEOPLE        | 10           |
| PEOPLE_SALTED | 10           |
| LOG           | 32           |
+---------------+--------------+
```

Considerations

- The number of buckets (the range of possible salt values) should be chosen based on the size of your cluster and the expected write volume. A good starting point is to have at least one bucket per region server. Experimentation is key to finding the optimal value.
- While salting generally improves write performance, it can potentially decrease read performance if your read queries rely on the original row key order. Phoenix will automatically scan across all salt buckets, which can be less efficient than a targeted region scan.
- Salting introduces complexity. You need to understand your data distribution and access patterns to determine if the benefits outweigh the costs.
- You might need to adjust your queries to account for the salted row keys. For example, you will probably have to use the `LIKE` operator along with wildcards on the salt portion of the key.
- Salting can break data locality. If you frequently need to access related data based on the original row key, salting can force you to scan multiple regions.

Map with existing HBase tables

Daria Barysheva

To Table of Contents

The section Use SQLLine to work with Phoenix describes how to create new tables in HBase via Phoenix. But you can also use Phoenix to work with existing HBase tables in a reading mode. Views are designed for this purpose. When creating them, the original HBase metadata is left AS-IS.

Let’s consider an example with the `people_ages` table. Use the HBase shell to check if the table exists:

```
list
```

The output is similar to the following:

```
TABLE
BEST_BOOKS
SYSTEM.CATALOG
SYSTEM.FUNCTION
SYSTEM.LOG
SYSTEM.MUTEX
SYSTEM.SEQUENCE
SYSTEM.STATS
ns1:temp2
people_ages
t4
t5
t6
table10
temp1
temp10
wordcount
wordcount2
wordcount3
wordcount4
19 row(s)
Took 0.0428 seconds
=> ["BEST_BOOKS", "SYSTEM.CATALOG", "SYSTEM.FUNCTION", "SYSTEM.LOG", "SYSTEM.MUTEX", "SYSTEM.SEQUENCE", "SYSTEM.STATS",
"ns1:temp2", "people_ages", "t4", "t5", "t6", "table10", "temp1", "temp10", "wordcount", "wordcount2", "wordcount3",
"wordcount4"]
```

This table is not available in Phoenix yet. Check it using the appropriate SQLLine command:

```
!tables
```

The output does not contain `people_ages`:

```
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------+
| TABLE_CAT  | TABLE_SCHEM  | TABLE_NAME  |  TABLE_TYPE   | REMARKS  | TYPE_NAME  | SELF_REFERENCING_COL_NAME  | REF_GENERATION
| INDEX_STATE  | IMMUTABLE |
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------+
|            | SYSTEM       | CATALOG     | SYSTEM TABLE  |          |            |                            |
|              | false     |
|            | SYSTEM       | FUNCTION    | SYSTEM TABLE  |          |            |                            |
|              | false     |
|            | SYSTEM       | LOG         | SYSTEM TABLE  |          |            |                            |
|              | true      |
|            | SYSTEM       | SEQUENCE    | SYSTEM TABLE  |          |            |                            |
|              | false     |
|            | SYSTEM       | STATS       | SYSTEM TABLE  |          |            |                            |
|              | false     |
|            |              | BEST_BOOKS  | TABLE         |          |            |                            |
|              | false     |
+------------+--------------+-------------+---------------+----------+------------+----------------------------+-----------------
+--------------+-----------+
```

To create a new view in Phoenix, run the following command using SQLLine:

```
CREATE VIEW "people_ages" ("pk" VARCHAR PRIMARY KEY, "basic"."age" UNSIGNED_SMALLINT);
```

Now, you can select data in Phoenix from the original table in HBase:

```
SELECT * FROM "people_ages" LIMIT 10;
```

The output:

```
+--------------------+--------+
|         pk         |  age   |
+--------------------+--------+
| Abbott Delia       | 13874  |
| Abbott Howard      | 12852  |
| Abbott Jack        | 12857  |
| Adams Clyde        | 12857  |
| Aguilar Myrtie     | 12851  |
| Aguilar Terry      | 13877  |
| Alexander Derrick  | 13366  |
| Alexander Gregory  | 13620  |
| Alexander Leon     | 13362  |
| Allen Austin       | 13108  |
+--------------------+--------+
10 rows selected (0.029 seconds)
```

Get the number of rows:

```
SELECT COUNT(*) FROM "people_ages";
```

The output:

```
+-----------+
| COUNT(1)  |
+-----------+
| 997       |
+-----------+
1 row selected (0.039 seconds)
```

If you get the list of Phoenix tables again, you can see the added view `people_ages`. Later you can delete it, if necessary, using the DROP VIEW command.

```
!tables
```

The output:

```
+------------+--------------+--------------+---------------+----------+------------+----------------------------+----------------
-+--------------+----------+
| TABLE_CAT  | TABLE_SCHEM  |  TABLE_NAME  |  TABLE_TYPE   | REMARKS  | TYPE_NAME  | SELF_REFERENCING_COL_NAME  | REF_GENERATION
| INDEX_STATE  | IMMUTABL |
+------------+--------------+--------------+---------------+----------+------------+----------------------------+----------------
-+--------------+----------+
|            | SYSTEM       | CATALOG      | SYSTEM TABLE  |          |            |                            |
|              | false    |
|            | SYSTEM       | FUNCTION     | SYSTEM TABLE  |          |            |                            |
|              | false    |
|            | SYSTEM       | LOG          | SYSTEM TABLE  |          |            |                            |
|              | true     |
|            | SYSTEM       | SEQUENCE     | SYSTEM TABLE  |          |            |                            |
|              | false    |
|            | SYSTEM       | STATS        | SYSTEM TABLE  |          |            |                            |
|              | false    |
|            |              | BEST_BOOKS   | TABLE         |          |            |                            |
|              | false    |
|            |              | people_ages  | VIEW          |          |            |                            |
|              | false    |
+------------+--------------+--------------+---------------+----------+------------+----------------------------+----------------
-+--------------+----------+
```

Notice that Phoenix views are designed only for reading. If you try to modify data in a view, you will get an error message similar to the following.

The error message

```
0: jdbc:phoenix:> UPSERT INTO "people_ages" values('Test Test', 100);
Error: ERROR 505 (42000): Table is read only. (state=42000,code=505)
org.apache.phoenix.schema.ReadOnlyTableException: ERROR 505 (42000): Table is read only.
        at org.apache.phoenix.compile.UpsertCompiler.compile(UpsertCompiler.java:359)
        at org.apache.phoenix.jdbc.PhoenixStatement$ExecutableUpsertStatement.compilePlan(PhoenixStatement.java:784)
        at org.apache.phoenix.jdbc.PhoenixStatement$ExecutableUpsertStatement.compilePlan(PhoenixStatement.java:770)
        at org.apache.phoenix.jdbc.PhoenixStatement$2.call(PhoenixStatement.java:401)
        at org.apache.phoenix.jdbc.PhoenixStatement$2.call(PhoenixStatement.java:391)
        at org.apache.phoenix.call.CallRunner.run(CallRunner.java:53)
        at org.apache.phoenix.jdbc.PhoenixStatement.executeMutation(PhoenixStatement.java:390)
        at org.apache.phoenix.jdbc.PhoenixStatement.executeMutation(PhoenixStatement.java:378)
        at org.apache.phoenix.jdbc.PhoenixStatement.execute(PhoenixStatement.java:1825)
        at sqlline.Commands.execute(Commands.java:822)
        at sqlline.Commands.sql(Commands.java:732)
        at sqlline.SqlLine.dispatch(SqlLine.java:813)
        at sqlline.SqlLine.begin(SqlLine.java:686)
        at sqlline.SqlLine.start(SqlLine.java:398)
        at sqlline.SqlLine.main(SqlLine.java:291)
```

Bulk loading in Phoenix

Daria Barysheva

Contents

To Table of Contents

Method 1. Use psql

Method 2. Use MapReduce

Bulk data loading is available not only in HBase, but in Phoenix as well. There are two basic methods:

- **Using psql**. Psql is a single-threaded tool well suited for downloading several megabytes or gigabytes of data. It is based on the typical write sequence in HBase, without direct writing to HFiles.
  All uploaded files should have the *.sql* or *.csv* extension. The first group of files is used for SQL queries, the second one — for loading data.
- **Using MapReduce**. This method is based on ETL processing (Extract, Transform, Load). Files are uploaded to HDFS and then transformed by the built-in MapReduce job CsvBulkLoadTool. The transformed data is loaded directly into HFiles in HBase by the same job. This method can be used for significantly larger amounts of data in comparison with psql because MapReduce uses a multi-threaded process. So, using MapReduce is recommended for production needs.
  The supported file extensions are *.csv* and *.json*.

NOTE

You can find more information about bulk data loading in Phoenix documentation.

Method 1. Use psql

1. Prepare the *best_books.sql* SQL file for creating a new table `BEST_BOOKS` with the following structure:

```
CREATE TABLE IF NOT EXISTS best_books
        (author VARCHAR not null,
        title VARCHAR not null,
        public_year UNSIGNED_SMALLINT,
        CONSTRAINT pk PRIMARY KEY (author, title));
```

This table will contain information about famous books, including their authors, titles, and dates of the first publication. You can download the source file best_books.sql.

CAUTION

For some reasons, psql does not work with lowercase table names. If you call the table `"best_books"`, it will lead to errors when loading data via psql, although when you do it manually, everything works perfectly. So, we recommend not to use double quotes in table names, allowing their converting to uppercase. This restriction does not apply to column names.

2. Prepare two CSV files for loading data into the `BEST_BOOKS` table: best_books1.csv and best_books2.csv.

3. Prepare the *best_books_query.sql* file for selecting data from the `BEST_BOOKS` table:

```
SELECT
        public_year as "Year",
        count(*) as "Books count"
FROM best_books
GROUP BY public_year
ORDER BY public_year;
```

This query will return the number of the books grouped by year of first publication. You can download the file best_books_query.sql.

4. Download all the files to the local file system of one of your HBase cluster hosts. Make sure that the files are uploaded successfully using the following command:

```
$ ls -la ~
```

The output looks similar to the following:

```
total 96
drwx------. 6 dasha dasha  4096 Dec  1 14:33 .
drwxr-xr-x. 3 root  root     19 Aug 31 11:54 ..
drwx------. 3 dasha dasha    17 Aug 31 15:06 .ansible
-rw-------. 1 dasha dasha 23386 Dec  1 12:45 .bash_history
-rw-r--r--. 1 dasha dasha    18 Apr  1  2020 .bash_logout
-rw-r--r--. 1 dasha dasha   193 Apr  1  2020 .bash_profile
-rw-r--r--. 1 dasha dasha   231 Apr  1  2020 .bashrc
-rw-rw-r--. 1 dasha dasha  2736 Dec  1 14:33 best_books1.csv
-rw-rw-r--. 1 dasha dasha  1140 Dec  1 14:33 best_books2.csv
-rw-rw-r--. 1 dasha dasha   122 Dec  1 14:33 best_books_query.sql
-rw-rw-r--. 1 dasha dasha   176 Dec  1 14:33 best_books.sql
drwxrwxrwx. 2 dasha dasha    64 Nov 26 07:45 dasha
-rw-rw-r--. 1 dasha dasha 17651 Nov 26 07:46 people_ages.csv
drwxrwxr-x. 2 dasha dasha    21 Nov 30 14:04 .sqlline
drwx------. 2 dasha dasha    29 Sep 23 17:35 .ssh
```

5. Change the directory:

```
$ cd /usr/lib/phoenix/bin
```

6. Use the following command to run psql. The `-t` argument means the table name. It should be explicitly specified in case of a mismatch with the CSV file name (which is used by default). The `-d` argument specifies the delimiter used in the CSV file (the default one is a comma `,`). Other possible arguments can be found in Phoenix documentation.
   You can define several files at once, but mind that they are passed one by one; so specify them in the right order. In our example, we firstly create a table, then load data into it, and then run a query:

```
$ ./psql.py -t BEST_BOOKS -d '|' ~/best_books.sql ~/best_books1.csv ~/best_books_query.sql
```

The output

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
21/12/01 16:13:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
no rows upserted
Time: 1.381 sec(s)

csv columns from database.
CSV Upsert complete. 70 rows upserted
Time: 0.068 sec(s)
                                    Year                              Books count
---------------------------------------- ----------------------------------------
                                    1924                                        1
                                    1925                                        2
                                    1926                                        1
                                    1927                                        3
                                    1929                                        2
                                    1932                                        1
                                    1934                                        1
                                    1936                                        1
                                    1937                                        1
                                    1939                                        2
                                    1940                                        3
                                    1945                                        2
                                    1946                                        1
                                    1947                                        1
                                    1949                                        1
                                    1950                                        1
                                    1951                                        1
                                    1952                                        1
                                    1953                                        1
                                    1954                                        2
                                    1955                                        2
                                    1957                                        1
                                    1958                                        1
                                    1959                                        1
                                    1960                                        2
                                    1961                                        4
                                    1962                                        3
                                    1963                                        1
                                    1965                                        1
                                    1966                                        2
                                    1969                                        4
                                    1970                                        2
                                    1973                                        1
                                    1975                                        1
                                    1981                                        1
                                    1984                                        1
                                    1985                                        2
                                    1987                                        2
                                    1990                                        1
                                    1992                                        1
                                    1996                                        1
                                    1997                                        1
                                    2000                                        2
                                    2001                                        2
                                    2005                                        1
Time: 0.027 sec(s)
```

So, you can see that the command does three things at the same time: creates a table, inserts data, and queries the results.

7. Check the results of the command. Run SQLLine and perform the following query to display the first ten records of the created table:

```
SELECT * FROM best_books LIMIT 10;
```

The output:

```
+-----------------------+------------------------------------------------------+--------------+
|        AUTHOR         |                        TITLE                         | PUBLIC_YEAR  |
+-----------------------+------------------------------------------------------+--------------+
| A.S. Byatt            | Possession                                           | 1990         |
| Alan Moore            | Watchmen                                             | 1987         |
| Anthony Burgess       | A Clockwork Orange                                   | 1962         |
| C.S. Lewis            | The Lion, the Witch and the Wardrobe                 | 1950         |
| Carson McCullers      | The Heart Is a Lonely Hunter                         | 1940         |
| Chinua Achebe         | Chinua Achebe                                        | 1958         |
| Cormac McCarthy       | Blood Meridian, or the Evening Redness in the West   | 1985         |
| Dashiell Hammett      | Red Harvest                                          | 1929         |
| David Foster Wallace  | Infinite Jest                                        | 1996         |
| Don DeLillo           | White Noise                                          | 1985         |
+-----------------------+------------------------------------------------------+--------------+
10 rows selected (0.059 seconds)
```

Use HBase shell to scan the created table. The first ten values are listed below. Pay attention that as we defined a complex primary key for this table, the two fields, which are included into it, are stored as a single row key in HBase (`AUTHOR` + `TITLE`):

```
scan 'BEST_BOOKS'
```

The output:

```
ROW                                                                                          COLUMN+CELL
 A.S. Byatt\x00Possession                                                                   column=0:\x00\x00\x00\x00,
timestamp=1638375201460, value=x
 A.S. Byatt\x00Possession                                                                   column=0:\x80\x0B,
timestamp=1638375201460, value=\x07\xC6
 Alan Moore\x00Watchmen                                                                     column=0:\x00\x00\x00\x00,
timestamp=1638375201460, value=x
 Alan Moore\x00Watchmen                                                                     column=0:\x80\x0B,
timestamp=1638375201460, value=\x07\xC3
 Anthony Burgess\x00A Clockwork Orange                                                      column=0:\x00\x00\x00\x00,
timestamp=1638375201460, value=x
 Anthony Burgess\x00A Clockwork Orange                                                      column=0:\x80\x0B,
timestamp=1638375201460, value=\x07\xAA
 C.S. Lewis\x00The Lion, the Witch and the Wardrobe                                         column=0:\x00\x00\x00\x00,
timestamp=1638375201460, value=x
 C.S. Lewis\x00The Lion, the Witch and the Wardrobe                                         column=0:\x80\x0B,
timestamp=1638375201460, value=\x07\x9E
 Carson McCullers\x00The Heart Is a Lonely Hunter                                           column=0:\x00\x00\x00\x00,
timestamp=1638375201460, value=x
 Carson McCullers\x00The Heart Is a Lonely Hunter                                           column=0:\x80\x0B,
timestamp=1638375201460, value=\x07\x94
```

8. After the table creation, you can run psql once more to upload the second data part. Notice that this command uses only two files (the first script is not required any more):

```
$ ./psql.py -t BEST_BOOKS -d '|' ~/best_books2.csv ~/best_books_query.sql
```

The output:

```
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
21/12/01 16:16:34 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
classes where applicable
csv columns from database.
CSV Upsert complete. 30 rows upserted
Time: 0.065 sec(s)

                                    Year                              Books count
---------------------------------------- ----------------------------------------
                                    1924                                        1
                                    1925                                        3
                                    1926                                        1
                                    1927                                        3
                                    1929                                        2
                                    1932                                        1
                                    1934                                        5
                                    1936                                        1
                                    1937                                        1
                                    1938                                        1
                                    1939                                        4
                                    1940                                        4
                                    1945                                        4
                                    1946                                        1
                                    1947                                        1
                                    1948                                        1
                                    1949                                        2
                                    1950                                        1
                                    1951                                        1
                                    1952                                        1
                                    1953                                        2
                                    1954                                        3
                                    1955                                        4
                                    1957                                        3
                                    1958                                        1
                                    1959                                        1
                                    1960                                        3
                                    1961                                        5
                                    1962                                        4
                                    1963                                        1
                                    1964                                        1
                                    1965                                        1
                                    1966                                        2
                                    1967                                        1
                                    1969                                        4
                                    1970                                        3
                                    1973                                        1
                                    1974                                        1
                                    1975                                        1
                                    1977                                        1
                                    1980                                        1
                                    1981                                        1
                                    1984                                        2
                                    1985                                        2
                                    1986                                        1
                                    1987                                        2
                                    1990                                        1
                                    1992                                        1
                                    1996                                        1
                                    1997                                        1
                                    2000                                        2
                                    2001                                        2
                                    2005                                        1
Time: 0.033 sec(s)
```

9. In conclusion, let's make sure that the data uploaded to HBase via psql is not stored in HFiles yet. To do this, run the `list_regions` command. The value of the `LOCALITY` column, equal to `0`, proves that:

```
list_regions 'BEST_BOOKS'
```

The output:

```
                                          SERVER_NAME |                                                          REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ----------------------------------------------------------- | ---------- |
---------- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1638426590048 | BEST_BOOKS,,1638430003108.36ee4f69dc1ab9d4a075091ff9ea3b80. |            |
|     0 |   140 |        0.0 |
 1 rows
Took 0.5920 seconds
```

Method 2. Use MapReduce

1. Using HBase shell, truncate the table content to make it empty again. Pay attention that the table should be created in Phoenix before using MapReduce.

```
truncate 'BEST_BOOKS'
```

The output looks like the following:

```
Truncating 'BEST_BOOKS' table (it may take a while):
Disabling table...
Truncating table...
Took 2.1019 seconds
```

NOTE

The steps 2 and 3 are necessary if you did not execute the example with bulk loading in HBase, as described in Bulk loading in HBase.

2. Create a directory for your user in HDFS if it does not exist yet:

```
$ sudo -u hdfs hdfs dfs -mkdir /user/dasha
$ hdfs dfs -ls /user
```

The output looks similar to the following:

```
Found 5 items
drwxr-xr-x   - hdfs   hadoop          0 2021-11-26 09:56 /user/dasha
drwx------   - hdfs   hadoop          0 2021-08-31 16:15 /user/hdfs
drwxr-xr-x   - mapred hadoop          0 2021-08-31 16:22 /user/history
drwxr-xr-x   - mapred mapred          0 2021-08-31 16:21 /user/mapred
drwxr-xr-x   - yarn   yarn            0 2021-09-01 06:57 /user/yarn
```

3. Expand access rights to your folder using the following command (see more information about file protection in Protect files in HDFS):

```
$ sudo -u hdfs hdfs dfs -chmod 777 /user/dasha
$ hdfs dfs -ls /user
```

The output looks similar to the following:

```
Found 5 items
drwxrwxrwx   - hdfs   hadoop          0 2021-11-26 09:57 /user/dasha
drwx------   - hdfs   hadoop          0 2021-08-31 16:15 /user/hdfs
drwxr-xr-x   - mapred hadoop          0 2021-08-31 16:22 /user/history
drwxr-xr-x   - mapred mapred          0 2021-08-31 16:21 /user/mapred
drwxr-xr-x   - yarn   yarn            0 2021-09-01 06:57 /user/yarn
```

4. Copy the *best_books1.csv* and *best_books2.csv* files from the local file system to HDFS:

```
$ hdfs dfs -copyFromLocal ~/best_books1.csv /user/dasha
$ hdfs dfs -copyFromLocal ~/best_books2.csv /user/dasha
```

5. Make sure the files are located in HDFS:

```
$ hdfs dfs -ls /user/dasha
```

The output:

```
Found 6 items
drwx------   - dasha hadoop          0 2021-11-26 09:59 /user/dasha/.staging
-rw-r--r--   3 dasha hadoop       2736 2021-12-02 06:55 /user/dasha/best_books1.csv
-rw-r--r--   3 dasha hadoop       1140 2021-12-02 06:55 /user/dasha/best_books2.csv
drwxrwxrwx   - dasha hadoop          0 2021-11-26 09:59 /user/dasha/hbase-staging
-rw-r--r--   3 dasha hadoop      17651 2021-11-26 09:57 /user/dasha/people_ages.csv
drwxr-xr-x   - hbase hbase           0 2021-11-26 09:59 /user/dasha/test_output
```

6. Use the following command for running the MapReduce job CsvBulkLoadTool. It will perform both data transformation and loading to HFiles.
   Enter `hadoop jar` and then define the full path to the *phoenix-<phoenix_version>-HBase-<hbase_version>client.jar* JAR file. Arguments are listed below:

Arguments

-t	The table name (mandatory)
-d	The delimiter used in the CSV file (optional). The default value is a comma `,`
-i	The path to the CSV file (mandatory)

Other possible arguments can be found in Phoenix documentation.

```
$ hadoop jar /usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-client.jar org.apache.phoenix.mapreduce.CsvBulkLoadTool -d '|' -t BEST_BOOKS -
i /user/dasha/best_books1.csv
```

The output:

```
2021-12-02 07:18:15,214 INFO util.QueryUtil: Creating connection with the jdbc url: jdbc:phoenix:localhost:2181:/hbase;
2021-12-02 07:18:15,543 INFO log.QueryLoggerDisruptor: Starting  QueryLoggerDisruptor for with ringbufferSize=8192,
waitStrategy=BlockingWaitStrategy, exceptionHandler=org.apache.phoenix.log.QueryLoggerDefaultExceptionHandler@1ee563dc...
2021-12-02 07:18:15,560 INFO query.ConnectionQueryServicesImpl: An instance of ConnectionQueryServices was created.
2021-12-02 07:18:15,690 INFO zookeeper.ReadOnlyZKClient: Connect 0x5ad1da90 to localhost:2181 with session timeout=90000ms,
retries 30, retry interval 1000ms, keepAlive=60000ms
2021-12-02 07:18:15,709 INFO zookeeper.ZooKeeper: Client environment:zookeeper.version=3.4.10-
39d3a4f269333c922ed3db283be479f9deacaa0f, built on 03/23/2017 10:13 GMT
2021-12-02 07:18:15,709 INFO zookeeper.ZooKeeper: Client environment:host.name=bds-adh-1.ru-central1.internal
2021-12-02 07:18:15,709 INFO zookeeper.ZooKeeper: Client environment:java.version=1.8.0_312
2021-12-02 07:18:15,709 INFO zookeeper.ZooKeeper: Client environment:java.vendor=Red Hat, Inc.
2021-12-02 07:18:15,709 INFO zookeeper.ZooKeeper: Client environment:java.home=/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.312.b07-1.
el7_9.x86_64/jre
2021-12-02 07:18:15,709 INFO zookeeper.ZooKeeper: Client environment:java.class.path=/etc/hadoop/conf:/usr/lib/hadoop/lib/jsch-
0.1.54.jar:/usr/lib/hadoop/lib/accessors-smart-1.2.jar:/usr/lib/hadoop/lib/stax2-api.jar:/usr/lib/hadoop/lib/asm-5.0.4.
jar:/usr/lib/hadoop/lib/jetty-io-9.3.24.v20180605.jar:/usr/lib/hadoop/lib/audience-annotations-0.5.0.
jar:/usr/lib/hadoop/lib/token-provider-1.0.1.jar:/usr/lib/hadoop/lib/avro-1.7.7.jar:/usr/lib/hadoop/lib/json-smart-2.3.
jar:/usr/lib/hadoop/lib/commons-beanutils-1.9.3.jar:/usr/lib/hadoop/lib/woodstox-core-5.0.3.jar:/usr/lib/hadoop/lib/commons-
cli-1.2.jar:/usr/lib/hadoop/lib/jsp-api-2.1.jar:/usr/lib/hadoop/lib/commons-codec-1.11.jar:/usr/lib/hadoop/lib/jetty-security-
9.3.24.v20180605.jar:/usr/lib/hadoop/lib/commons-collections-3.2.2.jar:/usr/lib/hadoop/lib/jsr305-3.0.0.
jar:/usr/lib/hadoop/lib/commons-compress-1.18.jar:/usr/lib/hadoop/lib/jetty-server-9.3.24.v20180605.
jar:/usr/lib/hadoop/lib/commons-configuration2-2.1.1.jar:/usr/lib/hadoop/lib/woodstox-core.jar:/usr/lib/hadoop/lib/commons-io-
2.5.jar:/usr/lib/hadoop/lib/jsr311-api-1.1.1.jar:/usr/lib/hadoop/lib/commons-lang-2.6.jar:/usr/lib/hadoop/lib/jul-to-slf4j-1.7.
25.jar:/usr/lib/hadoop/lib/commons-lang3-3.4.jar:/usr/lib/hadoop/lib/kerb-server-1.0.1.jar:/usr/lib/hadoop/lib/commons-logging-
1.1.3.jar:/usr/lib/hadoop/lib/kerb-simplekdc-1.0.1.jar:/usr/lib/hadoop/lib/commons-math3-3.1.1.
jar:/usr/lib/hadoop/lib/zookeeper-3.4.13.jar:/usr/lib/hadoop/lib/commons-net-3.6.jar:/usr/lib/hadoop/lib/kerby-config-1.0.1.
jar:/usr/lib/hadoop/lib/curator-client-2.13.0.jar:/usr/lib/hadoop/lib/jetty-servlet-9.3.24.v20180605.
jar:/usr/lib/hadoop/lib/curator-framework-2.13.0.jar:/usr/lib/hadoop/lib/kerby-asn1-1.0.1.jar:/usr/lib/hadoop/lib/curator-
recipes-2.13.0.jar:/usr/lib/hadoop/lib/gson-2.2.4.jar:/usr/lib/hadoop/lib/guava-11.0.2.jar:/usr/lib/hadoop/lib/jetty-util-9.3.
24.v20180605.jar:/usr/lib/hadoop/lib/htrace-core4-4.1.0-incubating.jar:/usr/lib/hadoop/lib/kerby-pkix-1.0.1.
jar:/usr/lib/hadoop/lib/httpclient-4.5.2.jar:/usr/lib/hadoop/lib/httpcore-4.4.4.jar:/usr/lib/hadoop/lib/jetty-webapp-9.3.24.
v20180605.jar:/usr/lib/hadoop/lib/jackson-annotations-2.7.8.jar:/usr/lib/hadoop/lib/kerby-util-1.0.1.
jar:/usr/lib/hadoop/lib/jackson-core-2.7.8.jar:/usr/lib/hadoop/lib/kerby-xdr-1.0.1.jar:/usr/lib/hadoop/lib/jackson-core-asl-1.9.
13.jar:/usr/lib/hadoop/lib/log4j-1.2.17.jar:/usr/lib/hadoop/lib/jackson-databind-2.7.8.jar:/usr/lib/hadoop/lib/metrics-core-3.2.
4.jar:/usr/lib/hadoop/lib/jackson-jaxrs-1.9.13.jar:/usr/lib/hadoop/lib/kerb-identity-1.0.1.jar:/usr/lib/hadoop/lib/jackson-
mapper-asl-1.9.13.jar:/usr/lib/hadoop/lib/nimbus-jose-jwt-4.41.1.jar:/usr/lib/hadoop/lib/jackson-xc-1.9.13.
jar:/usr/lib/hadoop/lib/javax.servlet-api-3.1.0.jar:/usr/lib/hadoop/lib/jaxb-api-2.2.11.jar:/usr/lib/hadoop/lib/netty-3.10.5.
Final.jar:/usr/lib/hadoop/lib/jaxb-impl-2.2.3-1.jar:/usr/lib/hadoop/lib/paranamer-2.3.jar:/usr/lib/hadoop/lib/jcip-annotations-
1.0-1.jar:/usr/lib/hadoop/lib/protobuf-java-2.5.0.jar:/usr/lib/hadoop/lib/jersey-core-1.19.jar:/usr/lib/hadoop/lib/re2j-1.1.
jar:/usr/lib/hadoop/lib/jersey-json-1.19.jar:/usr/lib/hadoop/lib/slf4j-api-1.7.25.jar:/usr/lib/hadoop/lib/jersey-server-1.19.
jar:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar:/usr/lib/hadoop/lib/jersey-servlet-1.19.jar:/usr/lib/hadoop/lib/jettison-1.1.
jar:/usr/lib/hadoop/lib/jetty-xml-9.3.24.v20180605.jar:/usr/lib/hadoop/lib/jetty-http-9.3.24.v20180605.
jar:/usr/lib/hadoop/lib/kerb-admin-1.0.1.jar:/usr/lib/hadoop/lib/snappy-java-1.0.5.jar:/usr/lib/hadoop/lib/kerb-client-1.0.1.
jar:/usr/lib/hadoop/lib/stax2-api-3.1.4.jar:/usr/lib/hadoop/lib/kerb-common-1.0.1.jar:/usr/lib/hadoop/lib/kerb-core-1.0.1.
jar:/usr/lib/hadoop/lib/kerb-crypto-1.0.1.jar:/usr/lib/hadoop/lib/kerb-util-1.0.1.jar:/usr/lib/hadoop/.//hadoop-annotations-3.1.
2.jar:/usr/lib/hadoop/.//hadoop-annotations.jar:/usr/lib/hadoop/.//hadoop-auth-3.1.2.jar:/usr/lib/hadoop/.//hadoop-auth.
jar:/usr/lib/hadoop/.//hadoop-common-3.1.2-tests.jar:/usr/lib/hadoop/.//hadoop-common-3.1.2.jar:/usr/lib/hadoop/.//hadoop-
common.jar:/usr/lib/hadoop/.//hadoop-kms-3.1.2.jar:/usr/lib/hadoop/.//hadoop-kms.jar:/usr/lib/hadoop/.//hadoop-nfs-3.1.2.
jar:/usr/lib/hadoop/.//hadoop-nfs.jar:/usr/lib/hadoop-hdfs/./:/usr/lib/hadoop-hdfs/lib/kerb-client-1.0.1.jar:/usr/lib/hadoop-
hdfs/lib/accessors-smart-1.2.jar:/usr/lib/hadoop-hdfs/lib/stax2-api-3.1.4.jar:/usr/lib/hadoop-hdfs/lib/asm-5.0.4.
jar:/usr/lib/hadoop-hdfs/lib/jetty-util-ajax-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/audience-annotations-0.5.0.
jar:/usr/lib/hadoop-hdfs/lib/token-provider-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/avro-1.7.7.jar:/usr/lib/hadoop-hdfs/lib/kerb-
common-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-beanutils-1.9.3.jar:/usr/lib/hadoop-hdfs/lib/woodstox-core-5.0.3.
jar:/usr/lib/hadoop-hdfs/lib/commons-cli-1.2.jar:/usr/lib/hadoop-hdfs/lib/kerb-core-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-
codec-1.11.jar:/usr/lib/hadoop-hdfs/lib/jetty-io-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/commons-collections-3.2.2.
jar:/usr/lib/hadoop-hdfs/lib/kerb-crypto-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-compress-1.18.jar:/usr/lib/hadoop-
hdfs/lib/jetty-xml-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/commons-configuration2-2.1.1.jar:/usr/lib/hadoop-
hdfs/lib/kerb-identity-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-daemon-1.0.13.jar:/usr/lib/hadoop-hdfs/lib/zookeeper-3.4.13.
jar:/usr/lib/hadoop-hdfs/lib/commons-io-2.5.jar:/usr/lib/hadoop-hdfs/lib/kerb-server-1.0.1.jar:/usr/lib/hadoop-
hdfs/lib/commons-lang-2.6.jar:/usr/lib/hadoop-hdfs/lib/kerb-simplekdc-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-lang3-3.4.
jar:/usr/lib/hadoop-hdfs/lib/kerb-util-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-logging-1.1.3.jar:/usr/lib/hadoop-
hdfs/lib/kerby-asn1-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/commons-math3-3.1.1.jar:/usr/lib/hadoop-hdfs/lib/commons-net-3.6.
jar:/usr/lib/hadoop-hdfs/lib/kerby-config-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/curator-client-2.13.0.jar:/usr/lib/hadoop-
hdfs/lib/jetty-webapp-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/curator-framework-2.13.0.jar:/usr/lib/hadoop-
hdfs/lib/kerby-pkix-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/curator-recipes-2.13.0.jar:/usr/lib/hadoop-hdfs/lib/gson-2.2.4.
jar:/usr/lib/hadoop-hdfs/lib/guava-11.0.2.jar:/usr/lib/hadoop-hdfs/lib/jsch-0.1.54.jar:/usr/lib/hadoop-hdfs/lib/htrace-core4-4.
1.0-incubating.jar:/usr/lib/hadoop-hdfs/lib/kerby-util-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/httpclient-4.5.2.jar:/usr/lib/hadoop-
hdfs/lib/httpcore-4.4.4.jar:/usr/lib/hadoop-hdfs/lib/json-simple-1.1.1.jar:/usr/lib/hadoop-hdfs/lib/jackson-annotations-2.7.8.
jar:/usr/lib/hadoop-hdfs/lib/kerby-xdr-1.0.1.jar:/usr/lib/hadoop-hdfs/lib/jackson-core-2.7.8.jar:/usr/lib/hadoop-
hdfs/lib/leveldbjni-all-1.8.jar:/usr/lib/hadoop-hdfs/lib/jackson-core-asl-1.9.13.jar:/usr/lib/hadoop-hdfs/lib/log4j-1.2.17.
jar:/usr/lib/hadoop-hdfs/lib/jackson-databind-2.7.8.jar:/usr/lib/hadoop-hdfs/lib/netty-3.10.5.Final.jar:/usr/lib/hadoop-
hdfs/lib/jackson-jaxrs-1.9.13.jar:/usr/lib/hadoop-hdfs/lib/json-smart-2.3.jar:/usr/lib/hadoop-hdfs/lib/jackson-mapper-asl-1.9.
13.jar:/usr/lib/hadoop-hdfs/lib/netty-all-4.0.52.Final.jar:/usr/lib/hadoop-hdfs/lib/jackson-xc-1.9.13.jar:/usr/lib/hadoop-
hdfs/lib/nimbus-jose-jwt-4.41.1.jar:/usr/lib/hadoop-hdfs/lib/javax.servlet-api-3.1.0.jar:/usr/lib/hadoop-hdfs/lib/jaxb-api-2.2.
11.jar:/usr/lib/hadoop-hdfs/lib/okhttp-2.7.5.jar:/usr/lib/hadoop-hdfs/lib/jaxb-impl-2.2.3-1.jar:/usr/lib/hadoop-hdfs/lib/okio-1.
6.0.jar:/usr/lib/hadoop-hdfs/lib/jcip-annotations-1.0-1.jar:/usr/lib/hadoop-hdfs/lib/paranamer-2.3.jar:/usr/lib/hadoop-
hdfs/lib/jersey-core-1.19.jar:/usr/lib/hadoop-hdfs/lib/protobuf-java-2.5.0.jar:/usr/lib/hadoop-hdfs/lib/jersey-json-1.19.
jar:/usr/lib/hadoop-hdfs/lib/re2j-1.1.jar:/usr/lib/hadoop-hdfs/lib/jersey-server-1.19.jar:/usr/lib/hadoop-hdfs/lib/snappy-java-
1.0.5.jar:/usr/lib/hadoop-hdfs/lib/jersey-servlet-1.19.jar:/usr/lib/hadoop-hdfs/lib/jettison-1.1.jar:/usr/lib/hadoop-
hdfs/lib/jsr305-3.0.0.jar:/usr/lib/hadoop-hdfs/lib/jetty-http-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/jetty-security-9.3.
24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/jetty-server-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/jsr311-api-1.1.1.
jar:/usr/lib/hadoop-hdfs/lib/jetty-servlet-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/lib/kerb-admin-1.0.1.jar:/usr/lib/hadoop-
hdfs/lib/jetty-util-9.3.24.v20180605.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-3.1.2-tests.jar:/usr/lib/hadoop-hdfs/.//hadoop-
hdfs-3.1.2.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-client-3.1.2-tests.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-client-3.1.2.
jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-client.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-httpfs-3.1.2.jar:/usr/lib/hadoop-
hdfs/.//hadoop-hdfs-httpfs.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-native-client-3.1.2-tests.jar:/usr/lib/hadoop-
hdfs/.//hadoop-hdfs-native-client-3.1.2.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-native-client.jar:/usr/lib/hadoop-
hdfs/.//hadoop-hdfs-nfs-3.1.2.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-nfs.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-rbf-3.1.2-
tests.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-rbf-3.1.2.jar:/usr/lib/hadoop-hdfs/.//hadoop-hdfs-rbf.jar:/usr/lib/hadoop-
hdfs/.//hadoop-hdfs.jar:/usr/lib/hadoop-mapreduce/.//hadoop-streaming.jar:/usr/lib/hadoop-mapreduce/.//aliyun-sdk-oss-2.8.3.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-examples.jar:/usr/lib/hadoop-mapreduce/.//aws-java-sdk-bundle-1.11.271.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-openstack-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//azure-data-lake-store-sdk-2.2.7.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-openstack.jar:/usr/lib/hadoop-mapreduce/.//azure-keyvault-core-1.0.0.
jar:/usr/lib/hadoop-mapreduce/.//jdom-1.1.jar:/usr/lib/hadoop-mapreduce/.//azure-storage-7.0.0.jar:/usr/lib/hadoop-
mapreduce/.//kafka-clients-0.8.2.1.jar:/usr/lib/hadoop-mapreduce/.//hadoop-aliyun-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-
aliyun.jar:/usr/lib/hadoop-mapreduce/.//hadoop-resourceestimator-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-archive-logs-3.1.
2.jar:/usr/lib/hadoop-mapreduce/.//lz4-1.2.0.jar:/usr/lib/hadoop-mapreduce/.//hadoop-archive-logs.jar:/usr/lib/hadoop-
mapreduce/.//netty-buffer-4.1.17.Final.jar:/usr/lib/hadoop-mapreduce/.//hadoop-archives-3.1.2.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-archives.jar:/usr/lib/hadoop-mapreduce/.//netty-codec-4.1.17.Final.jar:/usr/lib/hadoop-mapreduce/.//hadoop-
aws-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-aws.jar:/usr/lib/hadoop-mapreduce/.//netty-codec-http-4.1.17.Final.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-azure-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-resourceestimator.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-azure-datalake-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//netty-common-4.1.17.Final.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-azure-datalake.jar:/usr/lib/hadoop-mapreduce/.//hadoop-azure.jar:/usr/lib/hadoop-
mapreduce/.//netty-handler-4.1.17.Final.jar:/usr/lib/hadoop-mapreduce/.//hadoop-datajoin-3.1.2.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-datajoin.jar:/usr/lib/hadoop-mapreduce/.//netty-resolver-4.1.17.Final.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-distcp-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-distcp.jar:/usr/lib/hadoop-mapreduce/.//netty-
transport-4.1.17.Final.jar:/usr/lib/hadoop-mapreduce/.//hadoop-extras-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-extras.
jar:/usr/lib/hadoop-mapreduce/.//ojalgo-43.0.jar:/usr/lib/hadoop-mapreduce/.//hadoop-fs2img-3.1.2.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-fs2img.jar:/usr/lib/hadoop-mapreduce/.//hadoop-gridmix-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-
gridmix.jar:/usr/lib/hadoop-mapreduce/.//hadoop-kafka-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-kafka.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-mapreduce-client-jobclient-3.1.2-tests.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-app-3.1.2.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-rumen-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-app.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-jobclient-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-
client-common-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-common.jar:/usr/lib/hadoop-mapreduce/.//hadoop-
mapreduce-client-jobclient.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-core-3.1.2.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-sls-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-core.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-mapreduce-client-nativetask-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-hs-3.1.2.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-uploader-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-
client-hs-plugins-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-uploader.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-mapreduce-client-hs-plugins.jar:/usr/lib/hadoop-mapreduce/.//hadoop-sls.jar:/usr/lib/hadoop-
mapreduce/.//hadoop-mapreduce-client-hs.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-nativetask.
jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-examples-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-
shuffle-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-streaming-3.1.2.jar:/usr/lib/hadoop-mapreduce/.//hadoop-mapreduce-client-
shuffle.jar:/usr/lib/hadoop-mapreduce/.//hadoop-rumen.jar:/usr/lib/hadoop-yarn/./:/usr/lib/hadoop-yarn/lib/HikariCP-java7-2.4.
12.jar:/usr/lib/hadoop-yarn/lib/aopalliance-1.0.jar:/usr/lib/hadoop-yarn/lib/dnsjava-2.1.7.jar:/usr/lib/hadoop-
yarn/lib/ehcache-3.3.1.jar:/usr/lib/hadoop-yarn/lib/fst-2.50.jar:/usr/lib/hadoop-yarn/lib/geronimo-jcache_1.0_spec-1.0-alpha-1.
jar:/usr/lib/hadoop-yarn/lib/guice-4.0.jar:/usr/lib/hadoop-yarn/lib/guice-servlet-4.0.jar:/usr/lib/hadoop-yarn/lib/jackson-
jaxrs-base-2.7.8.jar:/usr/lib/hadoop-yarn/lib/jackson-jaxrs-json-provider-2.7.8.jar:/usr/lib/hadoop-yarn/lib/jackson-module-
jaxb-annotations-2.7.8.jar:/usr/lib/hadoop-yarn/lib/java-util-1.9.0.jar:/usr/lib/hadoop-yarn/lib/javax.inject-1.
jar:/usr/lib/hadoop-yarn/lib/jersey-client-1.19.jar:/usr/lib/hadoop-yarn/lib/jersey-guice-1.19.jar:/usr/lib/hadoop-
yarn/lib/json-io-2.5.1.jar:/usr/lib/hadoop-yarn/lib/metrics-core-3.2.4.jar:/usr/lib/hadoop-yarn/lib/mssql-jdbc-6.2.1.jre7.
jar:/usr/lib/hadoop-yarn/lib/objenesis-1.0.jar:/usr/lib/hadoop-yarn/lib/snakeyaml-1.16.jar:/usr/lib/hadoop-yarn/lib/swagger-
annotations-1.5.4.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-api-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-api.
jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-services-core.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-applications-distributedshell-3.
1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-applications-distributedshell.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-applications-
unmanaged-am-launcher-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-applications-unmanaged-am-launcher.jar:/usr/lib/hadoop-
yarn/.//hadoop-yarn-client-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-client.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-common-
3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-common.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-registry-3.1.2.jar:/usr/lib/hadoop-
yarn/.//hadoop-yarn-registry.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-services-core-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-
yarn-server-applicationhistoryservice-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-applicationhistoryservice.
jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-common-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-common.
jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-nodemanager-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-nodemanager.
jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-resourcemanager-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-
resourcemanager.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-router-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-
router.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-sharedcachemanager-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-
sharedcachemanager.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-tests-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-
tests.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-timeline-pluginstorage-3.1.2.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-
server-timeline-pluginstorage.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-server-web-proxy-3.1.2.jar:/usr/lib/hadoop-
yarn/.//hadoop-yarn-server-web-proxy.jar:/usr/lib/hadoop-yarn/.//hadoop-yarn-services-api-3.1.2.jar:/usr/lib/hadoop-
yarn/.//hadoop-yarn-services-api.jar
2021-12-02 07:18:15,710 INFO zookeeper.ZooKeeper: Client environment:java.library.path=/usr/lib/hadoop/lib/native
2021-12-02 07:18:15,710 INFO zookeeper.ZooKeeper: Client environment:java.io.tmpdir=/tmp
2021-12-02 07:18:15,710 INFO zookeeper.ZooKeeper: Client environment:java.compiler=<NA>
2021-12-02 07:18:15,711 INFO zookeeper.ZooKeeper: Client environment:os.name=Linux
2021-12-02 07:18:15,711 INFO zookeeper.ZooKeeper: Client environment:os.arch=amd64
2021-12-02 07:18:15,711 INFO zookeeper.ZooKeeper: Client environment:os.version=3.10.0-1160.11.1.el7.x86_64
2021-12-02 07:18:15,711 INFO zookeeper.ZooKeeper: Client environment:user.name=dasha
2021-12-02 07:18:15,711 INFO zookeeper.ZooKeeper: Client environment:user.home=/home/dasha
2021-12-02 07:18:15,711 INFO zookeeper.ZooKeeper: Client environment:user.dir=/usr/lib/phoenix/bin
2021-12-02 07:18:15,714 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=localhost:2181
sessionTimeout=90000 watcher=org.apache.hadoop.hbase.zookeeper.ReadOnlyZKClient$$Lambda$13/1224047103@df7fe85
2021-12-02 07:18:15,736 INFO zookeeper.ClientCnxn: Opening socket connection to server localhost/127.0.0.1:2181. Will not
attempt to authenticate using SASL (unknown error)
2021-12-02 07:18:15,741 INFO zookeeper.ClientCnxn: Socket connection established to localhost/127.0.0.1:2181, initiating
session
2021-12-02 07:18:15,750 INFO zookeeper.ClientCnxn: Session establishment complete on server localhost/127.0.0.1:2181, sessionid
= 0x100000335ca0024, negotiated timeout = 40000
2021-12-02 07:18:15,915 INFO query.ConnectionQueryServicesImpl: HConnection established. Stacktrace for informational purposes:
hconnection-0xe487ec java.lang.Thread.getStackTrace(Thread.java:1559)
org.apache.phoenix.util.LogUtil.getCallerStackTrace(LogUtil.java:55)
org.apache.phoenix.query.ConnectionQueryServicesImpl.openConnection(ConnectionQueryServicesImpl.java:432)
org.apache.phoenix.query.ConnectionQueryServicesImpl.access$400(ConnectionQueryServicesImpl.java:272)
org.apache.phoenix.query.ConnectionQueryServicesImpl$12.call(ConnectionQueryServicesImpl.java:2556)
org.apache.phoenix.query.ConnectionQueryServicesImpl$12.call(ConnectionQueryServicesImpl.java:2532)
org.apache.phoenix.util.PhoenixContextExecutor.call(PhoenixContextExecutor.java:76)
org.apache.phoenix.query.ConnectionQueryServicesImpl.init(ConnectionQueryServicesImpl.java:2532)
org.apache.phoenix.jdbc.PhoenixDriver.getConnectionQueryServices(PhoenixDriver.java:255)
org.apache.phoenix.jdbc.PhoenixEmbeddedDriver.createConnection(PhoenixEmbeddedDriver.java:150)
org.apache.phoenix.jdbc.PhoenixDriver.connect(PhoenixDriver.java:221)
java.sql.DriverManager.getConnection(DriverManager.java:664)
java.sql.DriverManager.getConnection(DriverManager.java:208)
org.apache.phoenix.util.QueryUtil.getConnection(QueryUtil.java:400)
org.apache.phoenix.util.QueryUtil.getConnection(QueryUtil.java:392)
org.apache.phoenix.mapreduce.AbstractBulkLoadTool.loadData(AbstractBulkLoadTool.java:206)
org.apache.phoenix.mapreduce.AbstractBulkLoadTool.run(AbstractBulkLoadTool.java:180)
org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:90)
org.apache.phoenix.mapreduce.CsvBulkLoadTool.main(CsvBulkLoadTool.java:109)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:498)
org.apache.hadoop.util.RunJar.run(RunJar.java:318)
org.apache.hadoop.util.RunJar.main(RunJar.java:232)

2021-12-02 07:18:18,132 INFO zookeeper.ReadOnlyZKClient: Connect 0x4ecebaad to localhost:2181 with session timeout=90000ms,
retries 30, retry interval 1000ms, keepAlive=60000ms
2021-12-02 07:18:18,133 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=localhost:2181
sessionTimeout=90000 watcher=org.apache.hadoop.hbase.zookeeper.ReadOnlyZKClient$$Lambda$13/1224047103@df7fe85
2021-12-02 07:18:18,135 INFO zookeeper.ClientCnxn: Opening socket connection to server localhost/127.0.0.1:2181. Will not
attempt to authenticate using SASL (unknown error)
2021-12-02 07:18:18,135 INFO zookeeper.ClientCnxn: Socket connection established to localhost/127.0.0.1:2181, initiating
session
2021-12-02 07:18:18,139 INFO zookeeper.ClientCnxn: Session establishment complete on server localhost/127.0.0.1:2181, sessionid
= 0x100000335ca0025, negotiated timeout = 40000
2021-12-02 07:18:18,145 INFO zookeeper.ReadOnlyZKClient: Connect 0x0aecb893 to localhost:2181 with session timeout=90000ms,
retries 30, retry interval 1000ms, keepAlive=60000ms
2021-12-02 07:18:18,146 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=localhost:2181
sessionTimeout=90000 watcher=org.apache.hadoop.hbase.zookeeper.ReadOnlyZKClient$$Lambda$13/1224047103@df7fe85
2021-12-02 07:18:18,147 INFO zookeeper.ClientCnxn: Opening socket connection to server localhost/127.0.0.1:2181. Will not
attempt to authenticate using SASL (unknown error)
2021-12-02 07:18:18,147 INFO zookeeper.ClientCnxn: Socket connection established to localhost/127.0.0.1:2181, initiating
session
2021-12-02 07:18:18,150 INFO zookeeper.ClientCnxn: Session establishment complete on server localhost/127.0.0.1:2181, sessionid
= 0x100000335ca0026, negotiated timeout = 40000
2021-12-02 07:18:18,362 INFO mapreduce.MultiHfileOutputFormat:  the table logical name is BEST_BOOKS
2021-12-02 07:18:18,362 INFO client.ConnectionImplementation: Closing master protocol: MasterService
2021-12-02 07:18:18,363 INFO zookeeper.ReadOnlyZKClient: Close zookeeper connection 0x0aecb893 to localhost:2181
2021-12-02 07:18:18,365 INFO mapreduce.MultiHfileOutputFormat: Configuring 1 reduce partitions to match current region count
2021-12-02 07:18:18,365 INFO mapreduce.MultiHfileOutputFormat: Writing partition information to /tmp/hadoop-
dasha/partitions_dfe383e6-d920-4829-8340-a86b2926db27
2021-12-02 07:18:18,366 INFO zookeeper.ZooKeeper: Session: 0x100000335ca0026 closed
2021-12-02 07:18:18,367 INFO zookeeper.ClientCnxn: EventThread shut down for session: 0x100000335ca0026
2021-12-02 07:18:18,489 INFO zlib.ZlibFactory: Successfully loaded & initialized native-zlib library
2021-12-02 07:18:18,490 INFO compress.CodecPool: Got brand-new compressor [.deflate]
2021-12-02 07:18:18,643 INFO mapreduce.AbstractBulkLoadTool: Running MapReduce import job from /user/dasha/best_books1.csv to
/tmp/2533c78e-2a96-47f4-9e06-aaec783f979f
2021-12-02 07:18:18,836 INFO client.RMProxy: Connecting to ResourceManager at bds-adh-2.ru-central1.internal/10.92.6.9:8032
2021-12-02 07:18:18,965 INFO client.AHSProxy: Connecting to Application History server at bds-adh-3.ru-central1.internal/10.92.
6.90:10200
2021-12-02 07:18:19,083 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /user/dasha/.
staging/job_1638426545038_0003
2021-12-02 07:18:20,061 INFO input.FileInputFormat: Total input files to process : 1
2021-12-02 07:18:20,159 INFO mapreduce.JobSubmitter: number of splits:1
2021-12-02 07:18:20,198 INFO Configuration.deprecation: io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2021-12-02 07:18:20,291 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1638426545038_0003
2021-12-02 07:18:20,291 INFO mapreduce.JobSubmitter: Executing with tokens: []
2021-12-02 07:18:20,445 INFO conf.Configuration: resource-types.xml not found
2021-12-02 07:18:20,445 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2021-12-02 07:18:20,506 INFO impl.YarnClientImpl: Submitted application application_1638426545038_0003
2021-12-02 07:18:20,642 INFO mapreduce.Job: The url to track the job: http://bds-adh-2.ru-central1.
internal:8088/proxy/application_1638426545038_0003/
2021-12-02 07:18:20,642 INFO mapreduce.Job: Running job: job_1638426545038_0003
2021-12-02 07:18:29,739 INFO mapreduce.Job: Job job_1638426545038_0003 running in uber mode : false
2021-12-02 07:18:29,741 INFO mapreduce.Job:  map 0% reduce 0%
2021-12-02 07:18:37,797 INFO mapreduce.Job:  map 100% reduce 0%
2021-12-02 07:18:46,833 INFO mapreduce.Job:  map 100% reduce 100%
2021-12-02 07:18:46,838 INFO mapreduce.Job: Job job_1638426545038_0003 completed successfully
2021-12-02 07:18:46,920 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=5824
                FILE: Number of bytes written=534283
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=2839
                HDFS: Number of bytes written=8614
                HDFS: Number of read operations=14
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=5
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=6406
                Total time spent by all reduces in occupied slots (ms)=6027
                Total time spent by all map tasks (ms)=6406
                Total time spent by all reduce tasks (ms)=6027
                Total vcore-milliseconds taken by all map tasks=6406
                Total vcore-milliseconds taken by all reduce tasks=6027
                Total megabyte-milliseconds taken by all map tasks=6559744
                Total megabyte-milliseconds taken by all reduce tasks=6171648
        Map-Reduce Framework
                Map input records=70
                Map output records=70
                Map output bytes=5678
                Map output materialized bytes=5824
                Input split bytes=103
                Combine input records=0
                Combine output records=0
                Reduce input groups=70
                Reduce shuffle bytes=5824
                Reduce input records=70
                Reduce output records=140
                Spilled Records=140
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=230
                CPU time spent (ms)=8510
                Physical memory (bytes) snapshot=1084018688
                Virtual memory (bytes) snapshot=5900730368
                Total committed heap usage (bytes)=815267840
                Peak Map Physical memory (bytes)=597655552
                Peak Map Virtual memory (bytes)=2946670592
                Peak Reduce Physical memory (bytes)=486363136
                Peak Reduce Virtual memory (bytes)=2954059776
        Phoenix MapReduce Import
                Upserts Done=70
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=2736
        File Output Format Counters
                Bytes Written=8614
2021-12-02 07:18:46,922 INFO mapreduce.AbstractBulkLoadTool: Loading HFiles from /tmp/2533c78e-2a96-47f4-9e06-aaec783f979f
2021-12-02 07:18:46,950 INFO Configuration.deprecation: io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2021-12-02 07:18:46,953 INFO zookeeper.ReadOnlyZKClient: Connect 0x59a2153e to localhost:2181 with session timeout=90000ms,
retries 30, retry interval 1000ms, keepAlive=60000ms
2021-12-02 07:18:46,953 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=localhost:2181
sessionTimeout=90000 watcher=org.apache.hadoop.hbase.zookeeper.ReadOnlyZKClient$$Lambda$13/1224047103@df7fe85
2021-12-02 07:18:46,955 INFO zookeeper.ClientCnxn: Opening socket connection to server localhost/127.0.0.1:2181. Will not
attempt to authenticate using SASL (unknown error)
2021-12-02 07:18:46,955 INFO zookeeper.ClientCnxn: Socket connection established to localhost/127.0.0.1:2181, initiating
session
2021-12-02 07:18:46,958 INFO zookeeper.ClientCnxn: Session establishment complete on server localhost/127.0.0.1:2181, sessionid
= 0x100000335ca0027, negotiated timeout = 40000
2021-12-02 07:18:46,959 INFO mapreduce.AbstractBulkLoadTool: Loading HFiles for BEST_BOOKS from /tmp/2533c78e-2a96-47f4-9e06-
aaec783f979f/BEST_BOOKS
2021-12-02 07:18:47,126 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2-hbase.properties
2021-12-02 07:18:47,132 WARN impl.MetricsSystemImpl: Error creating sink 'tracing'
org.apache.hadoop.metrics2.impl.MetricsConfigException: Error creating plugin: org.apache.phoenix.trace.PhoenixMetricsSink
        at org.apache.hadoop.metrics2.impl.MetricsConfig.getPlugin(MetricsConfig.java:211)
        at org.apache.hadoop.metrics2.impl.MetricsSystemImpl.newSink(MetricsSystemImpl.java:531)
        at org.apache.hadoop.metrics2.impl.MetricsSystemImpl.configureSinks(MetricsSystemImpl.java:503)
        at org.apache.hadoop.metrics2.impl.MetricsSystemImpl.configure(MetricsSystemImpl.java:479)
        at org.apache.hadoop.metrics2.impl.MetricsSystemImpl.start(MetricsSystemImpl.java:188)
        at org.apache.hadoop.metrics2.impl.MetricsSystemImpl.init(MetricsSystemImpl.java:163)
        at org.apache.hadoop.metrics2.lib.DefaultMetricsSystem.init(DefaultMetricsSystem.java:62)
        at org.apache.hadoop.metrics2.lib.DefaultMetricsSystem.initialize(DefaultMetricsSystem.java:58)
        at org.apache.hadoop.hbase.metrics.BaseSourceImpl$DefaultMetricsSystemInitializer.init(BaseSourceImpl.java:54)
        at org.apache.hadoop.hbase.metrics.BaseSourceImpl.<init>(BaseSourceImpl.java:116)
        at org.apache.hadoop.hbase.io.MetricsIOSourceImpl.<init>(MetricsIOSourceImpl.java:46)
        at org.apache.hadoop.hbase.io.MetricsIOSourceImpl.<init>(MetricsIOSourceImpl.java:38)
        at org.apache.hadoop.hbase.regionserver.MetricsRegionServerSourceFactoryImpl.
createIO(MetricsRegionServerSourceFactoryImpl.java:84)
        at org.apache.hadoop.hbase.io.MetricsIO.<init>(MetricsIO.java:35)
        at org.apache.hadoop.hbase.io.hfile.HFile.<clinit>(HFile.java:195)
        at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.visitBulkHFiles(LoadIncrementalHFiles.java:1025)
        at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.discoverLoadQueue(LoadIncrementalHFiles.java:941)
        at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.prepareHFileQueue(LoadIncrementalHFiles.java:224)
        at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.doBulkLoad(LoadIncrementalHFiles.java:331)
        at org.apache.hadoop.hbase.tool.LoadIncrementalHFiles.doBulkLoad(LoadIncrementalHFiles.java:256)
        at org.apache.phoenix.mapreduce.AbstractBulkLoadTool.completebulkload(AbstractBulkLoadTool.java:355)
        at org.apache.phoenix.mapreduce.AbstractBulkLoadTool.submitJob(AbstractBulkLoadTool.java:331)
        at org.apache.phoenix.mapreduce.AbstractBulkLoadTool.loadData(AbstractBulkLoadTool.java:267)
        at org.apache.phoenix.mapreduce.AbstractBulkLoadTool.run(AbstractBulkLoadTool.java:180)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:76)
        at org.apache.hadoop.util.ToolRunner.run(ToolRunner.java:90)
        at org.apache.phoenix.mapreduce.CsvBulkLoadTool.main(CsvBulkLoadTool.java:109)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:318)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:232)
Caused by: java.lang.ClassNotFoundException: org.apache.phoenix.trace.PhoenixMetricsSink
        at java.net.URLClassLoader.findClass(URLClassLoader.java:387)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
        at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
        at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Class.java:348)
        at org.apache.hadoop.metrics2.impl.MetricsConfig.getPlugin(MetricsConfig.java:205)
        ... 32 more
2021-12-02 07:18:47,203 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2021-12-02 07:18:47,203 INFO impl.MetricsSystemImpl: HBase metrics system started
2021-12-02 07:18:47,220 INFO metrics.MetricRegistries: Loaded MetricRegistries class org.apache.hadoop.hbase.metrics.impl.
MetricRegistriesImpl
2021-12-02 07:18:47,346 INFO hfile.CacheConfig: Created cacheConfig: CacheConfig:disabled
2021-12-02 07:18:47,412 INFO tool.LoadIncrementalHFiles: Trying to load hfile=hdfs://adh1/tmp/2533c78e-2a96-47f4-9e06-
aaec783f979f/BEST_BOOKS/0/f544e36fd2044f07bce46e920d6c4c60 first=Optional[A.S. Byatt\x00Possession] last=Optional[Zora Neale
Hurston\x00Their Eyes Were Watching God]
2021-12-02 07:18:47,526 INFO mapreduce.AbstractBulkLoadTool: Incremental load complete for table=BEST_BOOKS
2021-12-02 07:18:47,526 INFO mapreduce.AbstractBulkLoadTool: Removing output directory /tmp/2533c78e-2a96-47f4-9e06-
aaec783f979f
```

7. Check the results of the command. You can run SQLLine and perform the following query to display the first ten records of the created table:

```
SELECT * FROM BEST_BOOKS LIMIT 10;
```

The output:

```
+-----------------------+------------------------------------------------------+--------------+
|        AUTHOR         |                        TITLE                         | PUBLIC_YEAR  |
+-----------------------+------------------------------------------------------+--------------+
| A.S. Byatt            | Possession                                           | 1990         |
| Alan Moore            | Watchmen                                             | 1987         |
| Anthony Burgess       | A Clockwork Orange                                   | 1962         |
| C.S. Lewis            | The Lion, the Witch and the Wardrobe                 | 1950         |
| Carson McCullers      | The Heart Is a Lonely Hunter                         | 1940         |
| Chinua Achebe         | Chinua Achebe                                        | 1958         |
| Cormac McCarthy       | Blood Meridian, or the Evening Redness in the West   | 1985         |
| Dashiell Hammett      | Red Harvest                                          | 1929         |
| David Foster Wallace  | Infinite Jest                                        | 1996         |
| Don DeLillo           | White Noise                                          | 1985         |
+-----------------------+------------------------------------------------------+--------------+
10 rows selected (0.092 seconds)
```

You can also get the rows count of the table:

```
SELECT COUNT(*) FROM BEST_BOOKS;
```

The output:

```
+-----------+
| COUNT(1)  |
+-----------+
| 70        |
+-----------+
1 row selected (0.048 seconds)
```

8. Finally, run the `list_regions` command in HBase shell. You can see that the value of the `LOCALITY` column is `1`. It proves that the second method of bulk loading writes data directly into HFiles in HBase.

```
list_regions 'BEST_BOOKS'
```

The output:

```
                                          SERVER_NAME |                                                          REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ----------------------------------------------------------- | ---------- |
---------- | ----- | ----- | ---------- |
 bds-adh-1.ru-central1.internal,16020,1638426590054 | BEST_BOOKS,,1638427992879.8a84103a2b4ed50e718587bcc8a1e63b. |            |
|     0 |    81 |        1.0 |
 1 rows
Took 0.6029 seconds
```

Phoenix performance tuning

Vladimir Adamenkov

Contents

To Table of Contents

- Configuration tuning
  - Connection
  - Query
  - JDBC URL
  - Region server
- Database tuning
  - Row key design
  - Column family design
  - Indexing strategies
  - Data compression
  - Data locality
- Query optimization techniques
  - Query execution plan
  - SQL efficiency
  - JOIN optimization
  - Hints
  - Query server

This article contains recommendations on how to optimize the performance of Phoenix in ADH.

Configuration tuning

Connection

Parameters described below are related to connection details in Phoenix.

Connection parameters

Parameter	Default and description	Recommendation
phoenix.query.timeoutMs	Default: `600000` (in milliseconds). Defines the time the query is allowed to run before it is terminated on the client side. This prevents long-running or runaway queries from consuming excessive resources and potentially causing performance issues	Set a relatively conservative value (`30000` or `60000`) and monitor the actual execution times of your queries. Use the HBase web UI, Phoenix `EXPLAIN PLAN` command, or application-level logging for that purpose. Identify the queries that consistently exceed the current timeout. If you find that the timeout is too restrictive and causes the queries that should succeed to consistently fail, gradually increase the value by small amounts (e.g. 30 seconds or 1 minute at a time) and monitor the results. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.query.threadPoolSize	Default: `128`. This parameter defines the maximum number of threads available in the thread pool dedicated to executing incoming Phoenix queries. Each thread handles a single query (or a portion of a complex query)	Start with the default value and run the typical workload. Monitor the key metrics: - CPU usage; - RAM usage; - query latency; - thread pool metrics. If the CPUs are underutilized, the query latency is high, or query queue depth is consistently high, increase the parameter by 25%. If the CPUs are maxed out and the RAM is exhausted, decrease the parameter by 25%. After each adjustment, rerun the workload and monitor the metrics again. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
hbase.client.scanner.timeout.period	Default: `60000` (in milliseconds). This parameter defines a limit for a scanner inactivity before the Region server closes it	Start with the default value and run the typical workload. Search the logs for `scanner timed out` exceptions and/or long-running queries being interrupted. If there happens to be a lot of such occasions, increase the parameter value. Rerun the workload and check the logs again. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page
zookeeper.session.timeout	Default: `90000` (in milliseconds). This parameter defines the timeout for ZooKeeper sessions. HBase (and therefore Phoenix) relies on ZooKeeper for coordination. If a Region server loses its ZooKeeper session, it can lead to connection issues and instability	In environments with unstable networks or high ZooKeeper load, consider increasing this value. However, a higher value means it will take longer to detect a truly failed Region server. Do not set this too low, as transient network hiccups can cause unnecessary disconnections. This needs to be coordinated with the ZooKeeper configuration itself. This parameter is located in the ***hbase-site.xml*** section of the HBase service configuration page

Query

Parameters described below are related to query performance in Phoenix.

Query parameters

Parameter	Default and description	Recommendation
phoenix.query.spoolThresholdBytes	Default: `20971520` (in bytes). This parameter limits the amount of memory a query can use for sorting or grouping results before spilling data to disk for temporary storage. It is a critical parameter for queries that use `ORDER BY`, `GROUP BY`, or `JOIN` clauses, especially when dealing with large datasets. Insufficient memory can lead to excessive disk I/O, significantly impacting query performance	Set this parameter to `10485760` (10 MB) and run the typical workload. Monitor the key metrics: - Query latency — measure the average and 95th/99th percentile query latency. - Disk I/O utilization. - Garbage Collector (GC) activity. If any of those metrics are too high — double the parameter value, rerun the workload and monitor the metrics again. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.query.maxRows	Default: `-1` (no limit). This parameter limits the maximum number of rows that a single query can return. If a query attempts to return more rows than this limit, Phoenix will throw an exception (`org.apache.phoenix.exception.PhoenixIOException`) and terminate the query	Start with the default value and run the typical workload. Monitor query execution and identify any queries that return unexpectedly large result sets or cause performance problems. Examine query logs to understand the typical result set sizes for different types of queries. Identify any queries that consistently return a very large number of rows. Based on your findings, set the parameter value that is high enough to accommodate most of your legitimate queries but low enough to prevent runaway queries from overwhelming the system. Set a relatively conservative value (e.g. `10000` or `100000`) and rerun the workload. Monitor query execution and look for queries that are terminated due to exceeding the `phoenix.query.maxRows` parameter value. If you find legitimate queries that are being terminated, increase the limit slightly and rerun the workload again. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.query.resultset.cache.enabled	Default: `false`. This parameter defines whether the client-side caching of query results is enabled. If enabled, Phoenix will cache the results of queries, allowing subsequent identical queries to return results from the cache instead of re-executing on the server	Enable this parameter if you have frequently executed, read-heavy queries with stable data. This can drastically improve read performance. However, it increases client memory consumption and introduces a potential for stale data if the underlying data changes frequently and the cache is not invalidated. Monitor client memory and data freshness. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.query.optimizeMaxParallelScans	Default: `10`. This parameter defines the maximum number of parallel scans that Phoenix can initiate for certain types of queries (e.g. queries that involve multiple parallel scans across different regions or tables)	If you have many regions per table or a highly distributed dataset, increasing this value might improve performance for complex analytical queries that can benefit from parallelism. Increase with caution, as a very high value can put significant load on the cluster meta-data services and on the network. Monitor Region server and ZooKeeper activity. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.mutate.batchSize	Default: `1000`. This parameter specifies the number of mutation operations (`INSERT`, `UPDATE`, or `DELETE`) that are accumulated on the client side before being sent to the HBase server in a single batch. A larger batch size reduces the number of round trips to the server, which can improve write performance	Increase this value for higher write throughput if you are performing bulk loading or batch updates. Start by doubling the default value and monitor performance. If you are experiencing `OutOfMemoryError` on the client side or on Region servers, or if a Region server overloads due to large batches, decrease the value. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.mutate.maxSizeBytes	Default: `10485760` (in bytes). Defines the maximum size of a single mutation batch. This prevents very large mutations from overwhelming the region servers	If you are inserting rows with very large values (e.g. large binary data) and the default limit is consistently exceeded, increase this value. Be careful not to set this too high, as it can lead to region server issues. If you are seeing errors related to oversized mutations, decrease this value. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

CAUTION

The last two properties (`phoenix.mutate.batchSize` and `phoenix.mutate.maxSizeBytes`) work together. The batch will be sent when either limit is reached.

JDBC URL

Parameters described below can be used in the JDBC URL string when connecting to Phoenix.

JDBC URL parameters

Parameter	Default and description	Recommendation
phoenix.query.client.fetchSize	Default: `1000`. This parameter specifies the number of rows that the client requests from the server in each fetch operation during query execution. A larger fetch size reduces the number of network round trips, potentially improving performance for queries that return many rows	Increase the value for queries that retrieve a large number of rows to reduce network latency overhead. Decrease the value if the memory on the client becomes an issue or if network conditions are poor, causing large fetches to time out. It is often beneficial to match this value with the `phoenix.statement.prefetchRows` parameter value if you use it (described next)
phoenix.statement.prefetchRows	Default: `0` (disabled). This parameter specifies the number of rows pre-fetched into the client local buffer. When set to a value greater than `0`, Phoenix will attempt to fetch the specified number of rows ahead of what is currently being consumed by the application, smoothing out the execution of the `next()` calls	Specify a value greater than `0` (e.g. `1000` or `5000`) for queries where the client iterates through results using `next()`. This can significantly improve perceived query performance by reducing latency between row fetches. A value too high might consume excessive client memory
phoenix.schema.isNamespaceMappingEnabled	Default: `false`. This parameter determines whether Phoenix should use HBase namespaces to represent Phoenix schemas	Set this parameter to `true` if you want to use HBase namespaces for schema management. This provides better separation of concerns and can improve performance in multi-tenant environments by leveraging HBase namespace-based security and resource management. This requires proper setup and configuration of HBase namespaces. If HBase namespaces are not used, leave it as `false`
phoenix.use.stats	Default: `true`. This parameter defines whether the use of statistics is enabled when planning queries	Leave this value as `true` to allow Phoenix to make informed decisions about query execution based on table statistics. If you have a very small dataset or know that your statistics are significantly out of date, you might temporarily set it to `false` to force Phoenix to use a different query plan. However, in most cases, it is best to keep statistics enabled and ensure they are regularly updated. You can force the statistics update with the `UPDATE STATISTICS <table_name>` query
async	Default: `false`. This parameter defines whether asynchronous queries are enabled. This allows the client to submit queries and receive results later	Set this parameter to `true` if you want to submit queries without blocking. The results can be processed later. It requires additional code and infrastructure to handle the asynchronous results

Region server

Parameters described below are related to the Region servers functioning in respect to Phoenix.

Region server parameters

Parameter	Default and description	Recommendation
hbase.rpc.timeout	Default: `60000` (in milliseconds). This parameter defines the general timeout for RPC calls to HBase region servers. If a client does not receive a response from a Region server within this time, the RPC call is considered failed. Phoenix uses RPC calls for virtually all operations	If you are frequently seeing `RpcTimeoutException` or `CallTimeoutException` errors, which often happens when a Region server is under heavy load or there is network latency, increase this value. Increase by small increments (e.g. by 10000 at a time) and monitor your logs. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.coprocessor.maxServerCacheBytes	Default: `0` (disabled). This parameter defines the server side cache size for query results within the coprocessor on the Region servers. This can significantly improve performance for frequently executed queries with relatively small result sets. The value is defined in bytes	If you have many repeated queries, start with a modest value (e.g. `104857600` for 100 MB). Closely monitor the memory usage on your Region servers. A value too high can cause memory pressure and performance problems. You also need to monitor the cache usage rate. If it is too low, then you are wasting memory. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page
phoenix.index.builder.threads	Default: `1`. This parameter specifies the number of threads that will be used to build global indexes. Global indexes are built asynchronously in the background, and this parameter controls the level of parallelism used for index building	If you have a large number of rows to index and want to speed up the index building process, increase this value up to the number of available CPU cores on your Region servers. If the index building process is putting too much load on the Region servers, decrease this value. This parameter has to be added manually in the ***Custom hbase-site.xml*** section of the HBase service configuration page

Database tuning

Row key design

The row key is the most important design consideration in HBase and, therefore, in Phoenix. A well-designed row key can significantly improve query performance and data locality. A poorly designed row key can lead to hotspots, slow queries, and overall system inefficiency.

Below is the list of key considerations for row key design:

- **Query patterns**. The row key should be chosen based on how you intend to query the data. Identify the most common and performance-critical queries and design the row key to optimize for those. Consider the following:
  - **Range scans**. If you frequently need to retrieve data within a specific range, the row key should include the date or a derivative of it as a prefix.
  - **Point lookups**. If you need to retrieve individual rows quickly based on a unique identifier, make that identifier the row key.
  - **Prefix-based queries**. If you often use queries based on a prefix of a specific attribute, include that attribute at the beginning of the row key.
- **Hotspotting**. Avoid row key designs that lead to hotspotting. It occurs when a disproportionate amount of data is written to or read from a single Region server, creating a bottleneck. Common causes:
  - **Monotonically increasing keys**. Using timestamps, sequence numbers, or other monotonically increasing values as the row key will direct all new writes to a single region, overloading that server.
  - **Sequential keys**. Similar to monotonically increasing keys, sequential keys can also cause hotspots if they are concentrated within a small range.
- Salting. This is a technique to mitigate hotspotting by adding a prefix to the row key to distribute writes across multiple regions.
- **Pre-splitting**. Pre-splitting involves creating multiple regions upfront to distribute the initial load more evenly. This is especially important for large datasets.
- **Key length**. Keep row keys as short as possible without sacrificing uniqueness or query performance. Shorter keys reduce storage space and improve scan performance.
- **Composite keys**. Phoenix supports composite row keys, which combine multiple columns into a single key. This is useful when you need to execute queries based on a combination of attributes.
  - **Order of columns**. The order of columns in the composite key is significant. Place the most frequently queried columns at the beginning.
  - **Data types**. Choose appropriate data types for each column in the composite key to optimize storage space and performance.

Example

Consider a table storing sensor data with the following columns:

- `sensor_id`
- `timestamp`
- `temperature`
- `humidity`

If you frequently query for sensor data within a specific time range, you might design the composite row key as a combination of `sensor_id` and `timestamp`. However, if the number of sensors is small and writing to specific time ranges is concentrated, you will have the hotspotting issue. In that case, you would need to salt the `sensor_id` with some number of salt regions.

Column family design

In HBase and Phoenix, the data is stored in column families. Choosing the right column family design is crucial for optimizing storage efficiency, query performance, and overall system scalability.

Below is the list of key considerations for column family design:

- **Locality of reference**. Columns that are accessed together frequently should be grouped into the same column family. This improves read performance because HBase retrieves entire column families at once. Conversely, columns that are rarely accessed together should be placed in separate column families.
- **Number of column families**. Minimize the number of column families in a table. Each column family has associated overhead in terms of storage and memory. A small number of well-designed column families is generally preferable to a large number of fragmented families. Aim for a few (2-3) column families.
- **Storage characteristics**. Consider the storage characteristics of the data within each column family:
  - **Data type**. Columns with similar data types (e.g. strings, numbers, binaries) can be grouped together to optimize storage and compression.
  - **Data size**. Columns containing large objects (e.g. images, documents) should be placed in their own column family to avoid impacting the performance of other frequently accessed data.
- **Schema evolution**. Column families provide a degree of flexibility for schema evolution. You can add new columns to an existing column family without affecting other families. However, consider the impact of adding a large number of columns to a single column family, as it can affect performance.
- **Column family naming**. Choose descriptive and meaningful names for your column families, but keep them as short as possible.
- **Compression**. Column families can be configured with different compression algorithms. Choose the appropriate compression algorithm based on the data type and access patterns within each family.
- **In-memory storage**. You can configure a column family to be stored in memory for faster access. This is useful for frequently accessed data that is relatively small. However, be mindful of memory usage on your region servers.

Example

Consider a table storing user profile information with the following columns: `user_id`, `username`, `email`, `first_name`, `last_name`, `address`, `city`, `state`, `zip`, and `profile_picture`. You might design the column families as follows:

- `info`: `username`, `email`, `first_name`, `last_name`
- `address`: `address`, `city`, `state`, `zip`
- `media`: `profile_picture`

This design groups the frequently accessed user information into the `info` column family, separates the address information into its own family, and places the large profile picture into a separate family to avoid impacting the performance of other queries.

Indexing strategies

Phoenix provides powerful indexing capabilities to accelerate query performance on HBase data. Unlike traditional relational databases, where indexes are typically created on a small number of columns, Phoenix allows you to create indexes on any combination of columns, including expressions and functions. Choosing the right indexing strategy is crucial for optimizing query performance and minimizing index maintenance overhead.

There are two main types of indexes in Phoenix: global and local.

Global indexes

Global indexes are stored in a separate HBase table and cover all rows in the original table. When a query uses a global index, Phoenix retrieves the data from the index table instead of the original table.

Global indexes are best suited for read-heavy workloads where queries frequently filter on the indexed columns. They are updated synchronously during data writes. This can impact write performance, especially for large datasets.

Global indexes are not usable if the underlying table is not available.

Local indexes

Local indexes are stored on the same region servers as the original table data. When a query uses a local index, Phoenix retrieves the data from the index table and the original table within the same region server.

Local indexes are ideal for write-heavy workloads where queries need to filter on non-leading columns of the row key or columns that are not part of a global index. They are updated asynchronously after data writes. This minimizes the impact on write performance.

Local indexes are available even if the underlying table is not available.

Syntax examples for creating a global and a local index:

```
CREATE INDEX my_index ON my_table (column1, column2);
CREATE LOCAL INDEX my_index ON my_table (column1, column2);
```

Data compression

Data compression is a crucial technique for reducing storage space and improving I/O performance in HBase and Phoenix. By compressing data before it is stored, you can significantly reduce the amount of disk space required and the time it takes to read and write data. However, compression also introduces overhead due to the CPU cycles required to compress and decompress data. Choosing the right compression algorithm is a trade-off between storage savings, I/O performance, and CPU usage.

HBase supports several compression algorithms listed below, each with its own characteristics:

- **NONE**. No compression is used. This is the default algorithm. It provides the fastest read and write performance but consumes the most storage space.
- **LZO**. A fast and widely used compression algorithm that provides a good balance between compression ratio and CPU usage. Requires separate installation and configuration.
- **GZIP**. A popular compression algorithm that offers a high compression ratio but consumes more CPU cycles than LZO. Suitable for data that is not frequently accessed.
- **Snappy**. A fast compression algorithm that is designed for speed rather than high compression ratios. Ideal for data that is frequently accessed and requires low latency.
- **LZ4**. Extremely fast compression algorithm but offers lower compression ratios than other algorithms.

The best compression algorithm for your data depends on several factors:

- **Data type**. Some compression algorithms are better suited for certain data types than others. For example, GZIP is often a good choice for text data, while Snappy is suitable for binary data.
- **Access patterns**. If your data is frequently accessed, you should choose a compression algorithm that offers low latency, such as Snappy or LZ4. If your data is rarely accessed, you can choose a compression algorithm that provides a high compression ratio, such as GZIP.
- **CPU usage**. Be mindful of the CPU overhead of compression. If your system is already CPU-bound, you may want to choose a compression algorithm that is less CPU-intensive, such as LZO or Snappy.
- **Storage capacity**. If your storage capacity is limited, you should choose a compression algorithm that provides a high compression ratio.

Data locality

Data locality is a key concept in HBase and Phoenix that refers to the proximity of data to the compute resources that need to access it. Maximizing data locality is crucial for achieving high performance because it reduces the amount of data that needs to be transferred over the network, which can be a significant bottleneck.

HBase stores data in regions, which are distributed across Region servers in the cluster. Each Region server is responsible for managing a subset of the data. When a client application requests data, HBase attempts to serve the request from the region server that is closest to the data.

Data locality is affected by the following factors:

- **Row key design**. The row key is the most important factor affecting data locality. A well-designed row key will group related data together on the same region server, making it easier to access the data efficiently.
- **Region splitting and merging**. HBase automatically splits and merges regions based on data size and load. If regions are not split and merged appropriately, data locality can suffer. It is important to monitor region sizes and configure the region splitting and merging policies accordingly.
- **Data placement**. HBase attempts to place data on Region servers that are close to the data consumers. However, factors such as network topology and rack awareness can affect data placement.
- **Compaction**. Compaction is the process of merging and rewriting data files in HBase. Compaction can improve data locality by reorganizing data on disk.
- **Bulk loading**. Using bulk loading to load data into HBase allows you to pre-sort your data which can increase data locality.

You can use the following strategies for improving the data locality:

- **Optimal row key design**. A well-designed row key is essential for data locality. Choose a row key that groups related data together on the same region server. Consider salting to prevent hotspotting, while still maintaining locality for related data within the same salted region.
- **Region pre-splitting**. Pre-splitting the table into multiple regions can improve data locality by distributing the data across multiple Region servers from the start.
- **Rack awareness**. Configure HBase to be aware of the rack topology of your cluster. This allows HBase to place data on region servers that are in the same rack as the data consumers.
- **HDFS configuration**. Properly configure HDFS to ensure that data is stored close to the region servers.

Query optimization techniques

Query execution plan

Understanding the query execution plan is fundamental to optimizing SQL queries in Phoenix. The execution plan outlines the steps Phoenix will take to retrieve and process the data. By analyzing this plan, you can identify potential bottlenecks and areas for improvement.

You can obtain the query execution plan using the `EXPLAIN PLAN` statement in Phoenix. For example:

```
EXPLAIN PLAN SELECT * FROM users WHERE age > 30 AND city = 'New York';
```

Result example:

```
CLIENT 1-WAY PARALLEL 1/1 [SERVER FILTER]
    SERVER FILTER BY "AGE" > 30
        CLIENT 1-WAY FULL SCAN OVER USERS
```

The execution plan is typically presented as a tree structure, with the root node representing the final result and the leaf nodes representing the data sources. Each node in the tree describes an operation that Phoenix will perform. Below is a table with the description of the execution plan operations.

Execution plan operations

Operation	Description
SERIAL	Indicates that the operation will be performed serially on a single region server. This is generally less efficient than parallel execution
PARALLEL <n>-WAY	Indicates that the operation will be performed in parallel across multiple region servers, with `<n>` specifying the number of ways (degree of parallelism). Look for opportunities to increase parallelism where appropriate, but be mindful of resource constraints
FULL SCAN OVER <table_name>	Indicates a full table scan, meaning that Phoenix will read every row in the table to find the matching rows. Full table scans are generally inefficient, especially for large tables. A full scan over an index table is not necessarily inefficient
RANGE SCAN OVER <table_name> <startRow, stopRow>	Indicates a range scan, where Phoenix will read only a subset of the rows in the table based on a range of row keys. This is more efficient than a full table scan, but it can still be inefficient if the range is larger than necessary. An index table will define its ranges differently from the original table
POINT LOOKUP OVER <table_name> <rowKey>	Indicates a point lookup, where Phoenix will directly retrieve a single row based on its row key. This is the most efficient type of read operation
CLIENT <op type>	Indicates that the query execution is happening on the client side
JOIN <join type>	Indicates the type of the `JOIN` operation performed (e.g. `STAR JOIN`). Analyzing join operations is crucial for optimization
FILTER <predicate>	Indicates that Phoenix is filtering the data based on a predicate (the `WHERE` clause). Ensure the filter is as selective as possible
AGGREGATE <aggregation function>	Indicates that Phoenix is performing an aggregation function (e.g. `SUM`, `AVG`, `COUNT`)
SORT	Indicates that the results are being sorted. Sorting can be expensive, so avoid it if possible. If you can use an index to provide the required sort order, do so

Based on the analysis of the execution plan, you can take steps to improve query performance. In the example above, you could create an index on the `age` and `city` columns like this:

```
CREATE INDEX idx_users_age_city ON users (age, city);
```

After creating the index, the execution plan might look like the following:

```
CLIENT 1-WAY RANGE SCAN OVER IDX_USERS_AGE_CITY [30,]
    SERVER FILTER BY "CITY" = 'New York'
```

SQL efficiency

Writing efficient SQL queries is paramount for optimal performance in Phoenix. While Phoenix leverages HBase for storage, inefficient SQL can still lead to slow query execution, even with properly indexed tables. This section focuses on crafting SQL queries that minimize the workload on Phoenix and maximize resource utilization.

Key principles for writing efficient SQL:

- **Select only necessary columns**. Explicitly specify the columns you need in the `SELECT` clause. Using the `SELECT *` clause retrieves all columns from the table, even if you only need a few. This increases the amount of data that needs to be read, transferred, and processed.
- **Filter early and efficiently**. Applying filters late in the query execution (e.g. after a full table scan) can be very inefficient. Use the `WHERE` clause to filter data as early as possible. Ensure that your `WHERE` clause leverages available indexes. Place the most restrictive conditions in the `WHERE` clause first.
- **Use the correct data types**. Inconsistent data types in comparisons and predicates can prevent Phoenix from using indexes effectively and may lead to type conversion overhead. Ensure that the data types in your SQL queries match the data types of the corresponding columns in the table schema.
- **Leverage built-in functions efficiently**. Some built-in functions can be more efficient than others. Understand the performance characteristics of different built-in functions. Use functions that are optimized for HBase/Phoenix.
- **Avoid unnecessary subqueries**. Subqueries can often be rewritten as joins, which can be more efficient in Phoenix. Correlated subqueries (subqueries that depend on the outer query) can be particularly problematic. Try to rewrite subqueries as joins whenever possible. Consider denormalization as an alternative if the subquery is used frequently.
- **Limit the result set**. Retrieving large result sets can put a strain on the client and network. Use the `LIMIT` clause to restrict the number of rows returned. If pagination is needed, use `LIMIT` and `OFFSET` in conjunction.
- **Use COUNT(*) carefully**. `COUNT(*)` on a large table without a `WHERE` clause can be very expensive, as it requires scanning the entire table. If you need to count all rows in a table, consider using a pre-aggregated count or a lightweight approach if the table is not excessively large. If possible, add a `WHERE` clause to restrict the count to a smaller subset of the data.
- **Leverage UPSERT SELECT for efficient data updates and inserts**. Inserting data row-by-row can be very inefficient. Use `UPSERT SELECT` to insert or update data in batch. This is especially useful for loading data from other tables or external sources.

JOIN optimization

Joins in Phoenix can be relatively expensive operations compared to point lookups or range scans. Therefore, optimizing joins is crucial for achieving good performance when working with related data across multiple tables. HBase distributed nature makes join performance a more significant concern than it is in traditional relational databases. This section offers strategies for optimizing joins in Phoenix, including choosing the right join type, leveraging data locality, and, when appropriate, avoiding joins altogether through denormalization.

Phoenix supports several join types and related strategies, each with its own performance characteristics:

- **Star joins**. Optimized for scenarios where one large table (the **fact** table) is joined with multiple smaller **dimension** tables. Phoenix attempts to broadcast only the parts of the dimension tables required by the query to all relevant region servers (those containing data from the fact table in the join), allowing for efficient local joins. This is often the preferred join strategy in Phoenix when the data model and query patterns support it. Phoenix can typically determine if it is beneficial to execute a star join based on the data statistics available to it.
- **Broadcast joins**. Suited specifically for scenarios where one table is genuinely small (fits comfortably in memory on each region server) and is joined with one or more larger tables. Phoenix broadcasts the entire small table to all region servers. This is different from a star join where only relevant parts of the dimension tables are sent. This approach can yield excellent performance because the join operations happen locally on each region server, minimizing data transfer. Phoenix makes use of this optimization automatically when applicable.
- **Sort-merge joins**. While Phoenix supports sort-merge joins, they are generally less efficient than star or broadcast joins, especially in distributed environments like HBase. They involve sorting the input tables based on the join keys and then merging the sorted results. The performance bottleneck arises from the sorting overhead and the potential for data shuffling across the network. Phoenix typically resorts to sort-merge joins only when other more efficient join strategies cannot be applied. This might occur if the tables are very large and cannot be effectively broadcast, or if the join keys are not conducive to a hash-based join.

Possible strategies for optimizing the joins:

- **Denormalization**. Often the best optimization for joins is to avoid them entirely. Consider denormalizing data by embedding related information directly into the fact table or creating pre-joined views. For example, instead of joining `orders` and `customers` tables to retrieve customer information with each order, you could include relevant customer details (e.g. customer name, city) directly in the `orders` table. However, denormalization can lead to data redundancy and potential consistency issues, so carefully weigh the pros and cons based on your specific use case.
- **Data locality**. Place related tables on the same Region servers to minimize network traffic during joins. This can be achieved through co-location strategies in HBase.
- **Choosing the correct join order**. The order in which tables are joined can significantly impact performance. Generally, it is best to start with the table that has the most restrictive filter and join it with smaller tables first. Filtering early reduces the amount of data that needs to be processed in subsequent join operations.
- **Filtering before joining**. Apply the `WHERE` clauses to filter data from each table before the `JOIN` operation. Reducing the size of the tables being joined improves join performance.
- **Broadcast joins for small tables**. For joins involving a small dimension table and a large fact table, use a broadcast join. Broadcast joins copy the smaller table to all region servers, allowing for local joins with the larger table. Phoenix may automatically choose a broadcast join when it detects a small table. You can often influence this decision using hints.
- **Leverage the star join optimization**. If your data model conforms to a star schema (one large fact table joined with multiple smaller dimension tables), take advantage of Phoenix star join optimization.

Hints

Hints are directives that you can add to your SQL queries to influence Phoenix query execution plan. They provide a way to guide the optimizer and potentially improve performance when the default execution plan is not optimal. While Phoenix automatically optimizes queries, hints allow you to override its decisions in specific situations. Use hints carefully, as they can make your queries less portable and harder to maintain.

Hints are placed after the `SELECT`, `INSERT`, `UPDATE`, or `DELETE` keywords and have the following syntax:

```
SELECT /*+ hint1, hint2, ... */ column1, column2 FROM table_name WHERE condition;
```

Considerations when using hints:

- **Use with caution**. Hints override the default query optimizer, potentially masking underlying performance issues or leading to sub-optimal plans if used incorrectly.
- **Test thoroughly**. Always test the performance of your queries with and without hints to ensure that the hint is actually improving performance.
- **Retain maintainability**. Hints can make your queries less portable and harder to maintain. Document the reasons for using a hint and monitor its effectiveness over time.
- **Keep in mind the optimizer changes**. Phoenix query optimizer might change in future versions, rendering your hints ineffective or even detrimental. Regularly review and update your hints as needed.
- **Understand the execution plan**. Use the `EXPLAIN PLAN` function to understand how your hints are affecting the query execution plan.

Query server

The Phoenix Query Server (PQS) provides a way to access Phoenix from clients that do not have direct access to the HBase cluster. It acts as a proxy, allowing clients to submit SQL queries over a thin JDBC driver, abstracting away the complexities of interacting directly with HBase. This approach is beneficial in scenarios where security restrictions, network configurations, or client-side dependencies prevent direct connections to the HBase cluster.

The Phoenix Query Server follows a client-server architecture:

1. **Client**. The client application uses a thin JDBC driver to connect to the PQS. This driver is often more lightweight than the standard Phoenix JDBC driver because it does not need to handle direct HBase connections.
2. **Phoenix Query Server**. The PQS is a standalone process that runs outside of the HBase cluster (although it is often co-located with HBase nodes). It receives SQL queries from clients, translates them into Phoenix queries, executes them against the HBase cluster, and returns the results to the client. The PQS handles tasks such as:
   - Authentication and authorization.
   - Query parsing and optimization (though it leverages Phoenix internal optimizer).
   - Connection pooling to HBase.
   - Result set management.
3. **HBase Cluster**. The HBase cluster stores the data and executes the actual data access operations. The PQS connects to the HBase cluster using the standard Phoenix JDBC driver.

Benefits of using the Phoenix Query Server:

- **Simplified client access**. Clients can access Phoenix data using a standard JDBC interface, without needing to install the full Phoenix client libraries or have direct access to the HBase cluster.
- **Security**. The PQS provides a central point for authentication and authorization, allowing you to control access to Phoenix data. You can restrict direct access to the HBase cluster, enhancing security.
- **Network isolation**. The PQS can be deployed in a DMZ or other secure network zone, isolating the HBase cluster from direct client access. This is crucial for security in many organizations.
- **Resource management**. The PQS can manage connections to the HBase cluster, preventing clients from overloading the cluster with too many connections. It enables connection pooling and resource management.
- **Language independence**. Clients can be written in any language that supports JDBC, providing flexibility in application development.
- **Thin JDBC driver**. The JDBC client is lightweight, making it easy to deploy on various platforms.

Logging in Phoenix

Vladimir Adamenkov

Contents

To Table of Contents

Overview

Set up query logging

View queries log

Set up the Phoenix Query Server logging

Overview

Apache Phoenix has an option to log the executed `SELECT` statements, saving them to the `SYSTEM.LOG` table. You can specify the logging verbosity level and the average number of queries logged. Also, you can add the Phoenix Query Server log information to the HBase logs.

Set up query logging

To turn on the query logging in Phoenix, go to the ADCM UI and do the following:

1. Go to ***Clusters*** and select your cluster.
2. Open the ***Services*** tab and select the ***HBase*** service.
3. Turn on the ***Show advanced*** switch and expand the ***Custom hbase-site.xml*** section.
4. Click ***Add property*** and enter `phoenix.log.level` for the field name. For the field value, enter one of the following:
    - `OFF` — no queries will be logged (default);
    - `INFO` — basic details of the queries will be logged;
    - `DEBUG` — verbose details of the queries will be logged;
    - `TRACE` — verbose details and the bind parameters of the queries will be logged.
5. Click ***Add property*** and enter `phoenix.log.sample.rate` for the field name. For the field value, specify the numeric representation of the query logging probability from `0.0` to `1.0`. For example, `0.5` means approximately half of the executed `SELECT` queries will be logged. Default value is `1.0` (all queries).
6. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.
7. If the Phoenix shell was loaded at this point, exit it and load it back again for the changes to take effect.
8. If necessary, grant the write priviliges for the `SYSTEM.LOG` table to users whose queries need to be logged using the `GRANT` command.

IMPORTANT

Everyone who has access to the `SYSTEM.LOG` table will be able to view the query log. Use the query logging at your own discretion.

View queries log

To view the query log, compose and execute a proper query against the `SYSTEM.LOG` table. Example:

```
SELECT * FROM SYSTEM.LOG;
```

Result:

```
+---------------+--------------+--------------------------------------+-----------+------------------+--------------+------------
----------------------------------------+
|  START_TIME   |  TABLE_NAME  |               QUERY_ID               | TENANT_ID |       USER       |  CLIENT_IP   |
QUERY                       |
+---------------+--------------+--------------------------------------+-----------+------------------+--------------+------------
----------------------------------------+
| 1743693276315 | WEAPON_TYPES | efa9397d-56b4-4766-940f-b2dca9ed88ac |           | rsa-key-20240513 | 10.92.40.104 | SELECT
ammo_type FROM weapon_types GROUP BY ammo_t |
| 1744124555613 |              | a380c6f4-cffc-4fd8-b35c-1ef02aa52632 |           | rsa-key-20240513 | 10.92.40.104 | SELECT
ammo_type FROM weapon_types GROUP BY ammo_t |
+---------------+--------------+--------------------------------------+-----------+------------------+--------------+------------
----------------------------------------+
```

The `SYSTEM.LOG` table has far more columns than a terminal window of any width can display. You may use the `!outputformat vertical` command before executing the query and/or narrow its conditions down.

To clear the `SYSTEM.LOG` table, use the `DELETE` statement with a `WHERE` clause. For example, to remove all records from this table, use the following query:

```
DELETE FROM SYSTEM.LOG WHERE START_TIME > 0;
```

Set up the Phoenix Query Server logging

To set up the Phoenix Query Server logging, go to the ADCM UI and do the following:

1. Go to ***Clusters*** and select your cluster.
2. Open the ***Services*** tab and select the ***HBase*** service.
3. Turn on the ***Show advanced*** switch and select the ***Custom log4j.properties*** parameter.
4. Add the following lines to the parameter contents:

```
log4j.logger.org.apache.phoenix.queryserver=DEBUG
log4j.logger.org.apache.phoenix.queryserver.server=DEBUG
log4j.logger.org.apache.phoenix=DEBUG
log4j.logger.org.eclipse.jetty.io.AbstractEndPoint=DEBUG
```

5. Change the `log4j.threshold` value to `DEBUG` and click ***Apply***.
6. Save the configuration by clicking ***Save*** → ***Create*** and restart the service by clicking ***Actions*** → ***Reconfig and graceful restart***.

Connect to Solr via CLI

Konstantin Alpashkin

Contents

To Table of Contents

- Control commands: start, stop, restart
- System information commands
  - Command parameters
  - Examples
- Collection & core commands
  - Command parameters
  - Examples
  - Cores
- Authentication
  - Command parameters
  - Examples
- ZooKeeper commands
  - Command parameters
  - Examples
- Data export commands
  - Command parameters
  - Examples

Solr Server components installed on ADH hosts provide the */usr/lib/solr/bin/solr* script that allows you to run CLI commands to manage your Solr cluster. Although it is recommended to control the activity of your Solr cluster via ADCM, some of the Solr CLI commands can be useful to perform actions that are unavailable in the Solr Admin UI. These commands are highlighted further in the article.

The general usage of the */usr/lib/solr/bin/solr* script is as follows:

```
$ /usr/lib/solr/bin/solr <COMMAND> [<PARAMETERS>]
```

The detailed information about all the supported `COMMAND` and `PARAMETERS` values is available in the Solr reference documentation. To get help information for a particular command, use the following syntax:

```
$ /usr/lib/solr/bin/solr <COMMAND> -help
```

For more verbose command output, append the `-V` flag to your commands.

> **NOTE**
>
> As an ADH service, Solr always runs in the SolrCloud mode. Thus, some commands related to the standalone mode may have no effect at all.

Control commands: start, stop, restart

Although the */usr/lib/solr/bin/solr* script accepts commands like `start`, `stop`, and `restart`, it is highly recommended to use ADCM to perform these actions. Under the hood, ADCM invokes these commands to control Solr processes. However, ADCM additionally takes care about running these commands properly. For example, setting environment variables, submitting commands with sufficient user permissions, handling errors, etc.

Running `start`, `stop`, and `restart` commands via */usr/lib/solr/bin/solr* may lead to Solr startup issues and the malfunction of the entire Solr cluster.

System information commands

Solr provides several commands that might be useful for getting status information and diagnosing your Solr cluster. These commands are shown below:

```
$ /usr/lib/solr/bin/solr version
$ /usr/lib/solr/bin/solr status
$ /usr/lib/solr/bin/solr healthcheck [<PARAMETERS>]
```

Command parameters

Command: healthcheck

Parameter	Description
-c <collection_name>	The name of the collection
-z <zk_connection_string>	Optional. The ZooKeeper connection string. Can be omitted unless you have modified your ZooKeeper connection string

Examples

Get Solr version

```
$ /usr/lib/solr/bin/solr version
```

Sample output:

```
8.11.2
```

Get Solr node status

```
$ /usr/lib/solr/bin/solr status
```

Sample output:

```
Found 1 Solr nodes:

Solr process 23658 running on port 8983
{
  "solr_home":"/srv/solr/server",
  "version":"8.11.2 3cb85f40e394dcfb50fe77310908cce385381ba2 - jenkins - 2024-03-20 17:52:09",
  "startTime":"2024-05-23T22:52:07.271Z",
  "uptime":"0 days, 0 hours, 0 minutes, 6 seconds",
  "memory":"105.9 MB (%21.6) of 491 MB",
  "cloud":{
    "ZooKeeper":"ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.
internal:2181/Arenadata.Hadoop-7.solr.server",
    "liveNodes":"3",
    "collections":"0"}}
```

Run a collection healthcheck

```
$ /usr/lib/solr/bin/solr healthcheck -c test_collection
```

Sample output:

```
{
  "collection":"test_collection",
  "status":"healthy",
  "numDocs":0,
  "numShards":2,
  "shards":[
    {
      "shard":"shard1",
      "status":"healthy",
      "replicas":[
        {
          "name":"core_node3",
          "url":"http://ka-adh-3.ru-central1.internal:8983/solr/test_collection_shard1_replica_n1/",
          "numDocs":0,
          "status":"active",
          "uptime":"0 days, 11 hours, 21 minutes, 7 seconds",
          "memory":"143.6 MB (%29.4) of 489 MB"},
        {
          "name":"core_node5",
          "url":"http://ka-adh-1.ru-central1.internal:8983/solr/test_collection_shard1_replica_n2/",
          "numDocs":0,
          "status":"active",
          "uptime":"0 days, 11 hours, 21 minutes, 7 seconds",
          "memory":"67.9 MB (%14) of 485 MB",
          "leader":true},
        {
          "name":"core_node7",
          "url":"http://ka-adh-2.ru-central1.internal:8983/solr/test_collection_shard1_replica_n4/",
          "numDocs":0,
          "status":"active",
          "uptime":"0 days, 11 hours, 21 minutes, 7 seconds",
          "memory":"63 MB (%13) of 485 MB"}]},
    {
      "shard":"shard2",
      "status":"healthy",
      "replicas":[
        {
          "name":"core_node9",
          "url":"http://ka-adh-3.ru-central1.internal:8983/solr/test_collection_shard2_replica_n6/",
          "numDocs":0,
          "status":"active",
          "uptime":"0 days, 11 hours, 21 minutes, 7 seconds",
          "memory":"144.8 MB (%29.6) of 489 MB"},
        {
          "name":"core_node11",
          "url":"http://ka-adh-1.ru-central1.internal:8983/solr/test_collection_shard2_replica_n8/",
          "numDocs":0,
          "status":"active",
          "uptime":"0 days, 11 hours, 21 minutes, 7 seconds",
          "memory":"69.3 MB (%14.3) of 485 MB"},
        {
          "name":"core_node12",
          "url":"http://ka-adh-2.ru-central1.internal:8983/solr/test_collection_shard2_replica_n10/",
          "numDocs":0,
          "status":"active",
          "uptime":"0 days, 11 hours, 21 minutes, 7 seconds",
          "memory":"65.1 MB (%13.4) of 485 MB",
          "leader":true}]}]}
```

Collection & core commands

You can use the */usr/lib/solr/bin/solr* script to create and delete Solr collections.

The usage is as follows:

```
$ /usr/lib/solr/bin/solr create [<PARAMETERS>]
$ /usr/lib/solr/bin/solr create_collection [<PARAMETERS>]
$ /usr/lib/solr/bin/solr delete [<PARAMETERS>]
```

> **NOTE**
>
> In ADH, the Solr's `create` and `create_collection` commands are equivalent. Running `create` detects the current Solr mode (which is always SolrCloud in ADH) and invokes the `create_collection` command with the same set of parameters.

Command parameters

Commands: create/create_collection

Parameter	Description
-c <collection>	A name for the new collection
-d <conf_dir>	Specifies a configuration directory (configset) to use when creating a collection. If not specified, the */usr/lib/solr/server/solr/configsets/_default* configset is used
-n <config_name>	The name used to upload the configuration directory to ZooKeeper. By default, the collection name is used
-s <shards>, -shards <shards>	The number of shards to split the collection, defaults to `1`
-rf <replicas>, -replicationFactor <replicas>	The number of copies of each document in the collection. Defaults to `1`, assuming no replication

Command: delete

Parameter	Description
-c <collection>	The name of the collection to delete
-deleteConfig <true|false>	Defines whether to delete the configuration directory from ZooKeeper. The default behavior is `true`. If a ZooKeeper configuration directory is already used by another collection, it will not be deleted even if you pass `-deleteConfig true`

Examples

Create a basic collection

```
$ /usr/lib/solr/bin/solr create_collection -c test_collection1
```

Sample output:

```
Created collection 'test_collection1' with 1 shard(s), 1 replica(s) with config-set 'test_collection1'
```

Create a multi-sharded, replicated collection

```
$ /usr/lib/solr/bin/solr create_collection -c test_collection -shards 2 -replicationFactor 3 -V
```

Sample output:

```
Connecting to ZooKeeper at ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.
internal:2181/Arenadata.Hadoop-8.solr.server ...
INFO  - 2024-05-28 08:23:31.128; org.apache.solr.common.cloud.ConnectionManager; Waiting for client to connect to ZooKeeper
INFO  - 2024-05-28 08:23:31.145; org.apache.solr.common.cloud.ConnectionManager; zkClient has connected
INFO  - 2024-05-28 08:23:31.145; org.apache.solr.common.cloud.ConnectionManager; Client is connected to ZooKeeper
INFO  - 2024-05-28 08:23:31.158; org.apache.solr.common.cloud.ZkStateReader; Updated live nodes from ZooKeeper... (0) -> (3)
INFO  - 2024-05-28 08:23:31.269; org.apache.solr.client.solrj.impl.ZkClientClusterStateProvider; Cluster at ka-adh-1.ru-
central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-8.solr.server
ready
Uploading /usr/lib/solr/server/solr/configsets/_default/conf for config test_collection to ZooKeeper at ka-adh-1.ru-central1.
internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-8.solr.server

Creating new collection 'test_collection' using command:
http://ka-adh-1.ru-central1.internal:8983/solr/admin/collections?
action=CREATE&name=test_collection&numShards=2&replicationFactor=3&maxShardsPerNode=-1&collection.configName=test_collection

{
  "responseHeader":{
    "status":0,
    "QTime":2325},
  "success":{
    "ka-adh-1.ru-central1.internal:8983_solr":{
      "responseHeader":{
        "status":0,
        "QTime":1199},
      "core":"test_collection_shard2_replica_n8"},
    "ka-adh-3.ru-central1.internal:8983_solr":{
      "responseHeader":{
        "status":0,
        "QTime":1610},
      "core":"test_collection_shard1_replica_n1"},
    "ka-adh-2.ru-central1.internal:8983_solr":{
      "responseHeader":{
        "status":0,
        "QTime":1822},
      "core":"test_collection_shard2_replica_n10"}}}
```

Delete a collection

```
$ /usr/lib/solr/bin/solr delete -c test_collection1
```

Sample output:

```
{
  "responseHeader":{
    "status":0,
    "QTime":297},
  "success":{
    "ka-adh-1.ru-central1.internal:8983_solr":{"responseHeader":{
        "status":0,
        "QTime":50}},
    "ka-adh-3.ru-central1.internal:8983_solr":{"responseHeader":{
        "status":0,
        "QTime":119}},
    "ka-adh-2.ru-central1.internal:8983_solr":{"responseHeader":{
        "status":0,
        "QTime":130}}}}


Deleted collection 'test_collection1' using command:
http://ka-adh-1.ru-central1.internal:8983/solr/admin/collections?action=DELETE&name=test_collection1
```

Cores

Since ADH Solr Server components always run in the SolrCloud mode, Solr manages the data in the form of collections (a single search index distributed across multiple cluster nodes) rather than cores. For this reason, the core management commands have no effect and are beyond the scope of this article.

Authentication

Using the */usr/lib/solr/bin/solr* script, you can enable and disable Solr authentication.

The usage is as follows:

```
$ /usr/lib/solr/bin/solr auth enable [<PARAMETERS>]
$ /usr/lib/solr/bin/solr auth disable
```

Command parameters

Command: auth enable

Parameter	Description
-authConfDir <dir>	Specifies a directory to store configuration files related to authentication
-blockUnknown <true|false>	Blocks access to all Solr endpoints for unauthenticated users
-credentials <username:password>	The credentials pair in the `username:password` format
-prompt <true|false>	Prompts to provide the credentials. Either the `-prompt` or `-credentials` option must be provided
-type <type>	Defines the authentication mechanism (`basicAuth` or `kerberos`) to enable. Defaults to `basicAuth`
-updateIncludeFileOnly <true|false>	Updates only the *solr.in.sh* file without actually enabling or disabling the authentication
-z <zkString>	The ZooKeeper connection string. Can be omitted unless you have modified your ZooKeeper connection string
-d <dir>	Specifies the Solr server directory (defaults to */usr/lib/solr/server/*). Use this option only if you have customized your Solr home directory path
-s <dir>	Specifies the Solr home directory to store credentials and authentication configuration files (for example, *basicAuth.conf*)

Examples

Enable basic authentication

Before running the following example command, ensure that a security.json file is uploaded to ZooKeeper. You can upload the file using ZooKeeper CLI or using the */usr/lib/solr/server/scripts/cloud-scripts/zkcli.sh* script.

```
$ /usr/lib/solr/bin/solr auth enable \
    -type basicAuth \
    -prompt true \
    -z ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-
6.solr.server \
    -blockUnknown true
```

Sample output:

```
Security is enabled. You can disable it with 'bin/solr auth disable'. Existing security.json:
{"authentication":{"blockUnknown":true,"class":"solr.BasicAuthPlugin","realm":"My Solr
users","forwardCredentials":false},"authorization":{"class":"solr.RuleBasedAuthorizationPlugin","permissions":
[{"name":"security-edit","role":"admin"}],"user-role":{"solr":"admin"}}}
```

Disable authentication

```
$ /usr/lib/solr/bin/solr auth disable
```

Sample output:

```
Security has been disabled.
Please remove any SOLR_AUTH_TYPE or SOLR_AUTHENTICATION_OPTS configuration from solr.in.sh/solr.in.cmd.
```

ZooKeeper commands

The */usr/lib/solr/bin/solr* script supports commands to interact with ZooKeeper. For example, the `zk upconfig` and `zk downconfig` commands are used to upload/download Solr configsets from ZooKeeper. The usage is as follows:

```
$ /usr/lib/solr/bin/solr zk upconfig [<PARAMETERS>]
$ /usr/lib/solr/bin/solr zk downconfig [<PARAMETERS>]
```

Command parameters

Commands: zk upconfig/zk downconfig

Parameter	Description
-n <name>	A name for the configset that has to be uploaded or downloaded from ZooKeeper. All the uploaded configsets are available in the Solr Admin UI. Using an existing name overwrites the configset in ZooKeeper
-d <configset_dir>	The path to the configset directory to upload/download. A configset directory must contain the *conf* directory with the *solrconfig.xml* file inside
-z <zk_host>	Optional. The ZooKeeper connection string. Can be omitted unless you have modified your ZooKeeper connection string

> **NOTE**
>
> Uploading a configset to ZooKeeper does not automatically make the changes effective. If you modify a configset already used by a collection, use the RELOAD collection command to reload any collection that uses this configset.

Also, there are several `zk …` commands that might be useful to perform common ZooKeeper operations. The usage is shown below:

```
$ /usr/lib/solr/bin/solr zk cp <src> <dst> [-r] [-z <zk_string>] ①
$ /usr/lib/solr/bin/solr zk rm <path> [-r] [-z <zk_string>] ②
$ /usr/lib/solr/bin/solr zk mv <src> <dst> [-z <zk_string>] ③
$ /usr/lib/solr/bin/solr zk ls <path> [-r] [-z <zk_string>] ④
$ /usr/lib/solr/bin/solr zk mkroot <path> [-z <zk_string>] ⑤
```

① Transfers data between the local file system and ZooKeeper Znodes. The prefix `zk:` assumes a ZooKeeper location, and `file:` indicates the local file system. At least one of `<src>` or `<dest>` must be prefixed with `zk:`.

② Removes a Znode from ZooKeeper. Since `<path>` assumes a ZooKeeper node, no `zk:` prefix is necessary.

③ Moves (renames) a Znode.

④ Lists the children of a Znode.

⑤ Creates a new Znode. Intermediate Znodes are created, if necessary.

Examples

Download a configset from ZooKepper

```
$ /usr/lib/solr/bin/solr zk downconfig -n test_collection -d ~/my_test_collection_config
```

Sample output:

```
Downloading configset test_collection from ZooKeeper at ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.
internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-8.solr.server to directory
/home/admin/my_test_collection_config/conf
```

Upload a directory to ZooKeeper

```
$ /usr/lib/solr/bin/solr zk cp -r file:/home/admin/test_dir/ zk:/Arenadata.Hadoop-8.solr.server/test_upload
```

Sample output:

```
Copying from 'file:/home/admin/test_dir/' to 'zk:/Arenadata.Hadoop-8.solr.server/test_upload'. ZooKeeper at ka-adh-1.ru-
central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-8.solr.server
```

Remove a Znode with descendants

```
$ /usr/lib/solr/bin/solr zk rm -r zk:/Arenadata.Hadoop-8.solr.server/test_upload
```

Sample output:

```
Removing ZooKeeper node /Arenadata.Hadoop-8.solr.server/test_upload from ZooKeeper at ka-adh-1.ru-central1.internal:2181,ka-
adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-8.solr.server recurse: true
```

Data export commands

Using the `export` command, you can export documents from a Solr collection in the JSON or Javabin format. The usage is as follows:

```
$ /usr/lib/solr/bin/solr export [<PARAMETERS>]
```

Command parameters

Command: export

Parameter	Description
-fields <f1,f2,fn>	A comma-separated list of fields to export. By default, all fields are exported
-format <fmt>	Sets the output format for exported documents (JSON or javabin), defaulting to JSON
-limit <i>	The maximum number of documents to export. The default value is `100`, use `-1` to export all documents
-out	Specifies the file name to save exported data. By default, the `<collection-name>.<format>` template is used
-url <collection_url>	The URL address pointing to the collection

Examples

Export all data from a collection

```
$ sudo bin/solr export -url http://ka-adh-1.ru-central1.internal:8983/solr/test_collection
```

Sample output:

```
NO: of shards : 2

Export complete for : test_collection_shard1_replica_n2, docs : 0

Export complete for : test_collection_shard2_replica_n8, docs : 1

Total Docs exported: 0. Time taken: 0secs
```

Running this command creates the *test_collection.json* file in the local file system. The file stores the documents from the requested collection, for example:

```
{"key1":"foo","key2":"bar","id":"71b51c87-142c-45a6-8ea0-82ae924fd3c4"}
```

Export only specific fields to a custom file

```
$ sudo bin/solr export -url http://ka-adh-1.ru-central1.internal:8983/solr/test_collection -fields "key1"
```

Sample output:

```
NO: of shards : 2
.
Export complete for : test_collection_shard1_replica_n2, docs : 0

Export complete for : test_collection_shard2_replica_n8, docs : 1

Total Docs exported: 0. Time taken: 0secs
```

Running this command creates the *test_export_file.json* file in the local file system. The file stores the documents containing only the requested fields, for example:

```
{"key1":"foo"}
```

Solr indexing overview

Konstantin Alpashkin

Contents

To Table of Contents

Post tool
Index data
Update indexed data
Delete indexed data
Search data
- Solr query parsers
- Search examples

This article describes Solr indexing — the process of adding new content to a Solr index to make the content searchable, as well as modifying and deleting existing content.

There are several ways on how you can feed data to Solr for indexing:

- Upload XML/JSON/CSV files within an HTTP request body to a Solr server's REST endpoint. Solr comes with a utility Post tool that facilitates uploading documents through the REST endpoint. This method is used in examples throughout the article.
- Use Solr Admin UI to upload documents through the web interface.
- Create a client application that interacts with Solr over HTTP using Client APIs. For example, SolrJ library for JVM-based applications.
- Use frameworks like Solr Cell developed for efficient ingesting of data into Solr.

Regardless of the method used for delivering data to Solr, there is a common structure for all data submitted to a Solr index: this must be a document containing one or more named fields, each storing a content payload (which may be empty). The structure of documents is defined by a schema file; however, Solr can also accept schemaless documents. Typically, one of the fields is chosen to be unique (analogous to a primary key in traditional databases), although the use of unique keys is not strictly required by Solr.

Apart from JSON, XML, and CSV formats, Solr can accept "rich" file formats, such as HTML, PDF, etc. In most cases, Solr automatically detects the file type, digests the data, and requires no manual actions on specifying the file type.

For more information about Solr indexing concepts, see Solr documentation.

Post tool

Every Solr Server component deployed by ADH includes a utility tool that facilitates POSTing documents to a Solr server. You can use this tool by running the */usr/lib/solr/bin/post* script available on hosts with the Solr Server component.

The script usage syntax is as follows:

```
$ /usr/lib/solr/post -c <collection_name> [OPTIONS] <file|directory|url|-d ["...",...]>
```

The supported `OPTIONS` flags are described in the table below.

-c <collection_name>	Solr collection to upload data for indexing. Either a collection name or a URL must be specified
-url <URL>	Solr server's endpoint URL. If not specified, the tool generates the URL automatically, for example *http://localhost:<solr_port>/solr/<collection_name>/update*. Parameters embedded in the URL override the values provided as `-c`, `-host`, and `-port` options
-host <host_name>	The host name or IP address of the Solr server
-port <p>, -p	The port number used for the connection
-user <user:pass>, -u <user:pass>	Basic authentication credentials for requests
-type <content-type>	Sets the Content-Type header. Defaults to `application/xml`
-filetypes <type1,...>	Specifies the file types accepted by Solr. By default, the accepted file types are: xml, json, csv, pdf, doc, docx, ppt, pptx, xls, xlsx, odt, odp, ods, ott, otp, ots, rtf, htm, html, txt, log
-out <yes|no>	Outputs Solr's response to the console

Below are several examples of using the */usr/lib/solr/bin/post* tool.

```
/usr/lib/solr/bin/post -c demo_index_collection transactions.json
/usr/lib/solr/bin/post -c demo_index_collection test-*.xml
/usr/lib/solr/bin/post -c demo_index_collection -out yes test.csv
/usr/lib/solr/bin/post -c demo_index_collection ~/test/solr-docs
```

Index data

The following scenario highlights major steps of the Solr indexing process. The steps show how to index a simple JSON document using the Post tool.

1. Connect to an ADH host with the Solr Server component.
2. Create a test Solr collection using the following command:

   ```
   $ /usr/lib/solr/bin/solr create -c demo_index_collection -s 2 -rf 2
   ```

   Output:

   ```
   Created collection 'demo_index_collection' with 2 shard(s), 2 replica(s) with config-set 'demo_index_collection'
   ```

   This creates a new collection (`demo_index_collection`) with two shards and the replication factor set to `2`. The collection is created based on the `_default` configset that provides schemaless support.

   TIP

   You can also use Solr web UI to create a new collection.

3. Create a sample JSON document, as shown below. You can create a similar document using XML or CSV formats. Solr will automatically detect the file type and digest the document accordingly.

```
{
  "id": 1,
  "txn_id": "1",
  "acc_id": 1001,
  "txn_value": 75.0,
  "txn_date": "2024-01-02",
  "comment": "The first transaction."
}
```

   In this sample document, the `id` field acts as a primary key to uniquely identify the document among other indexed documents. Using this ID, you can update or delete the document in the future. The `id` field is chosen because it is specified in the default schema file (the `uniqueKey` definition). If you remove the `id` field from the document above, Solr will auto-generate one for you; however, you will have to track Solr-generated IDs to be able to work with indexed documents in the future.

4. Using the Post tool, submit the sample document to Solr with the following command:

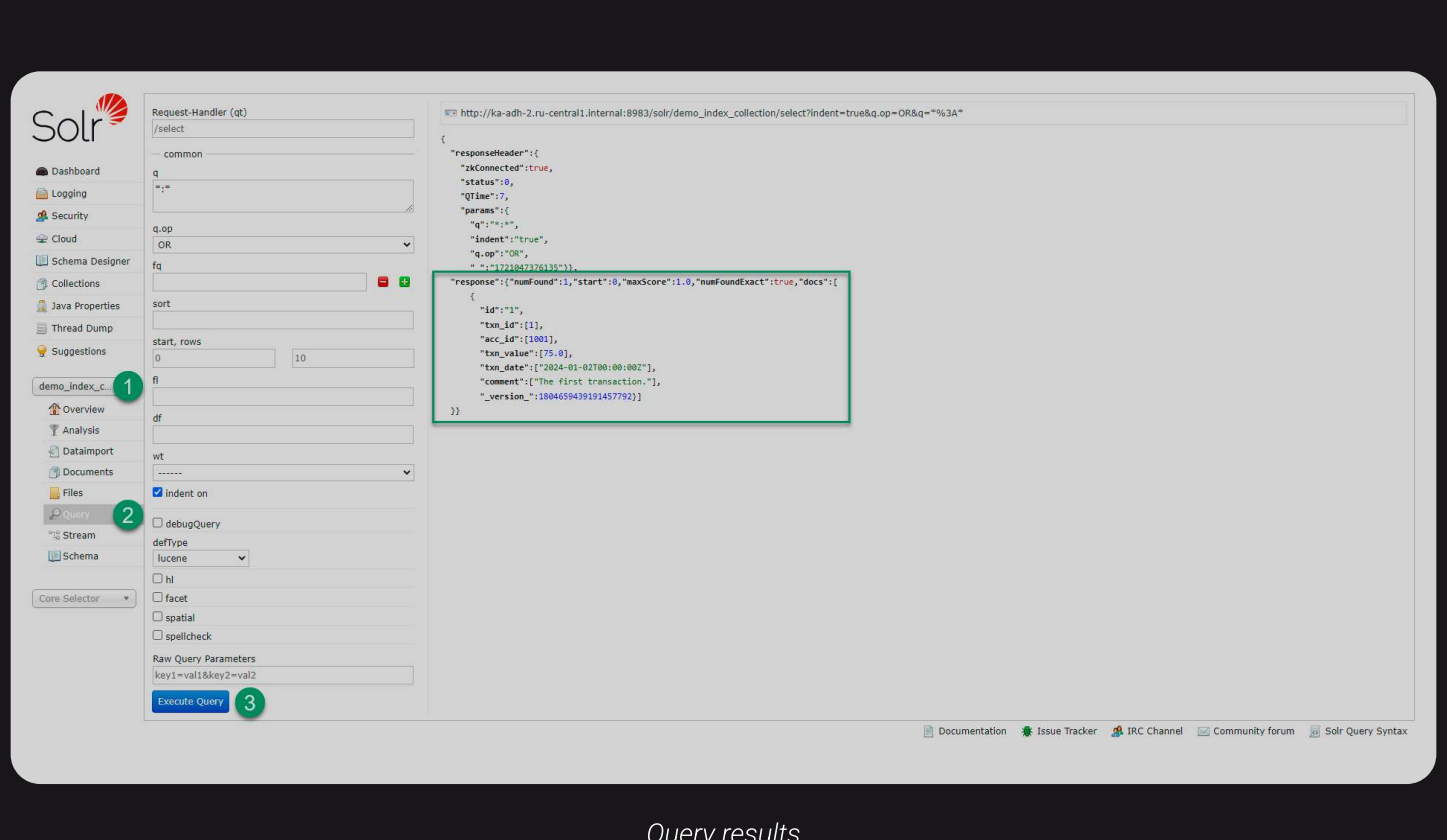
```
$ /usr/lib/solr/bin/post -c demo_index_collection transactions.json
```

   Running the script yields a similar output:

```
java -classpath /usr/lib/solr/dist/solr-core-8.11.2.jar -Dauto=yes -Dc=demo_index_collection -Ddata=files org.apache.solr.util.
SimplePostTool transactions.json
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/demo_index_collection/update...
Entering auto mode. File endings considered are
xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,html,txt,log
POSTing file transactions.json (application/json) to [base]/json/docs
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/demo_index_collection/update...
Time spent: 0:00:01.341
```

To verify that Solr has indexed the document successfully, follow the steps:

1. In Solr Admin UI, select the required collection (`demo_index_collection`).
2. Go to the collection's ***Query*** page.
3. Click ***Execute Query*** to run a query with default parameters. The query results are displayed as shown in the image below.



Query results

Update indexed data

To update an indexed document, you have to re-submit the modified document version to Solr and ensure that the modified document's unique key matches the document you want to update. If you omit or provide an incorrect unique key, Solr will create a new document instead of updating the document you need. The default Solr schema (the one that belongs to `_default` configset) uses the `id` field as a unique key.

The steps required to update the sample document are described below:

1. Create an updated version of the document that is already indexed by Solr. For example, the *transactions_updated.json* contents:

```
{
  "id": 1,
  "txn_id": "1",
  "acc_id": 1001,
  "txn_value": 75.0,
  "txn_date": "2024-01-02",
  "comment": "Updated transaction description."
}
```

2. Submit the updated document to Solr using the Post tool:

```
$ /usr/lib/solr/bin/post -c demo_index_collection transactions_updated.json
```

   The output:

```
java -classpath /usr/lib/solr/dist/solr-core-8.11.2.jar -Dauto=yes -Dc=demo_index_collection -Ddata=files org.apache.solr.util.
SimplePostTool transactions_updated.json
SimplePostTool version 5.0.0
Posting files to [base] url http://localhost:8983/solr/demo_index_collection/update...
Entering auto mode. File endings considered are
xml,json,jsonl,csv,pdf,doc,docx,ppt,pptx,xls,xlsx,odt,odp,ods,ott,otp,ots,rtf,htm,html,txt,log
POSTing file transactions_updated.json (application/json) to [base]/json/docs
1 files indexed.
COMMITting Solr index changes to http://localhost:8983/solr/demo_index_collection/update...
Time spent: 0:00:00.307
```

3. Using Solr UI, query the updated document using the `txn_id:1` query parameter (`q`). This is equivalent to SQL's `SELECT * FROM <collection_name> WHERE txn_id=1;`. The query response reflects updated document fields, as shown in the example:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":8,
    "params":{
      "q":"txn_id:1",
      "indent":"true",
      "q.op":"OR",
      "_":"1723478346726"}},
  "response":{"numFound":1,"start":0,"maxScore":1.0,"numFoundExact":true,"docs":[
      {
        "id":"1",
        "txn_id":[1],
        "acc_id":[1001],
        "txn_value":[75.0],
        "txn_date":["2024-01-02T00:00:00Z"],
        "comment":["Updated transaction description."],
        "_version_":1807198162248531968}]
  }}
```

Delete indexed data

You can delete a document from a Solr index by POSTing a delete command to the update URL and specifying the document's unique key or a query to match multiple documents. For example, to delete a document with the `id=1` field, use the command:

```
$ /usr/lib/solr/bin/post -c demo_index_collection -d "<delete><id>1</id></delete>"
```

To delete a document using Solr query syntax, use the syntax, as shown below:

```
$ /usr/lib/solr/bin/post -c demo_index_collection -d "<delete><query>txn_id:1</query></delete>"
```

The command that deletes all the documents from a collection looks as follows:

```
$ /usr/lib/solr/bin/post -c demo_index_collection -d "<delete><query>*:*</query></delete>"
```

Search data

Solr search queries are regular HTTP requests where search parameters are specified within the URL query string. To submit a search request to Solr, you can use HTTP clients like curl, wget, Postman, or use the Post tool. A sample URL to submit a search request has the following view:

```
http://ka-adh-1.ru-central1.internal:8983/solr/demo_index_collection/select?indent=true&q.op=AND&q=*%3A*
```

Where `ka-adh-1.ru-central1.internal` is name of the host with a running Solr Server component.

The response from Solr for such a query may look as follows:

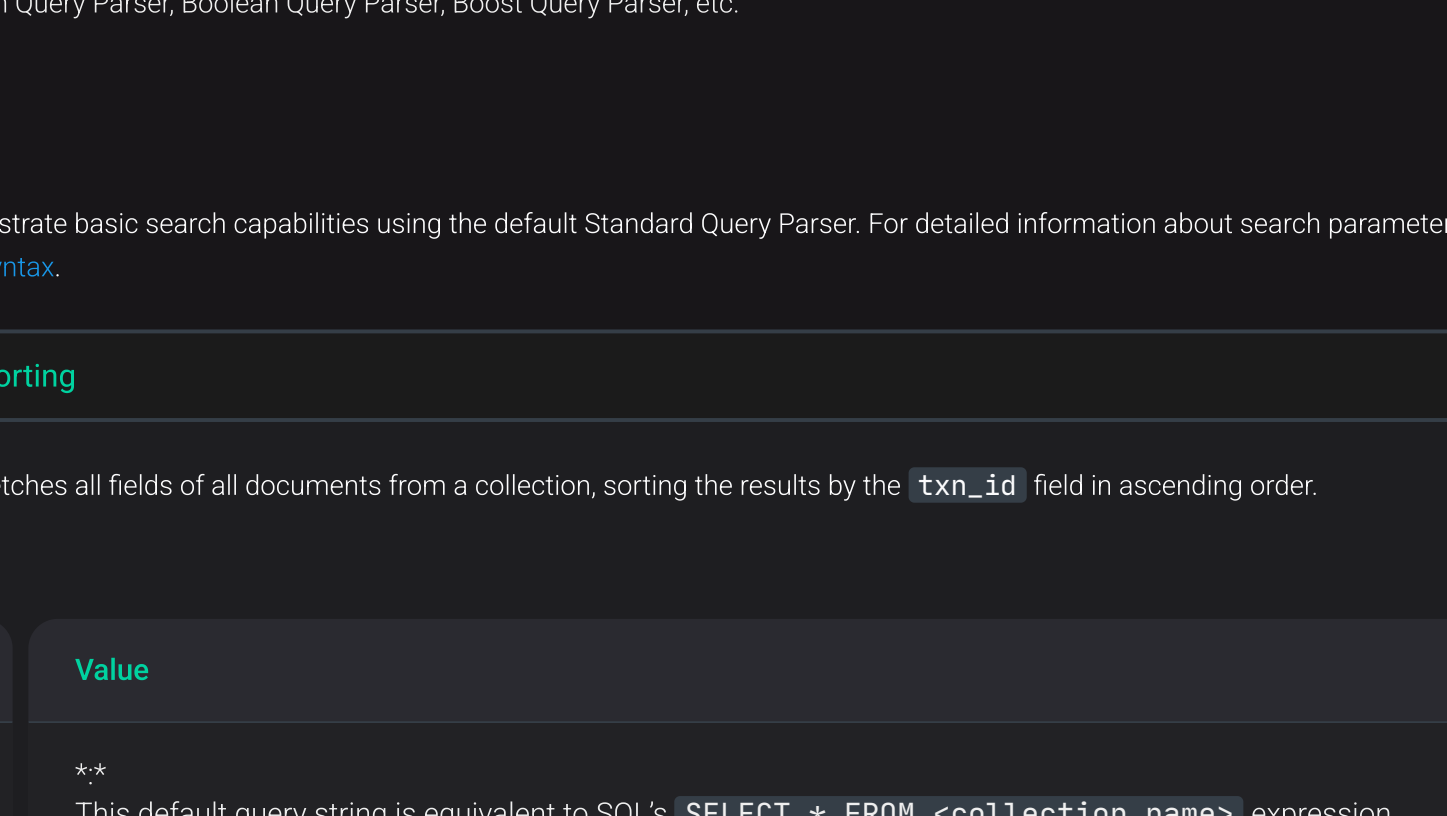
```
{
    "responseHeader": {
        "zkConnected": true,
        "status": 0,
        "QTime": 6,
        "params": {
            "q": "*:*",
            "indent": "true"
        }
    },
    "response": { ①
        "numFound": 1,
        "start": 0,
        "maxScore": 1.0,
        "numFoundExact": true,
        "docs": [ ②
            {
                "id": "1",
                "txn_id": [
                    1
                ],
                "acc_id": [
                    1001
                ],
                "txn_value": [
                    75.0
                ],
                "txn_date": [
                    "2024-01-02T00:00:00Z"
                ],
                "comment": [
                    "Updated transaction description."
                ],
                "_version_": 1807199238892814336
            }
        ]
    }
}
```

① The response section with query results.

② An array of documents that match the search query.

Also, you can use Solr Admin UI, which provides a query builder interface where you can construct and run search queries.

Using the query builder interface, you can enter query parameters in the UI fields and get the resulting query URL as shown in the image below.



Query URL

Solr query parsers

When a search query reaches Solr, it is processed by a request handler — a plugin that defines the request processing logic. Then, the request handler activates a query parser that interprets the parameters of the received query. Solr supports several query parsers listed below. Follow the links to get more information on using each parser.

- Standard Query Parser. The default query parser that supports robust and intuitive syntax. The major disadvantage is the parser's intolerance of syntax errors.
- DisMax. Allows processing simple phrases (without complex syntax) and searching for terms across several fields using a weighting mechanism based on the significance of each field.
- Extended DisMax (eDismax). An extended version of the DisMax Query Parser that provides features like partial escaping, advanced stopword handling, etc.
- Other parsers like Block Join Query Parser, Boolean Query Parser, Boost Query Parser, etc.

Search examples

The following examples demonstrate basic search capabilities using the default Standard Query Parser. For detailed information about search parameters supported by various Solr query parsers, see Solr query syntax.

Get all documents with sorting

The sample search request fetches all fields of all documents from a collection, sorting the results by the `txn_id` field in ascending order.

Request parameters

Request parameter	Value
q	*:* This default query string is equivalent to SQL's `SELECT * FROM <collection_name>` expression
sort	txn_id asc

Sample request URL:

```
http://ka-adh-1.ru-central1.internal:8983/solr/demo_index_collection/select?indent=true&q.op=OR&q=*%3A*&sort=txn_id%20asc
```

Sample response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":4,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "sort":"txn_id asc",
      "_":"1723495198651"}},
  "response":{"numFound":3,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"1",
        "txn_id":[1],
        "acc_id":[1001],
        "txn_value":[75.0],
        "txn_date":["2024-01-02T00:00:00Z"],
        "comment":["The first transaction."],
        "_version_":1807214831002976256},
      {
        "id":"3",
        "txn_id":[5],
        "acc_id":[1003],
        "txn_value":[105.0],
        "txn_date":["2024-02-02T00:00:00Z"],
        "comment":["Yet another transaction."],
        "_version_":1807215090454233088},
      {
        "id":"2",
        "txn_id":[256],
        "acc_id":[1002],
        "txn_value":[55.0],
        "txn_date":["2024-02-03T00:00:00Z"],
        "comment":["Another transaction."],
        "_version_":1807214995052691456}]
  }}
```

Get specific fields with filtering

The sample search request returns only specific fields (`fl=txn_id,acc_id,txn_value,comment`) of those documents, whose `txn_date` matches the specified date. Additionally, the result set gets filtered to retain only those records whose `comment` field contains the `first` substring.

Request parameters

Request parameter	Value
q	txn_date:"2024-01-02T00:00:00Z"
fl	txn_id,acc_id,txn_date,comment
fq	comment:\\"*first*\\"

Sample request URL:

```
http://ka-adh-1.ru-central1.internal:8983/solr/demo_index_collection/select?
fl=txn_id%2Cacc_id%2Ctxn_value%2Ccomment&fq=comment%3A%22*first1*%22&indent=true&q.op=OR&q=txn_date%3A%222024-01-
02T00%3A00%3A00Z%22
```

Sample response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":5,
    "params":{
      "q":"txn_date:\"2024-01-02T00:00:00Z\"",
      "indent":"true",
      "fl":"txn_id,acc_id,txn_value,comment",
      "q.op":"OR",
      "fq":"comment:\"*first*\"",
      "_":"1723492286745"}},
  "response":{"numFound":1,"start":0,"maxScore":1.0,"numFoundExact":true,"docs":[
      {
        "txn_id":[1],
        "acc_id":[1001],
        "txn_value":[75.0],
        "comment":["The first transaction."]}]
  }}
```

Queries combination

By using boolean operators, you can create complex query logic and combine multiple search conditions. For example, the sample request fetches documents whose `comment` field contains the `transaction` string and the `txn_id` field is `1`.

Request parameters

Request parameter	Value
q	comment:transaction AND txn_id:1

Sample request URL:

```
http://ka-adh-1.ru-central1.internal:8983/solr/demo_index_collection/select?indent=true&q.
op=OR&q=comment%3Atransaction%20AND%20txn_id%3A1
```

Sample response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":4,
    "params":{
      "q":"comment:transaction AND txn_id:1",
      "indent":"true",
      "q.op":"OR",
      "_":"1723492286745"}},
  "response":{"numFound":1,"start":0,"maxScore":1.1307646,"numFoundExact":true,"docs":[
      {
        "id":"1",
        "txn_id":[1],
        "acc_id":[1001],
        "txn_value":[75.0],
        "txn_date":["2024-01-02T00:00:00Z"],
        "comment":["The first transaction."],
        "_version_":1807214831002976256}]
  }}
```

Range search

The sample request returns documents whose `txn_value` falls in the range of 100-200.

Request parameters

Request parameter	Value
q	txn_value:[100 TO 200]

Sample request URL:

```
http://ka-adh-1.ru-central1.internal:8983/solr/demo_index_collection/select?indent=true&q.op=OR&q=txn_value%3A%5B100%20TO%20*%5D
```

Sample response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":9,
    "params":{
      "q":"txn_value:[100 TO 200]",
      "indent":"true",
      "q.op":"OR",
      "_":"1723492286745"}},
  "response":{"numFound":1,"start":0,"maxScore":1.0,"numFoundExact":true,"docs":[
      {
        "id":"3",
        "txn_id":[5],
        "acc_id":[1003],
        "txn_value":[105.0],
        "txn_date":["2024-02-02T00:00:00Z"],
        "comment":["Yet another transaction."],
        "_version_":1807215090454233088}]
  }}
```

Set response size and format

The sample request returns the first three documents that match the search query and instructs Solr to return the response as XML.

Request parameters

Request parameter	Value
q	*:*
rows	3
wt	xml

Sample request URL:

```
http://ka-adh-1.ru-central1.internal:8983/solr/demo_index_collection/select?indent=true&q.op=OR&q=*%3A*&rows=3&start=0&wt=xml
```

Sample response:

```
<?xml version="1.0" encoding="UTF-8"?>
<response>

<lst name="responseHeader">
  <bool name="zkConnected">true</bool>
  <int name="status">0</int>
  <int name="QTime">4</int>
  <lst name="params">
    <str name="q">*:*</str>
    <str name="indent">true</str>
    <str name="start">0</str>
    <str name="q.op">OR</str>
    <str name="rows">3</str>
    <str name="wt">xml</str>
    <str name="_">1723492286745</str>
  </lst>
</lst>
<result name="response" numFound="3" start="0" maxScore="1.0" numFoundExact="true">
  <doc>
    <str name="id">2</str>
    <arr name="txn_id">
      <long>256</long>
    </arr>
    <arr name="acc_id">
      <long>1002</long>
    </arr>
    <arr name="txn_value">
      <double>55.0</double>
    </arr>
    <arr name="txn_date">
      <date>2024-02-03T00:00:00Z</date>
    </arr>
    <arr name="comment">
      <str>Another transaction.</str>
    </arr>
    <long name="_version_">1807214995052691456</long></doc>
  <doc>
    <str name="id">3</str>
    <arr name="txn_id">
      <long>5</long>
    </arr>
    <arr name="acc_id">
      <long>1003</long>
    </arr>
    <arr name="txn_value">
      <double>105.0</double>
    </arr>
    <arr name="txn_date">
      <date>2024-02-02T00:00:00Z</date>
    </arr>
    <arr name="comment">
      <str>Yet another transaction.</str>
    </arr>
    <long name="_version_">1807215090454233088</long></doc>
  <doc>
    <str name="id">1</str>
    <arr name="txn_id">
      <long>1</long>
    </arr>
    <arr name="acc_id">
      <long>1001</long>
    </arr>
    <arr name="txn_value">
      <double>75.0</double>
    </arr>
    <arr name="txn_date">
      <date>2024-01-02T00:00:00Z</date>
    </arr>
    <arr name="comment">
      <str>The first transaction.</str>
    </arr>
    <long name="_version_">1807214831002976256</long></doc>
</result>
</response>
```

Index nested documents in Solr

Konstantin Alpashkin

Contents

To Table of Contents

Overview
  - Schema configuration
  - Anonymous nested documents

Index nested documents

Search nested data
  - Child document transformer
  - Block join query parsers

Overview

In Solr, nested documents are regular documents that were indexed as part of another (parent) document. For example, multiple comments related to the same blog post can be stored in Solr as nested documents within one post document. Below is a sample document that models such parent-child relationships:

```
[
  {
    "post_id":"blog_post0001",
    "nodeType":"post",
    "comments":[
      {
        "comment_id":"comment_1",
        "nodeType":"comment",
        "content":"foo bar"
      },
      {
        "comment_id":"comment_2",
        "nodeType":"comment",
        "content":"buzz"
      }
    ]
  }, {
    "post_id":"blog_post0002",
    "nodeType":"post",
    "comments":[
      {
        "comment_id":"comment_3",
        "nodeType":"comment",
        "content":"Some text"
      }
    ]
  }
]
```

Nesting documents may yield much faster queries since the parent-child relationships between the documents are already stored in the index and do not have to be computed at the query time. However, nested documents are less flexible than query-time joins since they impose strict rules that may be sometimes unacceptable for applications.

Nested documents can be indexed as XML, JSON, or SolrJ data formats. Regardless of the input format, each document must contain a field that identifies the document as either a parent or child. This field can have any name except for reserved field names defined in the schema. In the example above, it is the `nodeType` field that serves this purpose — the field carries either `post` or `comment` values. When searching for nested objects, this field is used as an input for the block join query parsers.

Schema configuration

Working with nested documents requires your schema to have an indexed field named `_root_` that looks as follows:

```
<field name="_root_" type="string" indexed="true" stored="false" docValues="false" />
```

Also, for proper operation of the child document transformer, your schema file must have the `_nest_path_` field, which is shown below:

```
<fieldType name="_nest_path_" class="solr.NestPathField" />
<field name="_nest_path_" type="_nest_path_" />
```

By default, these fields are present in the default schema file and will be inherited by schemas created from the default one.

Anonymous nested documents

Sometimes you may encounter the following syntax for nesting anonymous documents:

```
[
  {
    "post_id":"blog_post0001",
    "nodeType":"post",
    "_childDocuments_":[
      {
        "comment_id":"comment_1",
        "nodeType":"comment",
        "content":"foo bar"
      },
      {
        "comment_id":"comment_2",
        "nodeType":"comment",
        "content":"buzz"
      }
    ]
  }
]
```

Notice the `_childDocuments_` field holding an array of documents. This is an explicit indication to Solr that the array carries children elements. This approach was common in older versions of Solr and now is deprecated. However, it still works with single-level nested documents that do not contain any other nested fields apart from `_root_`.

Index nested documents

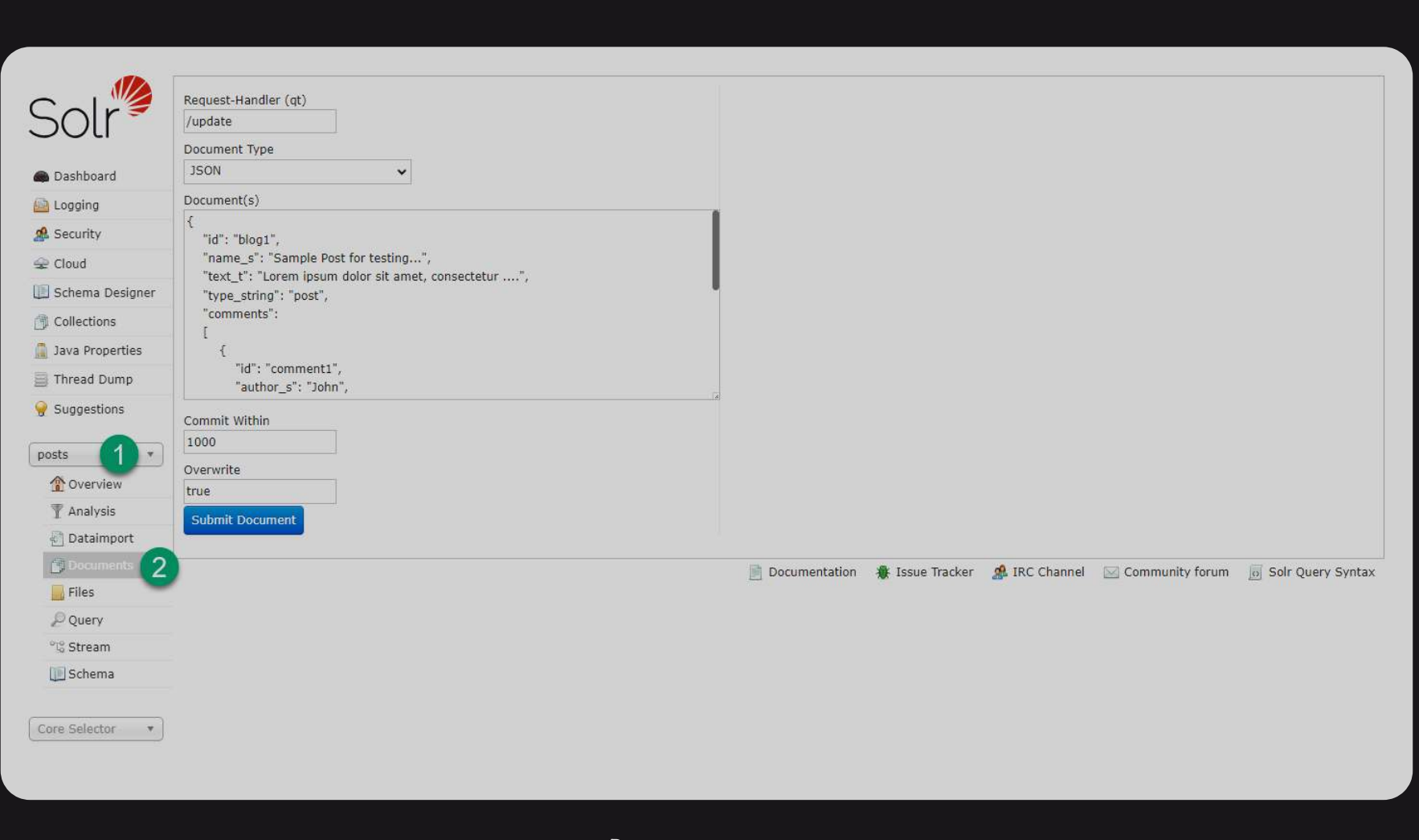
The following steps show how to index a Solr document with nested children.

1. Open Solr Admin UI. The up-to-date web UI link is available in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Solr*** → ***Info***).

2. Go to the ***Collections*** page and create a test collection with the following parameters.

Parameter	Description
name	posts
config set	_default * This inherits the default schema configuration with basic features to work with nested documents
numShards	1
replicationFactor	1

3. Select the newly created collection and open the collection's ***Documents*** page, as shown in the image.



Request-Handler (qt): /update

Document Type: JSON

Document(s):

```
{
  "id": "blog1",
  "name_s": "Sample Post for testing...",
  "text_t": "Lorem ipsum dolor sit amet, consectetur ....",
  "type_string": "post",
  "comments":
  [
    {
      "id": "comment1",
      "author_s": "John",
```

Commit Within: 1000

Overwrite: true

Documents page

4. Upload the following test documents to Solr by pasting their content into the ***Document(s)*** field and clicking ***Submit Document***.

post1.json

```
{
    "id": "post1",
    "name_s": "Sample Post for testing...",
    "text_t": "Lorem ipsum dolor sit amet, consectetur ....",
    "content_type": "post",
    "comments":
    [
        {
            "id": "comment1",
            "author_s": "John",
            "text_t": "Nice post!",
            "rated_i": 5,
            "content_type": "comment"
        },
        {
            "id": "comment2",
            "author_s": "Ann",
            "text_t": "Not interesting post...",
            "rated_i": 3,
            "content_type": "comment"
        }
    ]
}
```

post2.json

```
{
    "id": "post2",
    "name_s": "Another demo post...",
    "text_t": "Foo Bar Buzz ....",
    "content_type": "post",
    "comments":
    [
        {
            "id": "comment3",
            "author_s": "Luke",
            "text_t": "I like this post!",
            "rated_i": 4,
            "content_type": "comment"
        }
    ]
}
```

> **NOTE**
>
> Do not use the Post tool for uploading nested documents since the tool automatically flattens their JSON structure during the upload.

As an alternative to the UI, you can use curl to upload the sample documents to Solr. An example command is below.

curl upload command

```
curl -X POST -H 'Content-Type: application/json' 'http://ka-adh-1.ru-central1.internal:8983/solr/posts/update' --data-binary
'[
    {
        "id": "blog1",
        "name_s": "Sample Post for testing...",
        "text_t": "Lorem ipsum dolor sit amet, consectetur ....",
        "comments":
        [
            {
                "id": "comment1",
                "author_s": "John",
                "text_t": "Nice post!",
                "rated_i": 5
            },
            {
                "id": "comment2",
                "author_s": "Ann",
                "text_t": "Not interesting post...",
                "rated_i": 3
            }
        ]
    },
    {
        "id": "blog2",
        "name_s": "Another demo post...",
        "text_t": "Foo Bar Buzz ....",
        "comments":
        [
            {
                "id": "comment3",
                "author_s": "Luke",
                "text_t": "I like this post!",
                "rated_i": 4
            }
        ]
    }
]'
```

5. Go to the collection's ***Query*** page and click ***Execute Query*** with default parameters to retrieve all documents from the index. The response from Solr looks as follows:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1724336874767"}},
  "response":{"numFound":5,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"comment1",
        "author_s":"John",
        "text_t":"Nice post!",
        "rated_i":5,
        "content_type":["comment"],
        "_version_":1808098244634345472},
      {
        "id":"comment2",
        "author_s":"Ann",
        "text_t":"Not interesting post...",
        "rated_i":3,
        "content_type":["comment"],
        "_version_":1808098244634345472},
      {
        "id":"post1",
        "name_s":"Sample Post for testing...",
        "text_t":"Lorem ipsum dolor sit amet, consectetur ....",
        "content_type":["post"],
        "_version_":1808098244634345472},
      {
        "id":"comment3",
        "author_s":"Luke",
        "text_t":"I like this post!",
        "rated_i":4,
        "content_type":["comment"],
        "_version_":1808098259769491456},
      {
        "id":"post2",
        "name_s":"Another demo post...",
        "text_t":"Foo Bar Buzz ....",
        "content_type":["post"],
        "_version_":1808098259769491456}]
  }}
```

Notice that the documents are returned as a flat structure, with children documents piled upon the parent ones — this is the way Solr stores nested objects internally.

Search nested data

By default, documents that match a search query do not include nested children in the response. For example, submit a search request using the `q=id:post1` query string. The request URL looks as follows:

```
http://ka-adh-1.ru-central1.internal:8983/solr/posts/select?indent=true&q.op=OR&q=id%3Apost1
```

The response contains a single document without descendants:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"id:post1",
      "indent":"true",
      "q.op":"OR",
      "_":"1724336874767"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"post1",
        "name_s":"Sample Post for testing...",
        "text_t":"Lorem ipsum dolor sit amet, consectetur ....",
        "content_type":["post"],
        "_version_":1808098244634345472}]
  }}
```

Child document transformer

To add a document's children to the response, use the child document transformer. The following request includes the `fl=*,[child]` parameter that tells Solr to return all the document fields and additionally all the document's children.

Sample request URL:

```
http://ka-adh-1.ru-central1.internal:8983/solr/posts/select?fl=*%2C%5Bchild%5D&indent=true&q.op=OR&q=id%3Apost1
```

The response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"id:post1",
      "indent":"true",
      "fl":"*,[child]",
      "q.op":"OR",
      "_":"1724336874767"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"post1",
        "name_s":"Sample Post for testing...",
        "text_t":"Lorem ipsum dolor sit amet, consectetur ....",
        "content_type":["post"],
        "_version_":1808098244634345472,
        "comments":[
          {
            "id":"comment1",
            "author_s":"John",
            "text_t":"Nice post!",
            "rated_i":5,
            "content_type":["comment"],
            "_version_":1808098244634345472},

          {
            "id":"comment2",
            "author_s":"Ann",
            "text_t":"Not interesting post...",
            "rated_i":3,
            "content_type":["comment"],
            "_version_":1808098244634345472}]}]
  }}
```

Block join query parsers

There are two block join query parsers that allow indexing and searching for relational content that has been indexed as nested documents. These parsers are:

- Block join children query parser. Used to get the children of an object matched by a query.
- Block join parent parser. Matches one or more child documents and returns their parent.

Example: block join children query parser

The following search query engages the block join children query parser:

```
q={!child of="content_type:post"}text_t:"Lorem"
```

Solr interprets this query string as follows:

1. Find all documents whose `text_t` field contains the `Lorem` string.
2. Get children of all the matched documents and add them to the response.

Notice the `of="content_type:post"` part which is a Block Mask. It tells Solr that parent objects are those objects whose `content_type` value is `post`.

The request URL for such a query looks as follows:

```
http://ka-adh-1.ru-central1.internal:8983/solr/posts/select?indent=true&q.
op=OR&q=q%3D%7B!child%20of%3D%22content_type%3Apost%22%7Dtext_t%3A%22Lorem%22
```

The sample response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"q={!child of=\"content_type:post\"}text_t:\"Lorem\"",
      "indent":"true",
      "q.op":"OR",
      "_":"1724274693355"}},
  "response":{"numFound":2,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"comment1",
        "author_s":"John",
        "text_t":"Nice post!",
        "rated_i":5,
        "content_type":["comment"],
        "_version_":1808018337579401216},
      {
        "id":"comment2",
        "author_s":"Ann",
        "text_t":"Not interesting post...",
        "rated_i":3,
        "content_type":["comment"],
        "_version_":1808018337579401216}]
  }}
```

Example: block join parent query parser

The following search query engages the block join parent query parser:

```
q={!parent which="content_type:post"}author_s:John
```

Solr interprets this query string as follows:

1. Find documents whose `author_s` field is `John`.
2. Get the ancestor of the matched document and add it to the response.

Notice the `which="content_type:post"` part, which is a Block Mask. It tells Solr that parent objects are those objects whose `content_type` value is `post`.

The request URL for such a query looks as follows:

```
http://ka-adh-1.ru-central1.internal:8983/solr/posts/select?indent=true&q.
op=OR&q=q%3D%7B!parent%20which%3D%22content_type%3Apost%22%7Dauthor_s%3AJohn
```

The sample response:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"q={!parent which=\"content_type:post\"}author_s:John",
      "indent":"true",
      "q.op":"OR",
      "_":"1724260126428"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"blog1",
        "name_s":"Sample Post for testing...",
        "text_t":"Lorem ipsum dolor sit amet, consectetur ....",
        "content_type":["post"],
        "_version_":1808018337579401216}]
  }}
```

Partial document updates in Solr

Konstantin Alpashkin

Contents

To Table of Contents

Overview
Atomic updates
  Update nested documents
In-place updates
Optimistic concurrency

Overview

Solr supports three major approaches to updating documents partially. This article describes each strategy with examples, namely:

- Atomic updates. Allow changing one or more document fields by reindexing the target document internally.
- In-place updates. A subset of atomic updates that allow updating numeric fields without reindexing the entire document.
- Optimistic concurrency. Allows conditional updates based on the document version.

In this article, the update examples are designed to modify the following sample document:

```
{
  "id": 1,
  "post_name": "Sample blog post...",
  "categories": ["leisure", "hobby"],
  "post_rank": 75.0,
  "post_date": "2024-01-02",
  "post_text": "Lorem ipsum dolor sit amet ...",
  "description": "A sample post for testing purposes"
}
```

The sample document is stored in a Solr collection named `test_collection`. For information on adding a document to the index, see Solr indexing overview.

Atomic updates

This type of updates allows changing individual document fields. This approach should be chosen when the speed of index modifications is critical to the client applications.

Solr provides several modifiers that can be used to update individual document fields. These modifiers are as follows:

- `set` — sets a new value or replaces the existing field's value. Removes the value if `null` or an empty list is specified. Accepts a single value or a list.
- `add` — adds the specified value(s) to a `multiValued` field. Accepts a single value or a list.
- `add-distinct` — adds the specified value(s) to a `multiValued` field only if the value is not already present in the index. Accepts a single value or a list.
- `remove` — removes all occurrences of the specified value from a `multiValued` field. Accepts a single value or a list.
- `removeregex` — removes all occurrences matching the specified regex from a `multiValued` field. Accepts a single value or a list.
- `inc` — increments a numeric field by a specific value. Accepts a single numeric value.

To perform an update, specify the corresponding modifier as a value of the field to be updated and submit the document to Solr. The following example updates several fields of the sample document:

```
{
  "id": 1,
  "post_name": {"set": "Updated post name ..."},
  "categories": {"add": ["science"]},
  "post_rank": {"inc": 5},
  "post_date": "2024-01-02",
  "post_text": "Lorem ipsum dolor sit amet ...",
  "description": {"removeregex": ".*testing.*"}
}
```

curl upload command

```
$ curl -X POST 'http://ka-adh-1.ru-central1.internal:8983/solr/test_collection/update?commit=true' -H 'Content-Type:
application/json' --data-binary '[{
  "id": 1,
  "post_name": {"set": "Updated post name ..."},
  "categories": {"add": ["science"]},
  "post_rank": {"inc": 5},
  "post_date": "2024-01-02",
  "post_text": "Lorem ipsum dolor sit amet ...",
  "description": {"removeregex": ".*testing.*"}
}]'
```

When Solr receives such a document, it recognizes the modifier expressions and updates only the corresponding document's fields. After the update, the document has the following view:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1726172218281"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"1",
        "post_name":["Updated post name ..."],
        "categories":["leisure",
          "hobby",
          "science"],
        "post_rank":[80.0],
        "post_date":["2024-01-02T00:00:00Z"],
        "post_text":["Lorem ipsum dolor sit amet ..."],
        "_version_":1810022770405277696}]
  }}
```

Update nested documents

Solr allows modifying, adding, and removing nested documents using atomic updates. The process is very similar to updating non-nested documents, the major distinctions that should be kept in mind are as follows:

- An update operation of a nested object must be routed to the appropriate Solr shard (the one that stores the parent document). Since Solr selects the target shard based on the document's ID, it is important to use the ID of the root document, not the ID of the child document being updated. To address this requirement, you can either specify a router explicitly (via `_route_` parameter) or you can use the compositeId router (used by default) to direct update operations to the appropriate shard. For more information on routing rules for updating child elements, see Solr documentation.
- When updating nested objects, it is mandatory to tell Solr which document is the parent for the object being updated. This can be done by specifying the `_root_` field in the update request.
- Solr attempts to apply the updates to the entire tree of nested documents instead of modifying individual objects, which may result in additional overhead.

Assume there is the following document with nested objects (comments) that needs to be updated:

```
{
    "id": "post2",
    "name_s": "Another demo post...",
    "text_t": "Foo Bar Buzz ....",
    "content_type": "post",
    "comments":
    [
        {
            "id": "post2!comment3",
            "author_s": "Luke",
            "text_t": "I like this post!",
            "rated_i": 4,
            "content_type": "comment"
        }
    ]
}
```

For example, to increment the numeric `rated_i` field using atomic updates, you have to submit the following document to Solr:

```
{
  "id": "post2!comment3", ①
  "_root_": "post2", ②
  "rated_i": { "inc": 1 }
}
```

① The ID of the nested document to update. Notice that the ID is composite (contains the `!` char). This allows the default compositeId router to route the update operation to the appropriate Solr shard.

② The `_root_` field is mandatory to allow Solr to identify the parent object whose child will be updated. If not specified, Solr rejects the update.

The result of the update:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1726172218281"}},
  "response":{"numFound":2,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"post2!comment3",
        "author_s":"Luke",
        "text_t":"I like this post!",
        "rated_i":5,
        "content_type":["comment"],
        "_version_":1810022942789074944},
      {
        "id":"post2",
        "name_s":"Another demo post...",
        "text_t":"Foo Bar Buzz ....",
        "content_type":["post"],
        "_version_":1810022942789074944}]
  }}
```

Example: add nested documents

Submitting the following document to Solr adds another nested object to the parent document (`"id": "post2"`):

```
{
  "id": "post2",
  "comments": { "add": { "id": "post2!comment4",
                         "author_s": "Rick Sanchez",
                         "text_t": "Another added comment here..",
                         "rated_i": 5,
                         "content_type": "comment"
                       } }
}
```

The result:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1726172218281"}},
  "response":{"numFound":3,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"post2!comment3",
        "author_s":"Luke",
        "text_t":"I like this post!",
        "rated_i":5,
        "content_type":["comment"],
        "_version_":1810023009958756352},
      {
        "id":"post2!comment4",
        "author_s":"Rick Sanchez",
        "text_t":"Another added comment here..",
        "rated_i":5,
        "content_type":["comment"],
        "_version_":1810023009958756352},
      {
        "id":"post2",
        "name_s":"Another demo post...",
        "text_t":"Foo Bar Buzz ....",
        "content_type":["post"],
        "_version_":1810023009958756352}]
  }}
```

Example: remove nested documents

Submitting the following document to Solr removes the nested document by ID:

```
{
    "id": "post2",
    "comments": {
        "remove": {
            "id": "post2!comment4"
        }
    }
}
```

The result:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1726172218281"}},
  "response":{"numFound":2,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"post2!comment3",
        "author_s":"Luke",
        "text_t":"I like this post!",
        "rated_i":5,
        "content_type":["comment"],
        "_version_":1810023111907606528},
      {
        "id":"post2",
        "name_s":"Another demo post...",
        "text_t":"Foo Bar Buzz ....",
        "content_type":["post"],
        "_version_":1810023111907606528}]
  }}
```

In-place updates

In-place updates are a subset of atomic updates. However, during a regular atomic update, the entire document is reindexed internally. With in-place updates, only the required fields are affected, and the rest of the document remains intact. Thus, the efficiency of in-place updates does not depend on the number of fields in the updated document. Apart from the internal differences regarding efficiency, there is no functional difference between atomic updates and in-place updates.

An update operation runs as an in-place update if the fields to be updated meet the following requirements:

- The fields are numeric and have the following properties: `indexed="false"`, `stored="false"`, `multiValued="false"`, and `docValues="true"`.
- The updated document has the `_version_` field with properties: `indexed="false"`, `stored="false"`, `multiValued="false"`, and `docValues="true"`.
- The CopyField destination of the updated field (if any) is numeric and has the following properties: `indexed="false"`, `stored="false"`, `multiValued="false"`, and `docValues="true"`.

For in-place updates, Solr supports the following modifiers:

- `set` — sets a new value or replaces the existing field's value.
- `inc` — increments a numeric field by a specific value.

To update numeric fields using the in-place approach, the corresponding fields must be defined in the schema. For example:

```
<field name="post_rank" type="float" indexed="false" stored="false" docValues="true"/>
```

To update the document from the example, submit the following document to Solr:

```
{
    "id": 1,
    "post_rank": {
        "set": 100
    }
}
```

In this case, the document will be modified using the in-place update strategy. The results of the update operation:

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1726172218281"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"1",
        "post_name":["Updated post name ..."],
        "categories":["leisure",
          "hobby",
          "science"],
        "post_rank":[100.0],
        "post_date":["2024-01-02T00:00:00Z"],
        "post_text":["Lorem ipsum dolor sit amet ..."],
        "_version_":1810023165979525120}]
  }}
```

Optimistic concurrency

Optimistic concurrency (a.k.a. optimistic locking) is a Solr feature that allows client applications to ensure that the same document is not modified by multiple clients in parallel. This feature makes use of the `_version_` field and expects this field to be present in every document in the Solr index. By default, Solr schema includes the `_version_` field, and the field is automatically added to each new document. During an update, Solr compares the `_version_` value from the index with the value provided in the update request, thus ensuring that the document was not modified by other clients.

To update a document using the optimistic concurrency approach, the client has to pass a correct `_version_` value along with the document. `_version_` is an ordinary Solr field that can be queried just like any other field, for example:

```
$ curl -X GET 'http://ka-adh-1.ru-central1.internal:8983/solr/test_collection/query?
q=*:*&fl=id,post_name,_version_&omitHeader=true' -H 'Content-Type: application/json'
```

The response:

```
{
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "id":"1",
        "post_name":["Sample blog post..."],
        "_version_":1809999244663193600}]
  }}
```

You can pass the document's version to Solr in two ways:

- Using the `_version_=<version>` query parameter in the URL. For example:

```
$ curl -X POST 'http://ka-adh-1.ru-central1.internal:8983/solr/test_collection/update?_version_=123' -H 'Content-Type:
application/json' --data-binary '[{"id": 1, "updated_field": "updated value"}]'
```

- By embedding the `_version_` value in the update document. For example:

```
{
  "id": 1,
  "_version_": 123,
  "post_rank": {"set": 80}
}
```

This approach is handy when documents are sent in a batch and different `_version_` values need to be specified for each document.

If Solr receives an invalid version, it rejects the update, returns the 409 HTTP response and the following error:

```
"error":{
    "metadata":[
      "error-class","org.apache.solr.common.SolrException",
      "root-error-class","org.apache.solr.common.SolrException"],
    "msg":"version conflict for 1 expected=12345 actual=1809999244663193600",
    "code":409}
```

If a valid document version is provided, the update succeeds and Solr generates a new version for the updated document. The following request successfully updates the test document by passing a valid version number as a query parameter.

```
$ curl -X POST 'http://ka-adh-1.ru-central1.internal:8983/solr/test_collection/update?
_version_=1810031847346995208versions=true&omitHeader=true' -H 'Content-Type: application/json' --data-binary
'[{"id":1,"post_rank": {"set": 80}}]'
```

The successful update response:

```
{
  "adds":[
    "1",1810019385852559360]
}
```

TIP

The `versions=true` query parameter forces Solr to include the document version in each response. This may be useful to avoid redundant GET requests to learn a document's version.

Reindex data in Solr

Konstantin Alpashkin

Contents

To Table of Contents

- Cases when reindexing is required
  - Schema updates
  - Data updates
  - Corrupted index
  - Performance improvements
- Reindexing strategies
  - Delete all documents
  - Index to another collection
- REINDEXCOLLECTION command
  - Command parameters
  - Command usage example

In Solr, reindexing is the process of deleting all the content from a Solr index followed by re-ingesting the data from a data source (database, file, or other data storage) to build a fresh index. It is important that all the documents and Lucene segments have to be dropped in order to avoid collisions during the re-creation of an index.

This article lists common situations when reindexing is required and describes popular reindexing strategies. The major challenges while performing reindexing are possible downtime and performance impact. Reindexing large datasets is a time-consuming and resource-intensive process that may lead to a temporary collection downtime or inconsistent results returned by search queries. Techniques like aliasing described in the article can help mitigate this.

Cases when reindexing is required

Below are some common cases that require a full reindexing of a collection. For other cases, using partial updates may suffice.

Schema updates

> The main rule is as follows: whenever you change a part of Solr settings that affects the way the data is written into an index, full reindexing is required. If you modify only the query-time logic (for example, updating analyzer's query behavior), reindexing is not required, although still highly recommended.

Changing the schema of a Solr collection (for example, modifying field types, adding new fields, or changing the way how fields are analyzed) almost always requires full reindexing.

This requirement derives from the design of the underlying Lucene search engine. Lucene segments that actually store raw data are immutable, meaning that once a segment is written to disk, it cannot be changed. The segments can be added, merged, deleted, but not altered. Lucene segments are loosely coupled with the Solr schema since the schema is a Solr-only concept. The schema file only tells Solr how the data should be placed into a Lucene index and how it should be queried later. After the data is written into a Lucene index, Solr's schema has no control over the stored data structures.

Data updates

If the source data has been significantly updated or changed (for example, due to bulk updates or corrections), it might be more efficient to reindex the entire dataset rather than applying updates incrementally.

Corrupted index

If a Solr index becomes corrupted or some documents get indexed incorrectly, reindexing can restore the index to a consistent state.

Performance improvements

Changes to indexing settings (such as tokenization, filters, or analyzers) or optimizations may require reindexing to take effect. The rule is the same: if you update the settings related to writing data into the index, reindexing is necessary; updating the query-time logic does not necessarily require reindexing.

Reindexing strategies

There are two major reindexing strategies described below. These strategies help to ensure that the Lucene index is completely empty before re-ingesting data into it.

Delete all documents

The first approach assumes manually deleting all documents from the target Solr index. For example, you can delete all documents from a collection by submitting the following request:

```
$ curl http://ka-adh-1.ru-central1.internal:8983/solr/test_collection/update?commit=true -H "Content-Type: application/json" --
data-binary '{"delete": { "query": "*:*" }}'
```

Where:

- `ka-adh-1.ru-central1.internal` — the name of the host with running Solr server;
- `test_collection` — the name of the collection to purge.

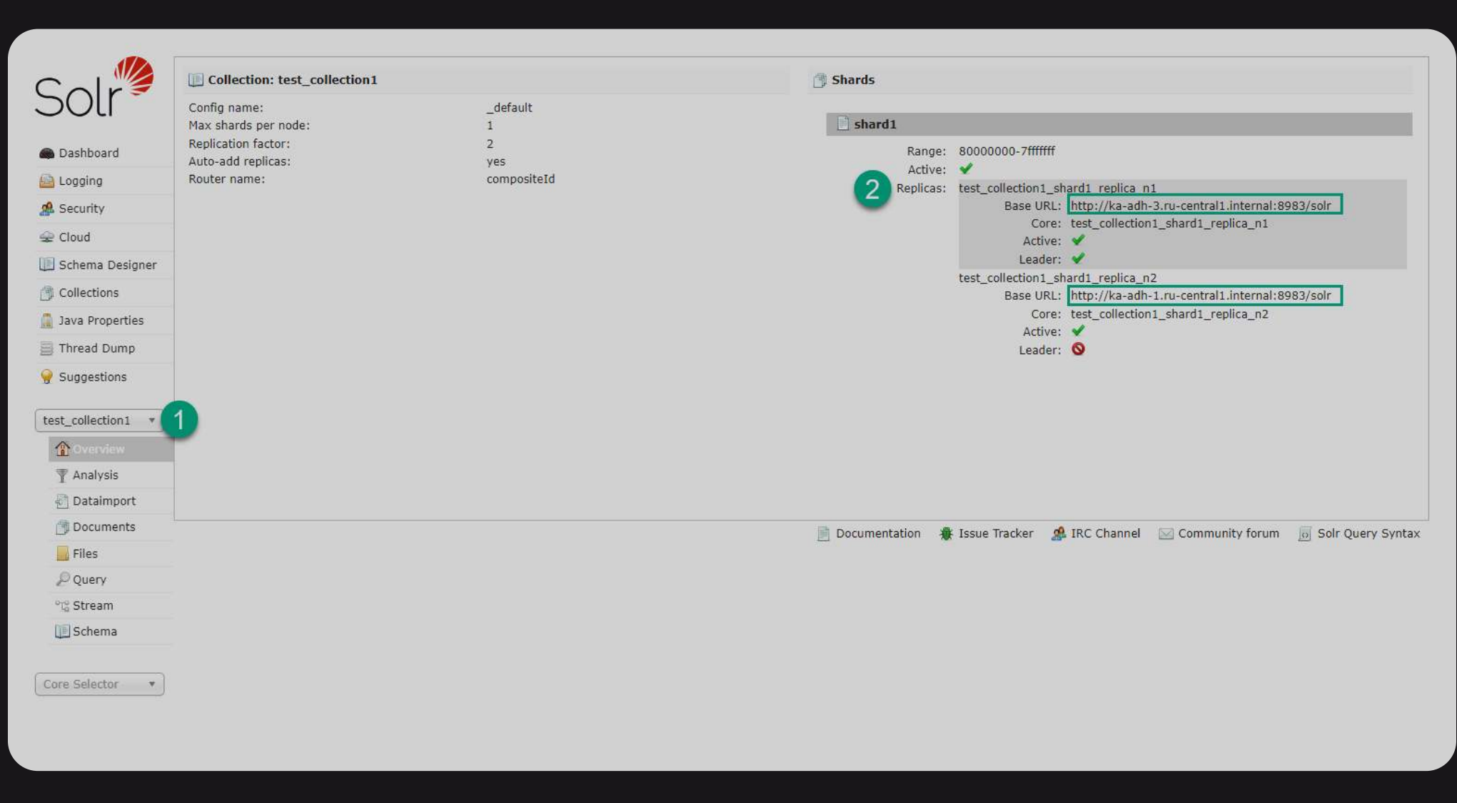
Sample response:

```
{
  "responseHeader":{
    "rf":1,
    "status":0,
    "QTime":40}}
```

TIP

After the delete operation, perform another search query to verify that all the documents have been deleted from the collection, as that ensures the Lucene index segments have been deleted as well.

To verify that there are no Lucene segments left in your index, check the collection's *{SOLR_HOME}/<collection_name>_*/data/index/* directory and confirm it has no segment files. It is necessary to verify that the index data has been removed in every shard and every replica of your Solr cluster. To locate Lucene shards and replicas that belong to a given collection, use the Solr UI page as shown in the image.



Collection: test_collection1
Config name: _default
Max shards per node: 1
Replication factor: 2
Auto-add replicas: yes
Router name: compositeId

Shards
shard1
Range: 80000000-7fffffff
Replicas: test_collection1_shard1_replica_n1
Base URL: http://ka-adh-3.ru-central1.internal:8983/solr
Core: test_collection1_shard1_replica_n1
test_collection1_shard1_replica_n2
Base URL: http://ka-adh-1.ru-central1.internal:8983/solr
Core: test_collection1_shard1_replica_n2

Solr shards location

Then, connect to each host that stores shards and replicas and run the command:

```
$ ls -l /srv/solr/server/test_collection_*/data/index
```

Where:

- `/srv/solr/server/` is the default location where Solr stores index data (`SOLR_HOME`);
- `test_collection` is the collection name.

The following output indicates that the collection data has been deleted and the index is empty:

```
[admin@ka-adh-3 ~]$ ls -l /srv/solr/server/test_collection_*/data/index
total 4
-rw-r--r--. 1 solr solr 135 Oct 11 12:37 segments_7
-rw-r--r--. 1 solr solr   0 Oct 11 09:00 write.lock
```

The *segments_N* and *write.lock* files are auxiliary and should not be deleted during the reindexing. If this is the state of your *data/index/* directory, your collection is ready for re-ingesting the data.

IMPORTANT

Do not delete files manually in the *data/index/* directory as this will likely lead to an index corruption.

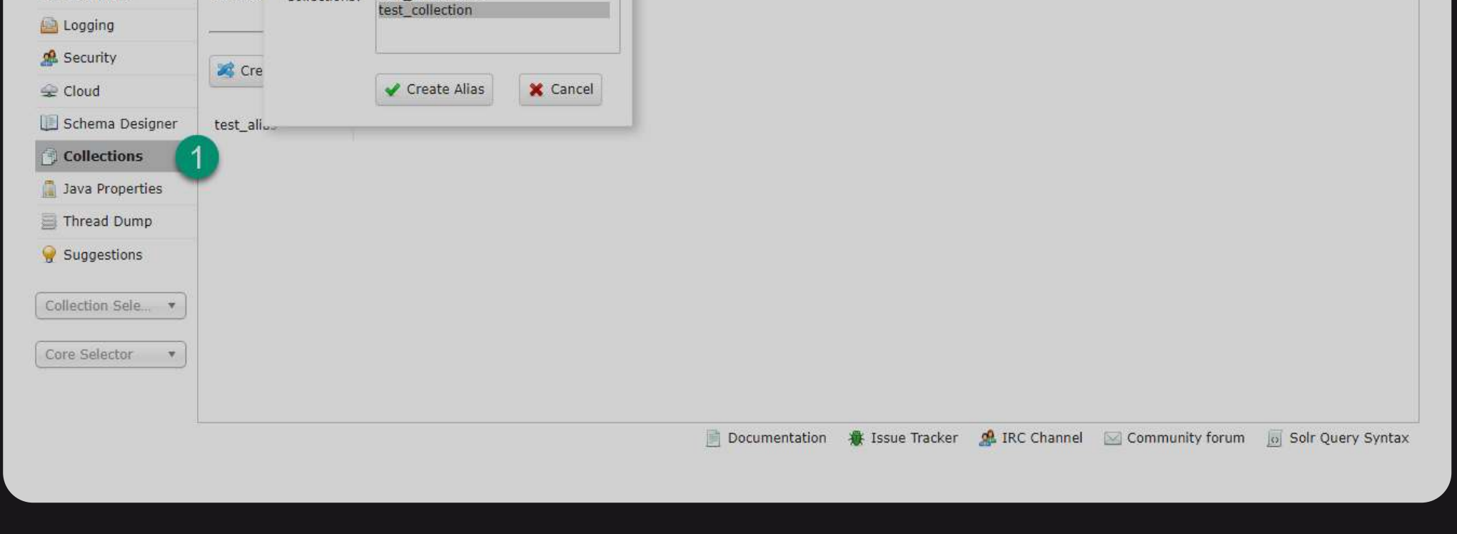
The major disadvantage of this approach is possible downtime or inconsistent search results during the re-ingesting of the fresh data. The next described approach can mitigate this.

Index to another collection

Another approach to reindexing involves using the collection alias feature that allows routing client applications to a different collection through an alias without downtime.

By following this approach, the simplified reindexing flow comprises the following steps:

1. Create a new collection. The old collection that requires reindexing continues its operation.
2. Index fresh data into the new collection.
3. After the indexing is complete, create an alias that routes your client application requests to the new collection. You can create a collection alias using Solr web UI as shown in the image below.



Alias Name: test_alias
Collections: test_collection1, test_collection

Create alias

4. Delete the old collection as no longer needed.

REINDEXCOLLECTION command

Solr provides the REINDEXCOLLECTION command that allows reindexing a collection using existing data from the source collection.

NOTE

`REINDEXCOLLECTION` is potentially a lossy operation since it preserves only the data with stored=true fields. It is recommended not to delete the source collection immediately after reindexing, but keep it as a backup in case some data is lost.

To run the command, submit a GET request to the */admin/collections?action=REINDEXCOLLECTION&name=<collection_name>* URL. When reindexing is started using this command, the source collection transitions to the read-only mode to ensure that all source documents are properly processed. Only one reindexing operation may run for a given source collection. Long-running, failed, or hanging reindexing operations can be terminated by using the abort option, which also removes partial results. If an error occurs while transferring documents to a new collection, Solr deletes temporary/target collections and resets the read-only state of the source collection.

Command parameters

The `REINDEXCOLLECTION` command accepts the following parameters as URL query strings.

Parameter	Description
name	Mandatory. The source collection name or alias
cmd	Specifies the operation to perform. The supported operations are: - `start` — starts the reindexing process (default). - `abort` — aborts a running or crashed reindexing session. Deletes partial results, if any. - `status` — returns status information on the running reindexing session.
target	The name of the target collection. If not specified, Solr creates a collection with a unique name and after reindexing is done, creates an alias that points to the newly created collection, thus "isolating" the original source collection from read and update operations
q	An optional query to select documents for reindexing. The default value is `*:*`
fl	An optional list of fields to reindex. The default value is `*`
rows	Specifies the batch size to transfer documents. Depending on the average size of the document, large batch sizes may cause memory issues. Defaults to 100
configName	An optional name of the configset for the target collection. By default, Solr uses the same configset as the source collection
removeSource	If set to `true`, removes the source collection after reindexing

Command usage example

To run the examples below, create a sample collection and index a dummy document as described in the Solr indexing overview article.

Example: running REINDEXCOLLECTION with default parameters

The following request starts reindexing of the source collection named `test_collection`.

```
$ curl 'http://localhost:8983/solr/admin/collections?action=REINDEXCOLLECTION&name=test_collection'
```

Sample response:

```
{
  "responseHeader":{
    "status":0,
    "QTime":6453},
  "reindexStatus":{
    "actualSourceCollection":"test_collection",
    "actualTargetCollection":".rx_test_collection_3",
    "alias":"test_collection -> .rx_test_collection_3",
    "checkpointCollection":".rx_ck_test_collection",
    "inputDocs":1,
    "phase":"done",
    "processedDocs":1,
    "state":"finished"}}
```

As the response suggests, Solr has created a new collection `.rx_test_collection_3` and the `test_collection` alias that points to the new collection. Check the contents of the new collection by explicitly specifying the collection name and bypassing the alias:

```
$ curl 'http://ka-adh-1.ru-central1.internal:8983/solr/.rx_test_collection_3/select?indent=true&q.op=OR&q=*%3A*'
```

The response contains the dummy document copied from the original source collection.

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "txn_id":[1],
        "txn_date":["2024-01-02T00:00:00Z"],
        "txn_value":[75.0],
        "comment":["The first transaction."],
        "id":"1",
        "acc_id":[1001],
        "_version_":1812905933055983616}]
  }}
```

The source collection (`test_collection`) is not removed, however now Solr routes all read/update requests to the new `.rx_test_collection_3` collection through the alias.

Example: reindexing specific data

The following example starts the reindexing process and specifies parameters to select only specific data to be reindexed to the target collection.

```
$ curl 'http://localhost:8983/solr/admin/collections?
action=REINDEXCOLLECTION&name=test_collection&q=txn_id:1&fl=id,txn_id,txn_value,acc_id&target=test_collection2'
```

Sample response:

```
{
  "responseHeader":{
    "status":0,
    "QTime":6215},
  "reindexStatus":{
    "actualSourceCollection":"test_collection",
    "actualTargetCollection":"test_collection2",
    "checkpointCollection":".rx_ck_test_collection",
    "inputDocs":1,
    "phase":"done",
    "processedDocs":1,
    "state":"finished"}}
```

Check the contents of the target collection:

```
$ curl 'http://ka-adh-1.ru-central1.internal:8983/solr/test_collection2/select?indent=true&q.op=OR&q=*%3A*'
```

Solr returns the test document that was reindexed with a reduced set of fields as defined by command parameters.

```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":0,
    "params":{
      "q":"*:*",
      "indent":"true",
      "q.op":"OR",
      "_":"1728921728844"}},
  "response":{"numFound":1,"start":0,"numFoundExact":true,"docs":[
      {
        "txn_id":[1],
        "txn_value":[75.0],
        "id":"1",
        "acc_id":[1001],
        "_version_":1812908971102568448}]
  }}
```

Deduplication in Solr

Konstantin Alpashkin

Contents

To Table of Contents

Overview

Unique key deduplication

- Auto-generated IDs

Signature-based deduplication

Deduplication example

- Step 1. Create a test collection
- Step 2. Modify a Solr schema
- Step 3. Modify solrconfig.xml
- Step 4. Submit data for indexing

Deduplication restrictions in SolrCloud

Overview

Deduplication in Solr refers to the process of identifying and deleting duplicate documents from the index to ensure that only unique entities are returned in search results. Duplicate or near-duplicate documents that appear in a Solr index may result in multiple documents representing the same business entity, leading to cluttered search results and inefficiencies.

This article describes two major techniques for detecting duplicates in a Solr index:

1. Unique key deduplication. This approach relies on preventing duplicate injections based on a document field chosen to be unique among all the documents in a collection. This is very similar to primary keys in traditional databases.
2. Signature-based deduplication. This method assumes calculating a signature (hash) for each document during the indexing phase and comparing the hashes before adding new documents to the collection.

More details about each technique are described below.

Unique key deduplication

In Solr, the `uniqueKey` configuration property defines a field to uniquely identify each document in a collection. It ensures that every document has a distinct identifier, preventing duplicate entries from getting into the index. When a new document is indexed with a `uniqueKey` value that is already present in the index, Solr replaces the old document with the new one rather than indexing it as a separate record.

The `uniqueKey` property is set in the Solr schema and by default uses the `id` field as a unique key. The default unique key definition in the Solr schema is as follows:

```
<uniqueKey>id</uniqueKey>
```

Auto-generated IDs

Ingesting documents to Solr without providing the `id` field (or any other field used as `uniqueKey`) forces Solr to auto-generate a UUID (universally unique identifier) for that field to identify the document internally. An auto-generated identifier is a random UUID that guarantees collection-wide uniqueness; however, it is agnostic of the data carried by other document fields. When ingesting `uniqueKey`-less documents to Solr, you might want to track the auto-generated UUIDs returned by the Solr server. However, in most cases, the responsibility to generate a unique key lies with the client or the data source feeding the documents to Solr.

Signature-based deduplication

Another approach to preventing duplicates assumes calculating a signature (hash) for each document during the indexing phase and checking whether the index already contains records with the matching hash. If the calculated signature is already present in the index, Solr can either overwrite or skip such a document. You can configure Solr to calculate the signature based on one or more fields, and specify a custom document field to carry the hash value.

Solr natively supports this approach using update request processors and a low collision or fuzzy hash algorithms. Out-of-the-box, Solr provides the following implementations of the `Signature` class used to generate the signature:

- MD5Signature — generates a 128-bit MD5 hash.
- Lookup3Signature — generates a 64-bit hash, which is much faster than MD5 and produces a smaller output to index.
- TextProfileSignature — a fuzzy hash implementation for near-duplicate detection that works best on longer texts.

Deduplication example

The following steps walk you through a basic signature-based deduplication scenario. To configure the deduplication processes using this approach, you have to modify the *solrconfig.xml* and *schema.xml* files as described further.

Step 1. Create a test collection

First, create a test collection by running the following command on a host with the Solr Server component:

```
$ /usr/lib/solr/bin/solr create -c test_collection -s 1 -rf 1
```

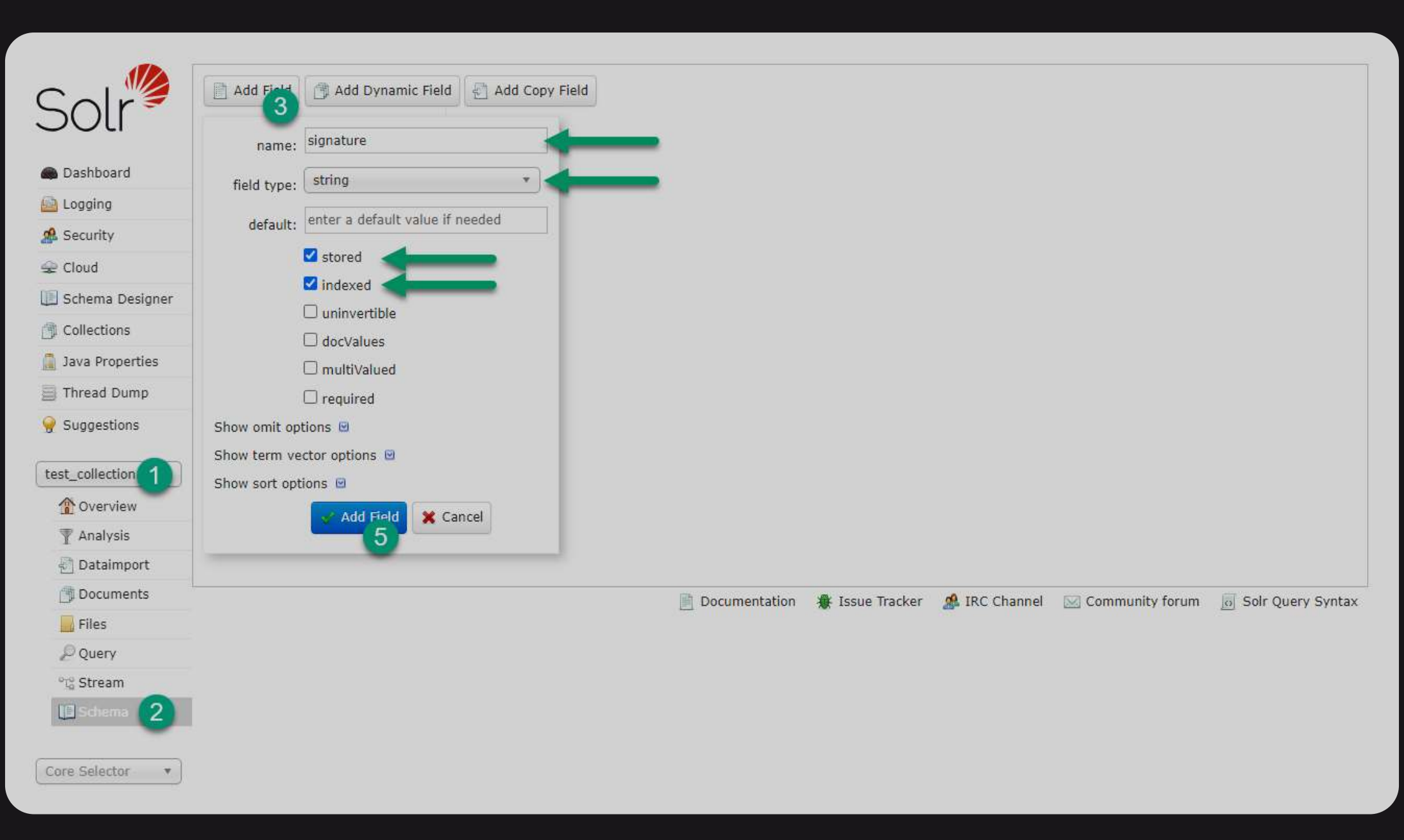
This creates a new collection (`test_collection`) with one shard and the replication factor set to `1`. The collection is created using the default `_default` configset.

The output:

```
WARNING: Using _default configset with data driven schema functionality. NOT RECOMMENDED for production use.
         To turn off: bin/solr config -c demo_collection_dedup -p 8983 -action set-user-property -property update.
autoCreateFields -value false
Created collection 'test_collection' with 1 shard(s), 1 replica(s) with config-set 'test_collection'
```

Step 2. Modify a Solr schema

In the Solr schema, define a new string field (for example, `signature`) where Solr will store hash values calculated for each document. The field must have the `stored=true` and `indexed=true` properties. To create the field, use Solr Admin UI as shown in the image below.



name: signature

field type: string

default: enter a default value if needed

☑ stored

☑ indexed

☐ uninvertible

☐ docValues

☐ multiValued

☐ required

Create a field in Solr schema

Step 3. Modify solrconfig.xml

Since in ADH Solr always runs in the SolrCloud mode, the configuration files like *solrconfig.xml* are managed by ZooKeeper to sync configs among several cluster nodes. Follow the steps below to update *solrconfig.xml* via ZooKeeper:

1. On the host with the Solr server and ZooKeeper server components, connect to ZooKeeper via CLI:

   ```
   $ sudo /usr/lib/zookeeper/bin/zkCli.sh
   ```

2. In the opened ZooKeeper shell, list all the znodes:

   ```
   $ ls /
   ```

   The output:

   ```
   [zk: localhost:2181(CONNECTED) 4] ls /
   [Arenadata.Hadoop-11.solr.server, arenadata, hadoop-ha, zookeeper]
   ```

   In this example, *Arenadata.Hadoop-11.solr.server* is the znode under which all Solr configuration files are stored. We will need it on further steps.

3. Download the Solr configset from ZooKeeper to the local file system using the zk downconfig command:

   ```
   $ /usr/lib/solr/bin/solr zk downconfig -n test_collection -d ~/test_collection
   ```

   Where:

   - `-n test_collection` is the name of the Solr configuration for the test collection in ZooKeeper. The full ZooKeeper path to the configuration is */Arenadata.Hadoop-11.solr.server/configs/test_collection*.
   - `-d ~/test_collection` is the path to a local directory to save the configuration.

   The command output is as follows:

   ```
   Downloading configset test_collection from ZooKeeper at ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.
   internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-11.solr.server to directory /home/admin/test_collection/conf
   ```

4. In the downloaded configuration directory, edit the *solrconfig.xml* file, for example, using `vi`:

   ```
   $ vi test_collection/conf/solrconfig.xml
   ```

5. Add the following update request processor chain definition to *solrconfig.xml*:

   ```
   <updateRequestProcessorChain name="my-dedup"> ①
      <processor class="solr.processor.SignatureUpdateProcessorFactory">
          <bool name="enabled">true</bool>
          <bool name="overwriteDupes">false</bool> ②
          <str name="signatureField">signature</str> ③
          <str name="fields">txn_id,acc_id,txn_value</str> ④
          <str name="signatureClass">solr.processor.Lookup3Signature</str> ⑤
     </processor>
     <processor class="solr.LogUpdateProcessorFactory" />
     <processor class="solr.RunUpdateProcessorFactory" />
   </updateRequestProcessorChain>
   ```

   ① Specifies a name for the processor chain. Later, when submitting an indexing request, this processor chain can be explicitly called by name.

   ② Tells Solr whether to overwrite documents with matching signatures. Overwriting works only if `uniqueKey` is used as `signatureField`. For more information, see Deduplication restrictions in SolrCloud.

   ③ Specifies the field name to store calculated signatures. In this scenario, the corresponding field has already been defined in the schema.

   ④ Specifies one or more document fields to calculate the signature.

   ⑤ The `Signature` class implementation to use.

6. Save the file and upload the modified configuration to ZooKeeper using the zk upconfig command:

   ```
   $ /usr/lib/solr/bin/solr zk upconfig -n test_collection -d ~/test_collection
   ```

   The output:

   ```
   Uploading /home/admin/test_collection/conf for config test_collection to ZooKeeper at ka-adh-1.ru-central1.internal:2181,ka-
   adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-11.solr.server
   ```

7. Restart the Solr service.
8. Verify that Solr has picked up the updated configuration. For this, in Solr Admin UI, select the test collection and go to ***Files***. The *solrconfig.xml* and *managed-schema* files must contain the updated configuration.

Step 4. Submit data for indexing

On a host with the Solr Server component, create a file with test documents as shown below.

```
[
  {
    "id": 1,
    "txn_id": "1",
    "acc_id": 1001,
    "txn_value": 75.0,
    "comment": "The first transaction."
  },
  {
    "id": 2,
    "txn_id": "1",
    "acc_id": 1001,
    "txn_value": 75.0,
    "comment": "Intentionally duplicate transaction record."
  }
]
```

The two documents refer to the same business entity — a transaction with identical `txn_id`, `acc_id`, `txn_value` fields. However, the documents have different `id` values, which makes Solr treat them as non-duplicates. To engage the deduplication mechanism, submit the test documents to Solr using the following request:

```
$ curl -X POST 'http://ka-adh-3.ru-central1.internal:8983/solr/demo_collection_dedup/update?commit=true&update.chain=mydedupechain'
-H 'Content-Type: application/json' --data-binary @transactions.json
```

Where:

- `ka-adh-3.ru-central1.internal` — the name of the host with a running Solr server component;
- `update.chain=mydedupechain` — the name of the update request processor chain to route the indexing request through.

The sample response:

```
{
  "responseHeader":{
    "rf":1,
    "status":0,
    "QTime":168}}
```

Now query Solr to view the indexed documents:

```
http://ka-adh-3.ru-central1.internal:8983/solr/test_collection/select?indent=true&q.op=OR&q=*%3A*&omitHeader=true
```

The response:

```
{
  "response": {
    "numFound": 2,
    "start": 0,
    "numFoundExact": true,
    "docs": [
      {
        "id": "1",
        "txn_id": [1],
        "acc_id": [1001],
        "txn_value": [75],
        "comment": [
          "The first transaction."
        ],
        "signature": "3db4527ba951dcd3",
        "_version_": 1.8136481935224996e+18
      },
      {
        "id": "2",
        "txn_id": [1],
        "acc_id": [1001],
        "txn_value": [75],
        "comment": [
          "Intentionally duplicate transaction record."
        ],
        "signature": "3db4527ba951dcd3",
        "_version_": 1.8136481935256453e+18
      }
    ]
  }
}
```

Notice the `signature` field, which is identical for both documents as they carry almost identical data. Having your documents tagged with signatures, you can easily detect duplicates in your collection and delete them if needed.

Deduplication restrictions in SolrCloud

Although Solr provides an option to overwrite documents with matching signatures (`<bool name="overwriteDupes">true</bool>`), this feature works only if the `uniqueKey` field is used as `signatureField`. This restriction applies to the SolrCloud which is how Solr works in ADH. Thus, attempting deduplication with `signatureField` other than `uniqueKey` will not work correctly due to the collisions when routing updates to replicas on other cluster nodes.

Solr UI overview

Konstantin Alpashkin

Contents

To Table of Contents

- Dashboard
- Logging
- Security
- Cloud
  - Nodes
  - Tree
  - ZK Status
  - Graph
- Schema Designer
- Collections
- Java Properties
- Thread Dump

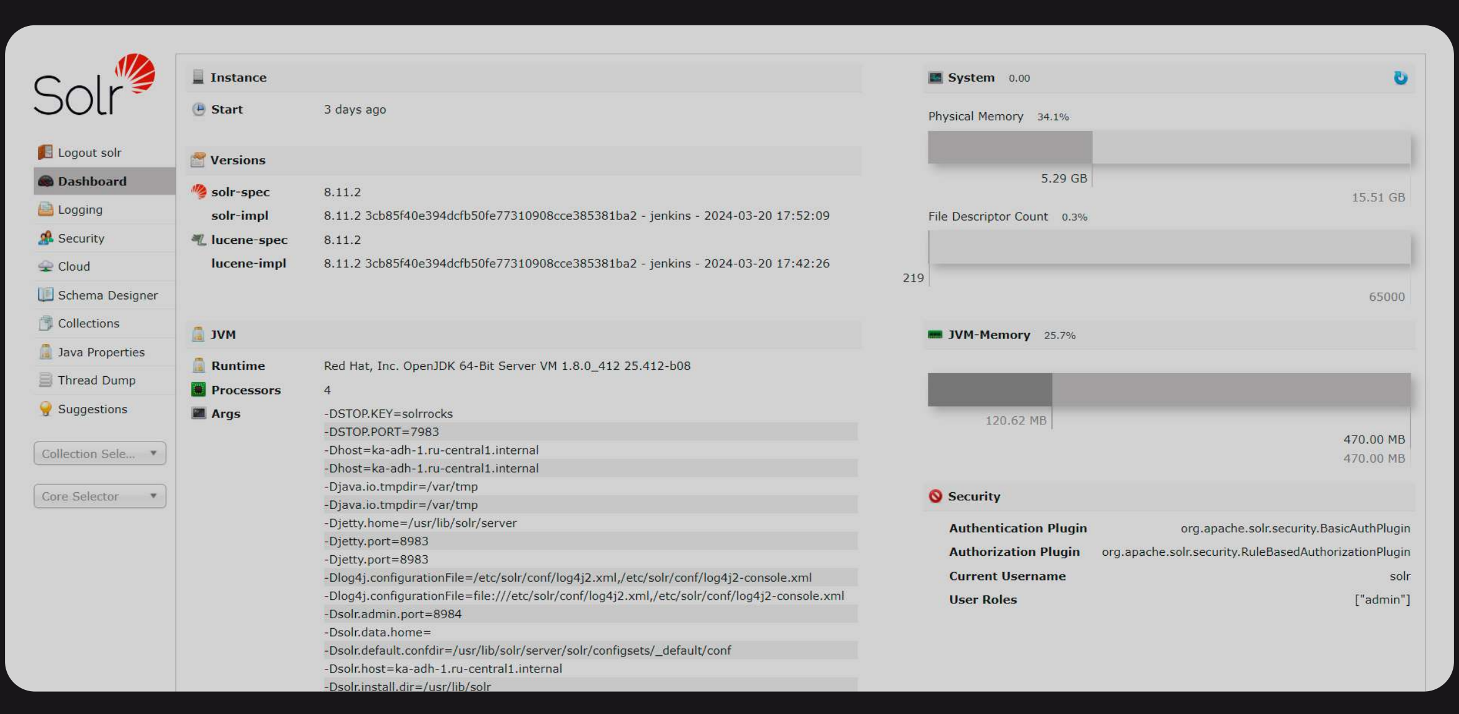
The Solr service provides a web interface where you can view diagnostic and status information, manage collections, configure security, and so on. By default, the web UI is available at *http://<host>:8983/solr/#/*, where `<host>` is the ADH host with the Solr Server component installed. You can change the default port number (`8983`) in the ***solr.xml*** Solr configuration section in ADCM. The up-to-date URL(s) of the Solr web UI can be found in ADCM (***Clusters → <YOUR_CLUSTER> → Services → Solr → Info***).

The Solr web UI includes the following sections described below:

- Dashboard
- Logging
- Security
- Cloud
- Schema Designer
- Collections
- Java Properties
- Thread Dump

Dashboard

This page displays information about the Solr server instance that runs on the given host, including system resource consumption, JVM parameters, and security details.



Solr
Logout solr
Dashboard
Logging
Security
Cloud
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
Instance
Versions
JVM
System 0.00
Physical Memory 34.1%
5.29 GB
15.51 GB
File Descriptor Count 0.3%
219
65000
JVM-Memory 25.7%
120.62 MB
470.00 MB
470.00 MB
Security

Start: 3 days ago

solr-spec: 8.11.2
solr-impl: 8.11.2 3cb85f40e394dcfb50fe77310908cce385381ba2 - jenkins - 2024-03-20 17:52:09
lucene-spec: 8.11.2
lucene-impl: 8.11.2 3cb85f40e394dcfb50fe77310908cce385381ba2 - jenkins - 2024-03-20 17:42:26

Runtime: Red Hat, Inc. OpenJDK 64-Bit Server VM 1.8.0_412 25.412-b08
Processors: 4
Args:

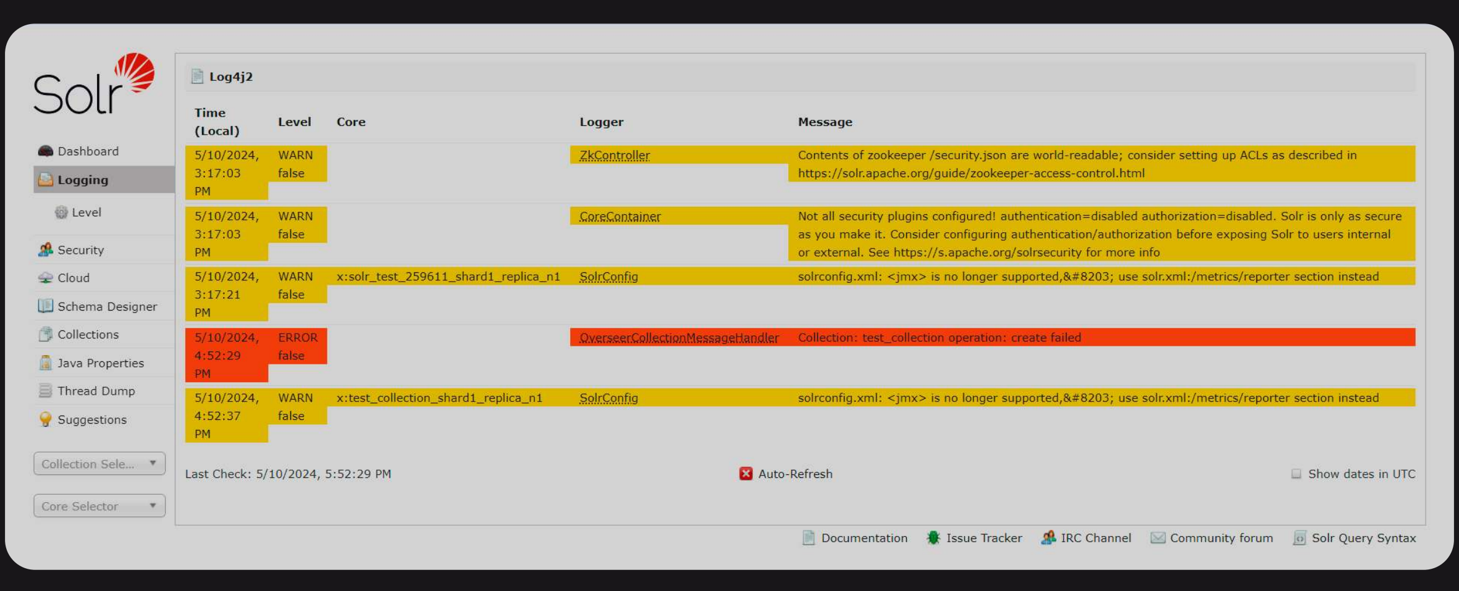
```
-DSTOP.KEY=solrrocks
-DSTOP.PORT=7983
-Dhost=ka-adh-1.ru-central1.internal
-Dhost=ka-adh-1.ru-central1.internal
-Djava.io.tmpdir=/var/tmp
-Djava.io.tmpdir=/var/tmp
-Djetty.home=/usr/lib/solr/server
-Djetty.port=8983
-Djetty.port=8983
-Dlog4j.configurationFile=/etc/solr/conf/log4j2.xml,/etc/solr/conf/log4j2-console.xml
-Dlog4j.configurationFile=file:///etc/solr/conf/log4j2.xml,/etc/solr/conf/log4j2-console.xml
-Dsolr.admin.port=8984
-Dsolr.data.home=
-Dsolr.default.confdir=/usr/lib/solr/server/solr/configsets/_default/conf
-Dsolr.host=ka-adh-1.ru-central1.internal
-Dsolr.install.dir=/usr/lib/solr
```

Authentication Plugin: org.apache.solr.security.BasicAuthPlugin
Authorization Plugin: org.apache.solr.security.RuleBasedAuthorizationPlugin
Current Username: solr
User Roles: ["admin"]

The Dashboard page

Logging

This page displays Log4j2 logs generated by the Solr server.



Solr
Dashboard
Logging
Level
Security
Cloud
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
Log4j2
Auto-Refresh
Show dates in UTC
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

Time (Local)	Level	Core	Logger	Message
5/10/2024, 3:17:03 PM	WARN false		ZkController	Contents of zookeeper /security.json are world-readable; consider setting up ACLs as described in https://solr.apache.org/guide/zookeeper-access-control.html
5/10/2024, 3:17:03 PM	WARN false		CoreContainer	Not all security plugins configured! authentication=disabled authorization=disabled. Solr is only as secure as you make it. Consider configuring authentication/authorization before exposing Solr to users internal or external. See https://s.apache.org/solrsecurity for more info
5/10/2024, 3:17:21 PM	WARN false	x:solr_test_259611_shard1_replica_n1	SolrConfig	solrconfig.xml: <jmx> is no longer supported,&#8203; use solr.xml:/metrics/reporter section instead
5/10/2024, 4:52:29 PM	ERROR false		OverseerCollectionMessageHandler	Collection: test_collection operation: create failed
5/10/2024, 4:52:37 PM	WARN false	x:test_collection_shard1_replica_n1	SolrConfig	solrconfig.xml: <jmx> is no longer supported,&#8203; use solr.xml:/metrics/reporter section instead

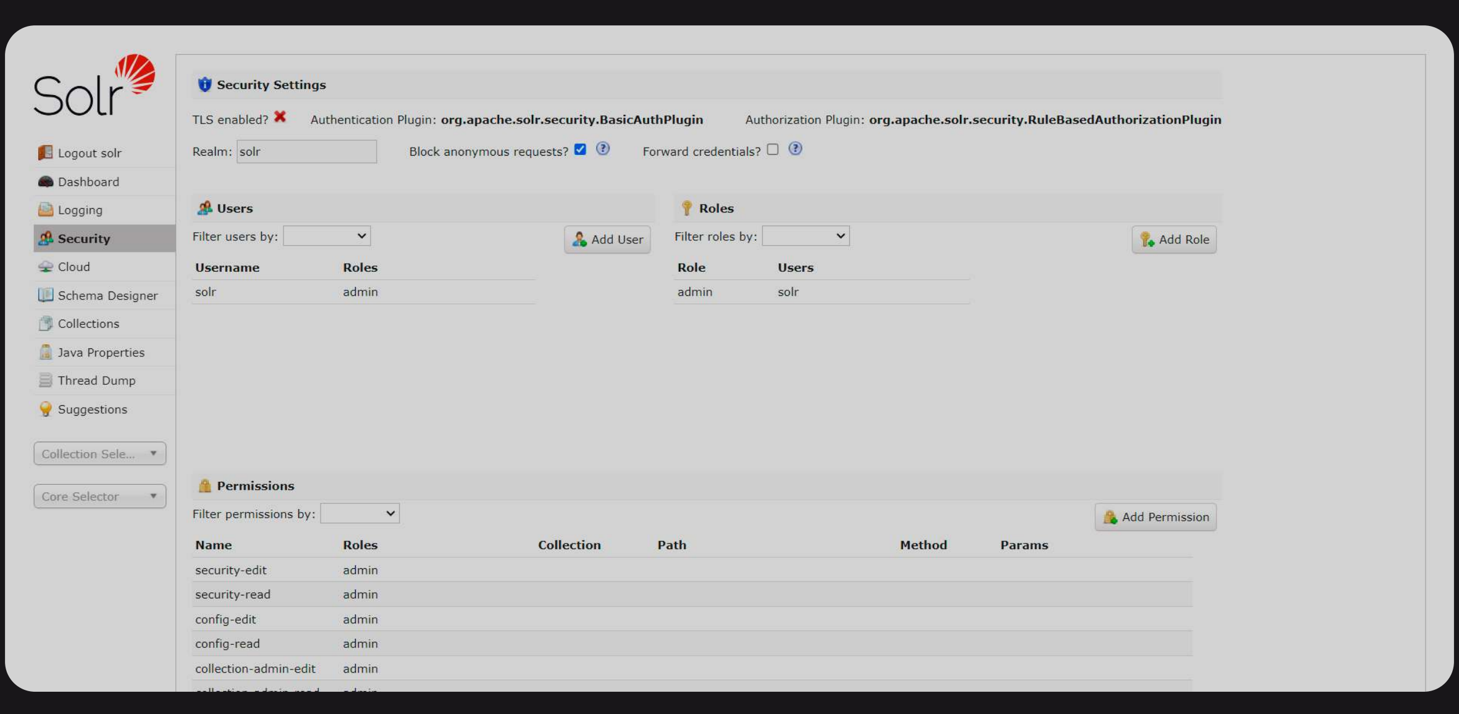
Last Check: 5/10/2024, 5:52:29 PM

The Logging page

On the ***Level*** tab, you can set logging levels for individual Solr components and classes.

Security

This page contains Solr security settings such as users, roles, permissions, and authentication/authorization parameters. These settings are cluster-wide, meaning that they affect security configuration for all Solr servers in the ADH cluster.



Solr
Logout solr
Dashboard
Logging
Security
Cloud
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
Security Settings
Users
Add User
Roles
Add Role
Permissions
Add Permission

TLS enabled? ✗ Authentication Plugin: org.apache.solr.security.BasicAuthPlugin Authorization Plugin: org.apache.solr.security.RuleBasedAuthorizationPlugin

Realm: solr Block anonymous requests? ☑ Forward credentials? ☐

Filter users by:

Username	Roles
solr	admin

Filter roles by:

Role	Users
admin	solr

Filter permissions by:

Name	Roles	Collection	Path	Method	Params
security-edit	admin				
security-read	admin				
config-edit	admin				
config-read	admin				
collection-admin-edit	admin				

The Security page

NOTE

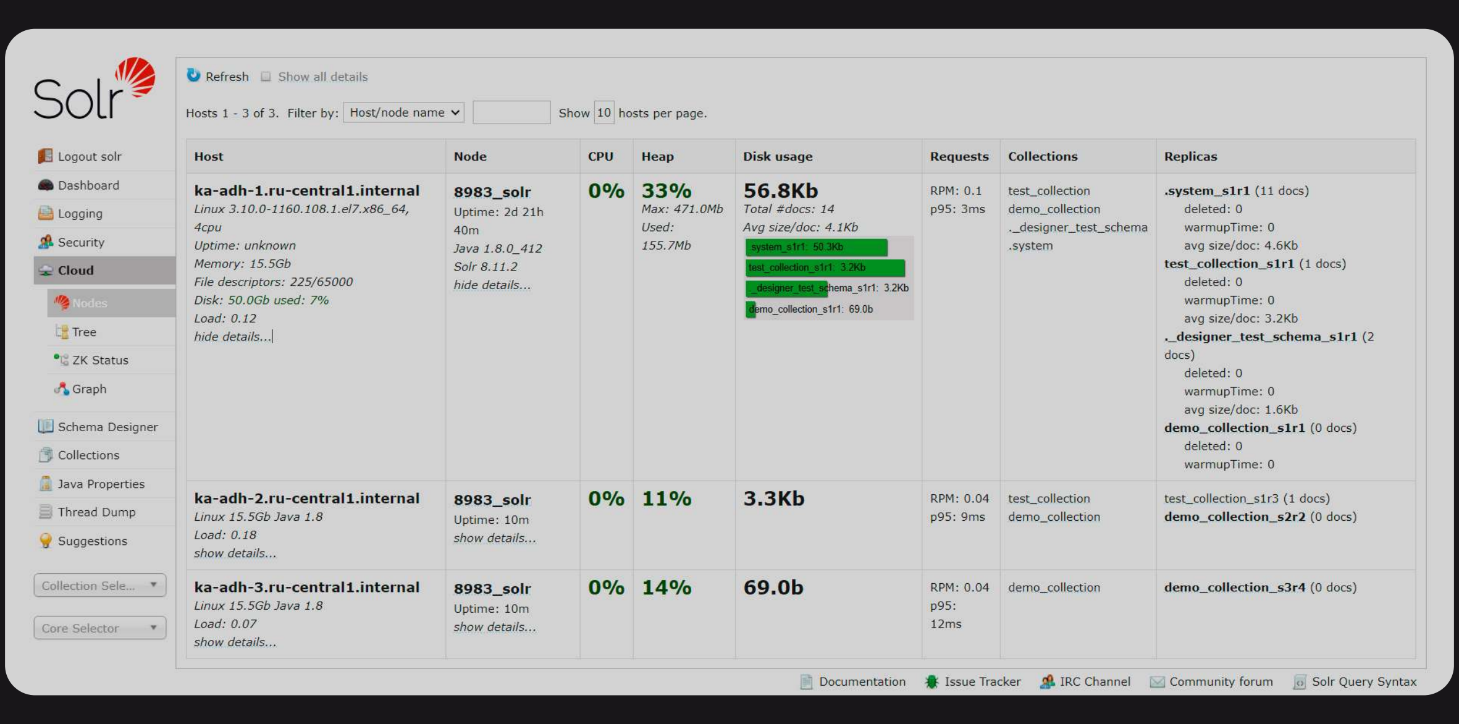
The page becomes available after you enable authentication and authorization for your Solr service.

Cloud

This tab contains several pages that reflect Solr activity in scope of an ADH cluster.

Nodes

This page displays status information about Solr servers distributed among ADH hosts.



Solr
Logout solr
Dashboard
Logging
Security
Cloud
Nodes
Tree
ZK Status
Graph
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
Refresh
Show all details
system_s1r1: 50.3Kb
test_collection_s1r1: 3.2Kb
._designer_test_schema_s1r1: 3.2Kb
demo_collection_s1r1: 69.0b
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

Hosts 1 - 3 of 3. Filter by: Host/node name Show 10 hosts per page.

Host	Node	CPU	Heap	Disk usage	Requests	Collections	Replicas
ka-adh-1.ru-central1.internal Linux 3.10.0-1160.108.1.el7.x86_64, 4cpu Uptime: unknown Memory: 15.5Gb File descriptors: 225/65000 Disk: 50.0Gb used: 7% Load: 0.12 hide details...	8983_solr Uptime: 2d 21h 40m Java 1.8.0_412 Solr 8.11.2 hide details...	0%	33% Max: 471.0Mb Used: 155.7Mb	56.8Kb Total #docs: 14 Avg size/doc: 4.1Kb	RPM: 0.1 p95: 3ms	test_collection demo_collection ._designer_test_schema .system	.system_s1r1 (11 docs) deleted: 0 warmupTime: 0 avg size/doc: 4.6Kb test_collection_s1r1 (1 docs) deleted: 0 warmupTime: 0 avg size/doc: 3.2Kb ._designer_test_schema_s1r1 (2 docs) deleted: 0 warmupTime: 0 avg size/doc: 1.6Kb demo_collection_s1r1 (0 docs) deleted: 0 warmupTime: 0
ka-adh-2.ru-central1.internal Linux 15.5Gb Java 1.8 Load: 0.18 show details...	8983_solr Uptime: 10m show details...	0%	11%	3.3Kb	RPM: 0.04 p95: 9ms	test_collection demo_collection	test_collection_s1r3 (1 docs) demo_collection_s2r2 (0 docs)
ka-adh-3.ru-central1.internal Linux 15.5Gb Java 1.8 Load: 0.07 show details...	8983_solr Uptime: 10m show details...	0%	14%	69.0b	RPM: 0.04 p95: 12ms	demo_collection	demo_collection_s3r4 (0 docs)

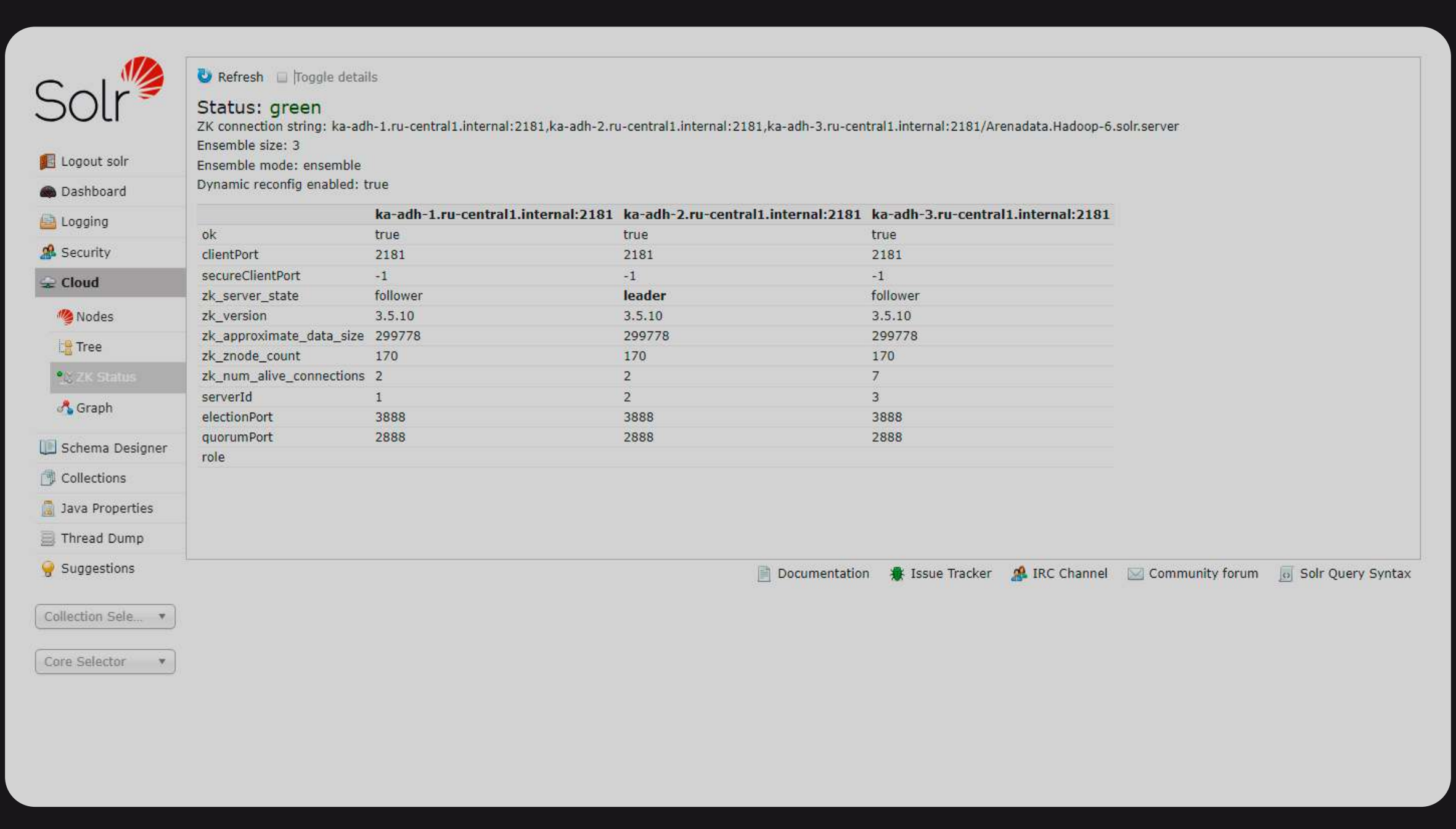
The Nodes page

Tree

This page displays the directory structure of ZooKeeper data in the form of a tree.

ZK Status

This page reflects the health status of the ZooKeeper ensemble used by Solr. Use the ***Toggle details*** option to view more details about each node.



Solr
Logout solr
Dashboard
Logging
Security
Cloud
Nodes
Tree
ZK Status
Graph
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
Refresh
Toggle details
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

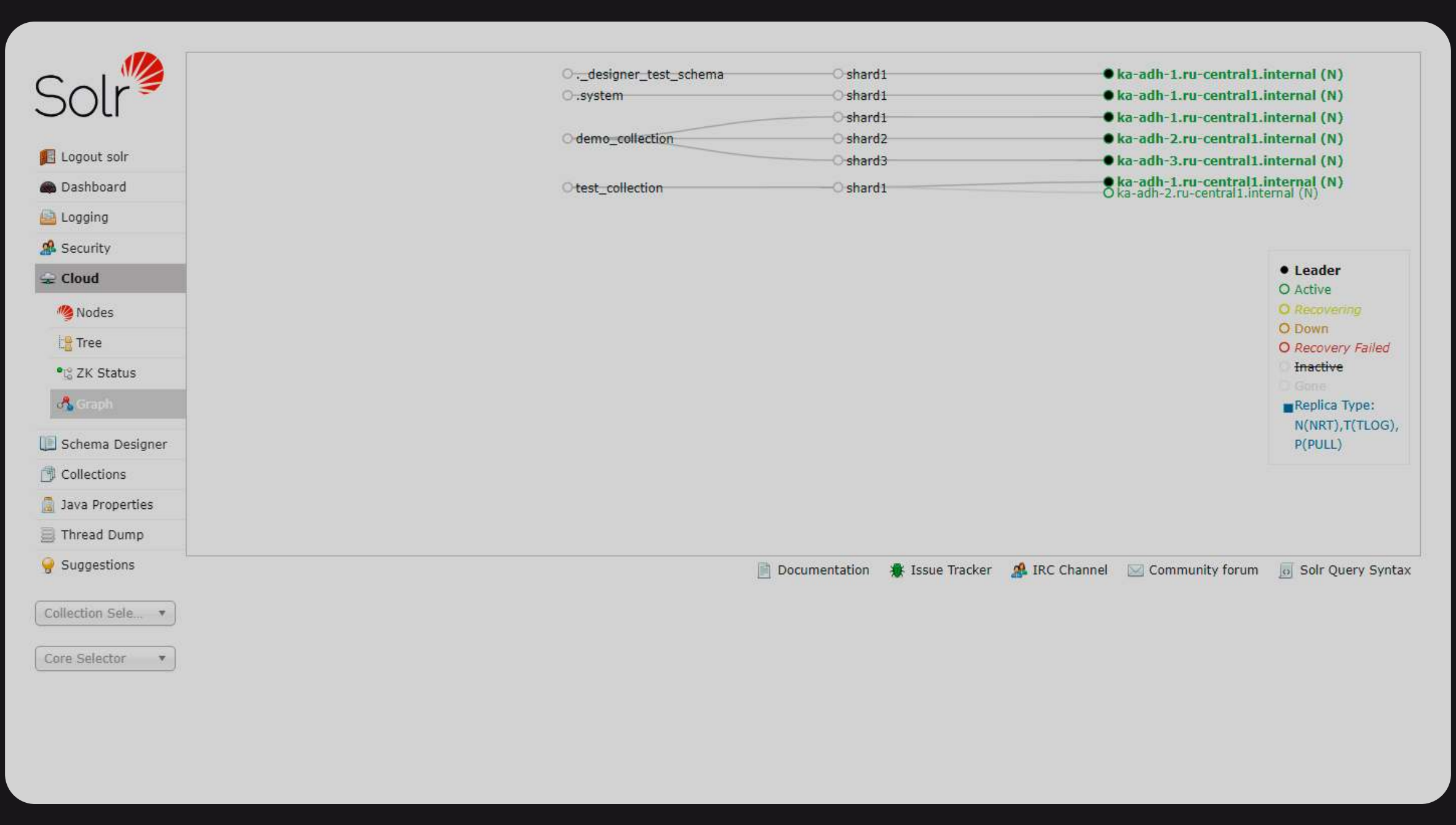
Status: green
ZK connection string: ka-adh-1.ru-central1.internal:2181,ka-adh-2.ru-central1.internal:2181,ka-adh-3.ru-central1.internal:2181/Arenadata.Hadoop-6.solr.server
Ensemble size: 3
Ensemble mode: ensemble
Dynamic reconfig enabled: true

	ka-adh-1.ru-central1.internal:2181	ka-adh-2.ru-central1.internal:2181	ka-adh-3.ru-central1.internal:2181
ok	true	true	true
clientPort	2181	2181	2181
secureClientPort	-1	-1	-1
zk_server_state	follower	**leader**	follower
zk_version	3.5.10	3.5.10	3.5.10
zk_approximate_data_size	299778	299778	299778
zk_znode_count	170	170	170
zk_num_alive_connections	2	2	7
serverId	1	2	3
electionPort	3888	3888	3888
quorumPort	2888	2888	2888
role			

The ZK Status page

Graph

This page renders a graph for each collection, including the shards that make up that collection and the nodes where replicas are hosted.

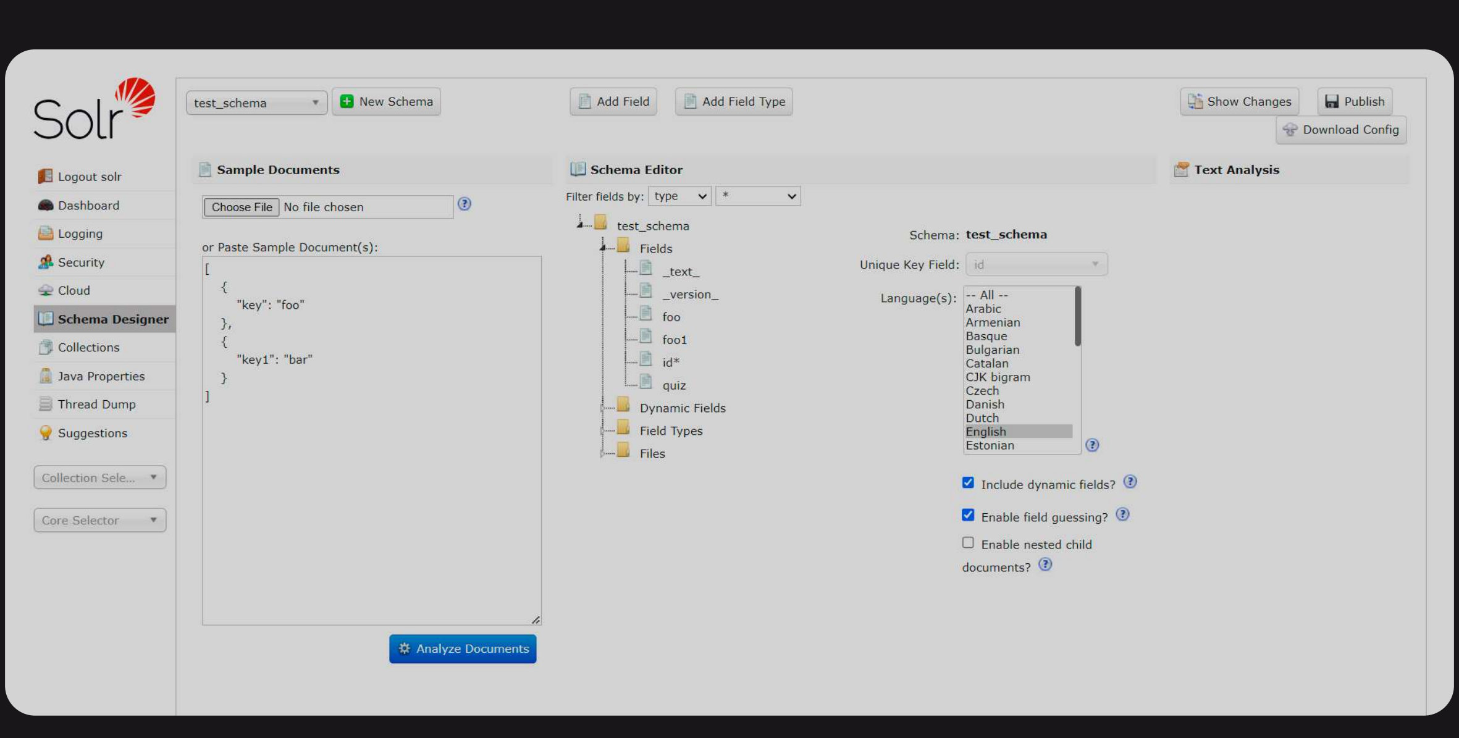


Solr
Logout solr
Dashboard
Logging
Security
Cloud
Nodes
Tree
ZK Status
Graph
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
._designer_test_schema
.system
demo_collection
test_collection
shard1
shard1
shard1
shard2
shard3
shard1
ka-adh-1.ru-central1.internal (N)
ka-adh-1.ru-central1.internal (N)
ka-adh-1.ru-central1.internal (N)
ka-adh-2.ru-central1.internal (N)
ka-adh-3.ru-central1.internal (N)
ka-adh-1.ru-central1.internal (N)
ka-adh-2.ru-central1.internal (N)
Leader
Active
Recovering
Down
Recovery Failed
Inactive
Gone
Replica Type: N(NRT),T(TLOG), P(PULL)
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

The Graph page

Schema Designer

On this page, you can define a new Solr schema. You can either specify all the schema fields manually or you can feed data to Solr to generate the schema on its basis. You can also test your newly defined schema by using the ***Query Tester*** section.



Solr
Logout solr
Dashboard
Logging
Security
Cloud
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
test_schema
New Schema
Add Field
Add Field Type
Show Changes
Publish
Download Config
Sample Documents
Choose File
Schema Editor
Text Analysis
test_schema
Fields
text
version
foo
foo1
id*
quiz
Dynamic Fields
Field Types
Files
-- All --
Arabic
Armenian
Basque
Bulgarian
Catalan
CJK bigram
Czech
Danish
Dutch
English
Estonian
Analyze Documents

No file chosen

or Paste Sample Document(s):

```
[
  {
    "key": "foo"
  },
  {
    "key1": "bar"
  }
]
```

Filter fields by: type *

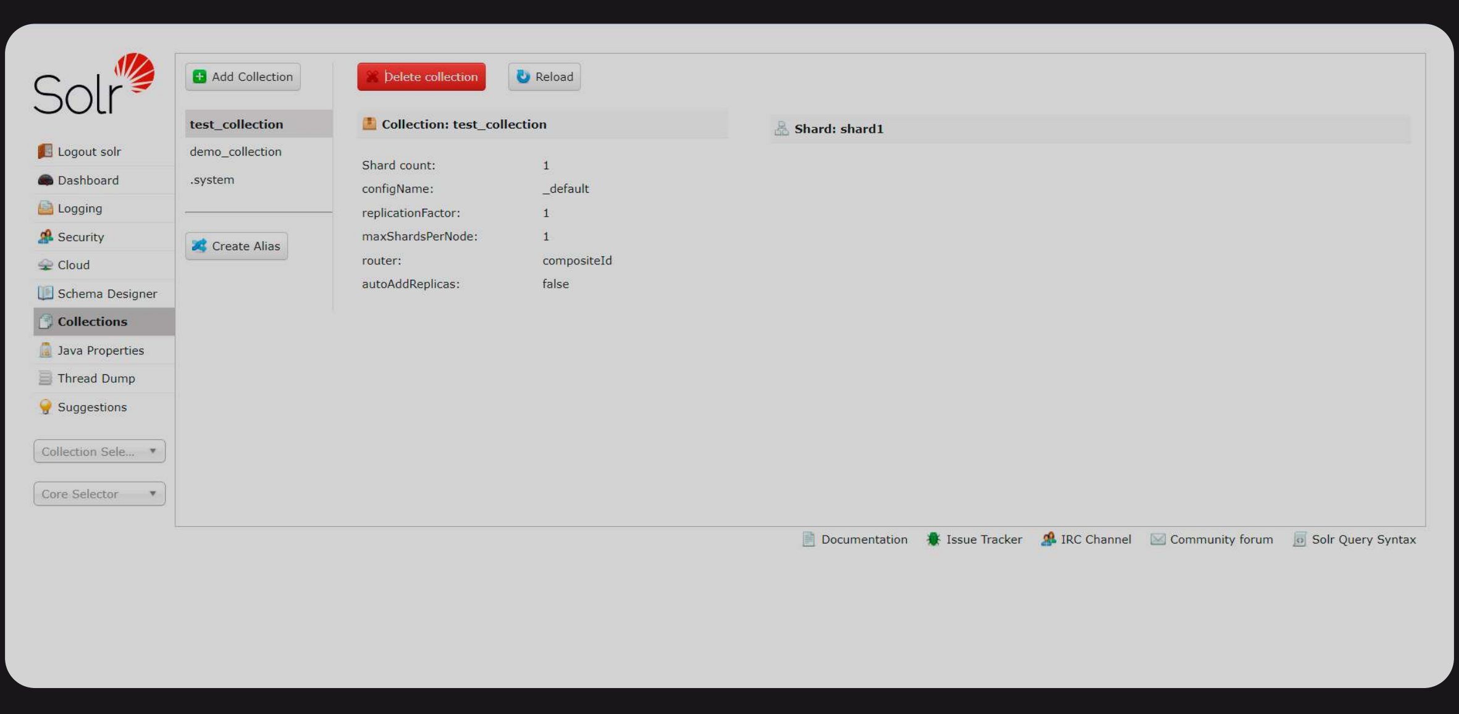
Schema: test_schema
Unique Key Field: id
Language(s): English

☑ Include dynamic fields?
☑ Enable field guessing?
☐ Enable nested child documents?

The Schema Designer page

Collections

On this page, you can manage Solr collections.



Solr
Logout solr
Dashboard
Logging
Security
Cloud
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
Add Collection
Delete collection
Reload
test_collection
demo_collection
.system
Create Alias
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

Collection: test_collection

Shard count: 1
configName: _default
replicationFactor: 1
maxShardsPerNode: 1
router: compositeId
autoAddReplicas: false

Shard: shard1

The Collections page

Java Properties

This page lists all the JVM properties used to run Solr including classpaths, file encodings, JVM memory settings, etc.

Thread Dump

This page provides expandable stacktraces allowing you to inspect the active threads of your Solr servers.

Solr collections

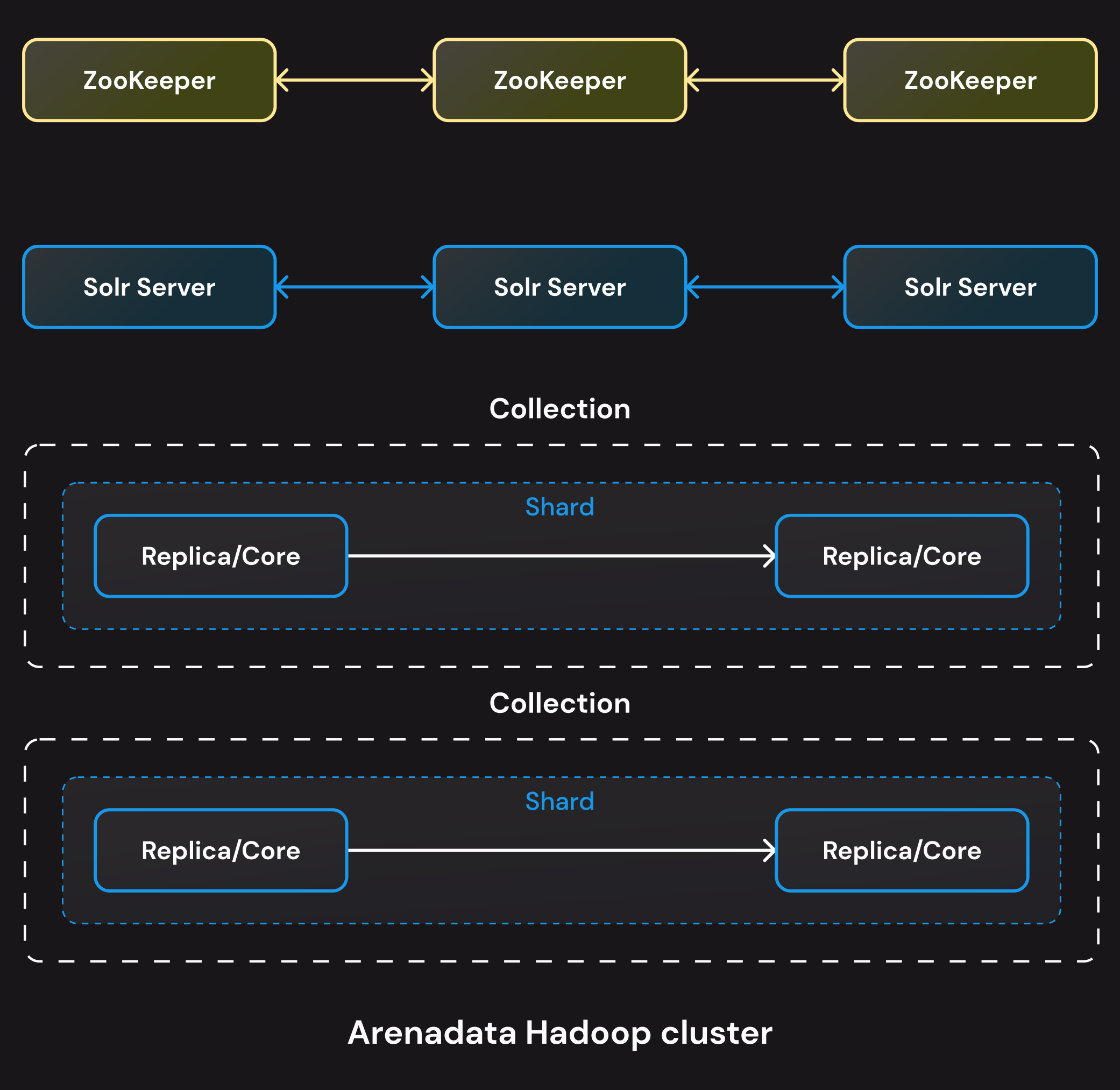
Konstantin Alpashkin

Contents

To Table of Contents

- Create collection via Solr web UI
- Create collections via CLI
  - Create a collection
  - List collections
  - Delete a collection
  - Collection health check

A **Collection** in Solr terminology is a collection of indexed documents. Each collection has its own set of configuration and schema definitions that can differ from those in other collections. We can also say that a Solr collection is a group of shards/cores that make up a single logical index. Solr splits data into shards, distributes the shards between the servers based on the number of shards per node, and maintains shards replicas to assure reliability using a replication factor that the core can establish. The logical concept is shown below.

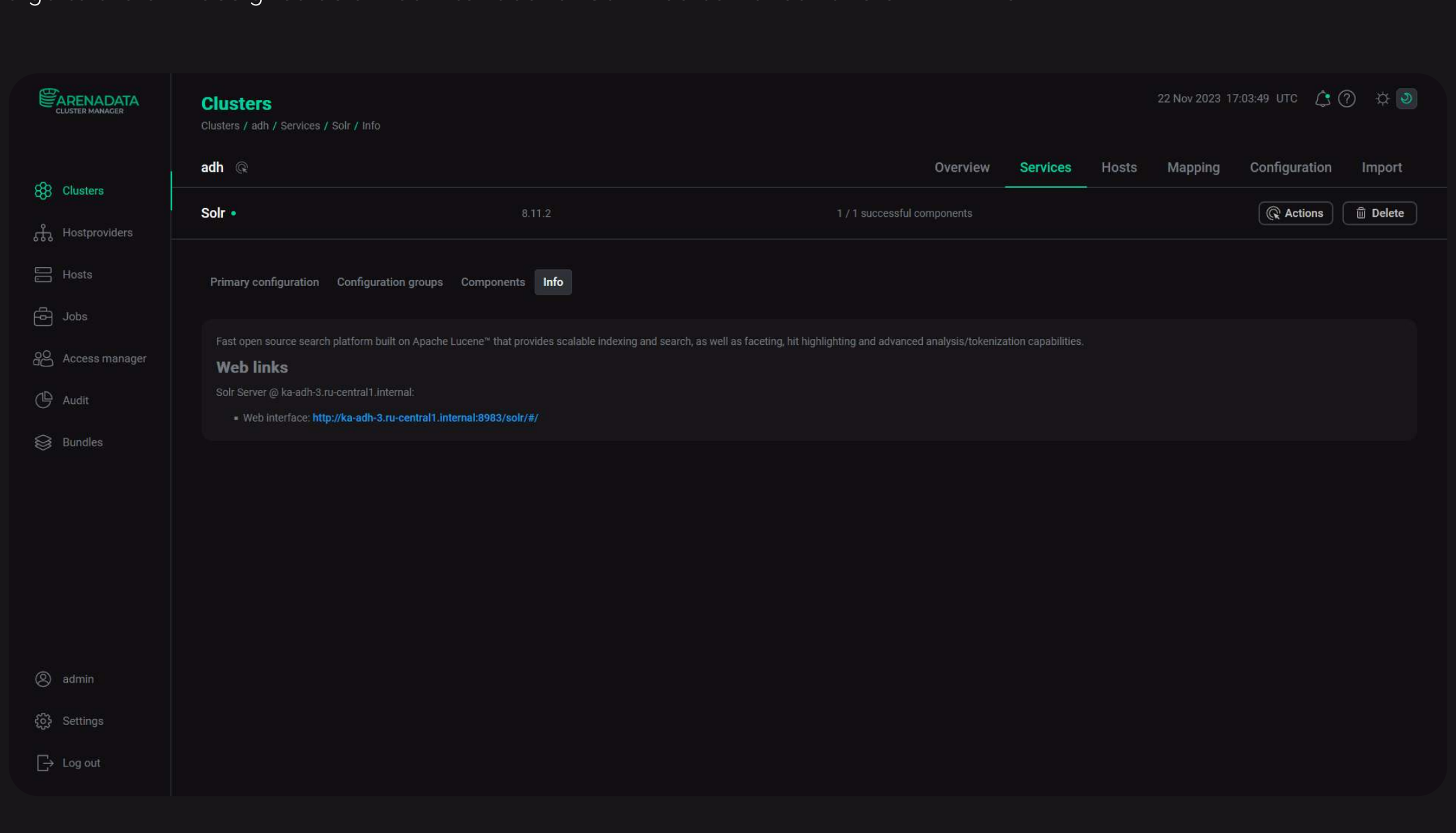


ZooKeeper
ZooKeeper
ZooKeeper
Solr Server
Solr Server
Solr Server
Collection
Shard
Replica/Core
Replica/Core
Collection
Shard
Replica/Core
Replica/Core
Arenadata Hadoop cluster

Solr collections in a cluster

Create collection via Solr web UI

1. Open your web browser and go to the URL assigned as a web interface for Solr. You can check this URL in ADCM.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / adh / Services / Solr / Info
22 Nov 2023 17:03:49 UTC
adh
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Solr
8.11.2
1 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Components
Info
admin
Settings
Log out

Fast open source search platform built on Apache Lucene™ that provides scalable indexing and search, as well as faceting, hit highlighting and advanced analysis/tokenization capabilities.

Web links

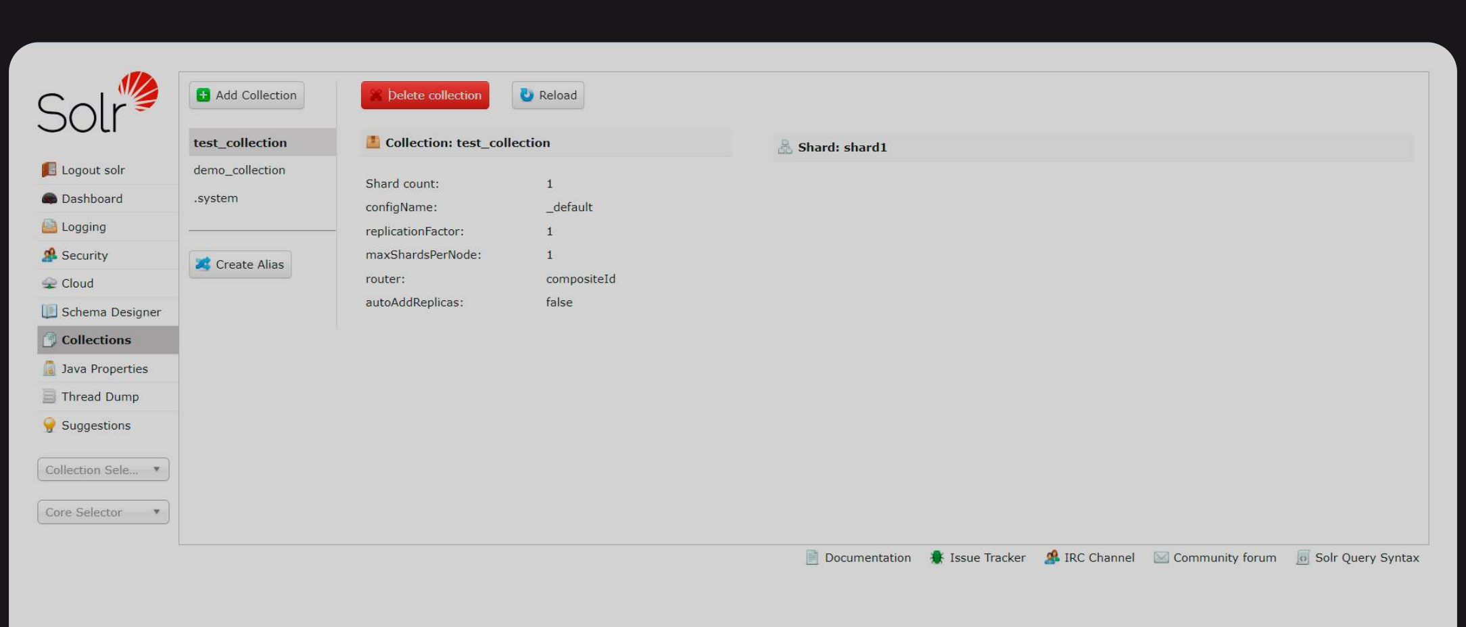
Solr Server @ ka-adh-3.ru-central1.internal:

- Web interface: http://ka-adh-3.ru-central1.internal:8983/solr/#/

Solr page in ADCM

2. The ***Collections*** tab displays a list of collections that exist in your cluster. Clicking on a collection name provides some basic metadata about how the collection is defined, its current shards and replicas, with options for adding and deleting individual replicas.
   The controls at the top of the page allow you to make various collection-related changes to your cluster, such as adding new collections or aliases, reloading or deleting a single collection, etc.

   To add a new collection, click ***Add Collection***.



Solr
Add Collection
Delete collection
Reload
Logout solr
Dashboard
Logging
Security
Cloud
Schema Designer
Collections
Java Properties
Thread Dump
Suggestions
Collection Sele...
Core Selector
test_collection
demo_collection
.system
Create Alias
Shard: shard1
Documentation
Issue Tracker
IRC Channel
Community forum
Solr Query Syntax

Collection: test_collection

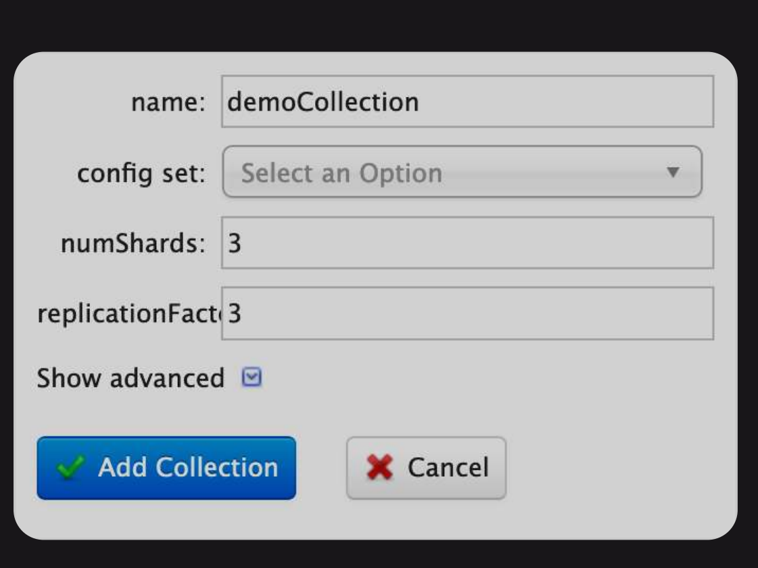
Shard count: 1
configName: _default
replicationFactor: 1
maxShardsPerNode: 1
router: compositeId
autoAddReplicas: false

Collections page

3. Specify the collection parameters in the opened pane:
   - ***name***. A user-defined name of the collection. It is a mandatory parameter. We recommend using the following rules:
     - Use only ASCII alphanumeric characters (`A-Za-z0-9`), hyphen (`-`), or underscore (`_`).
     - Avoid using the strings "shard" and "replica".
   - ***config set***. The name of an existing collection configuration. You can choose `_default` config or search for existing configurations.
   - ***numShards***. The number of shards to be created as part of the collection. It is a mandatory parameter if the `router` field is set to `Composite ID`.
   - ***replicationFactor***. The number of replicas to be created for each shard. It is an optional parameter.

   TIP

   The default value is `1`. The maximum value is the number of running Solr server nodes.

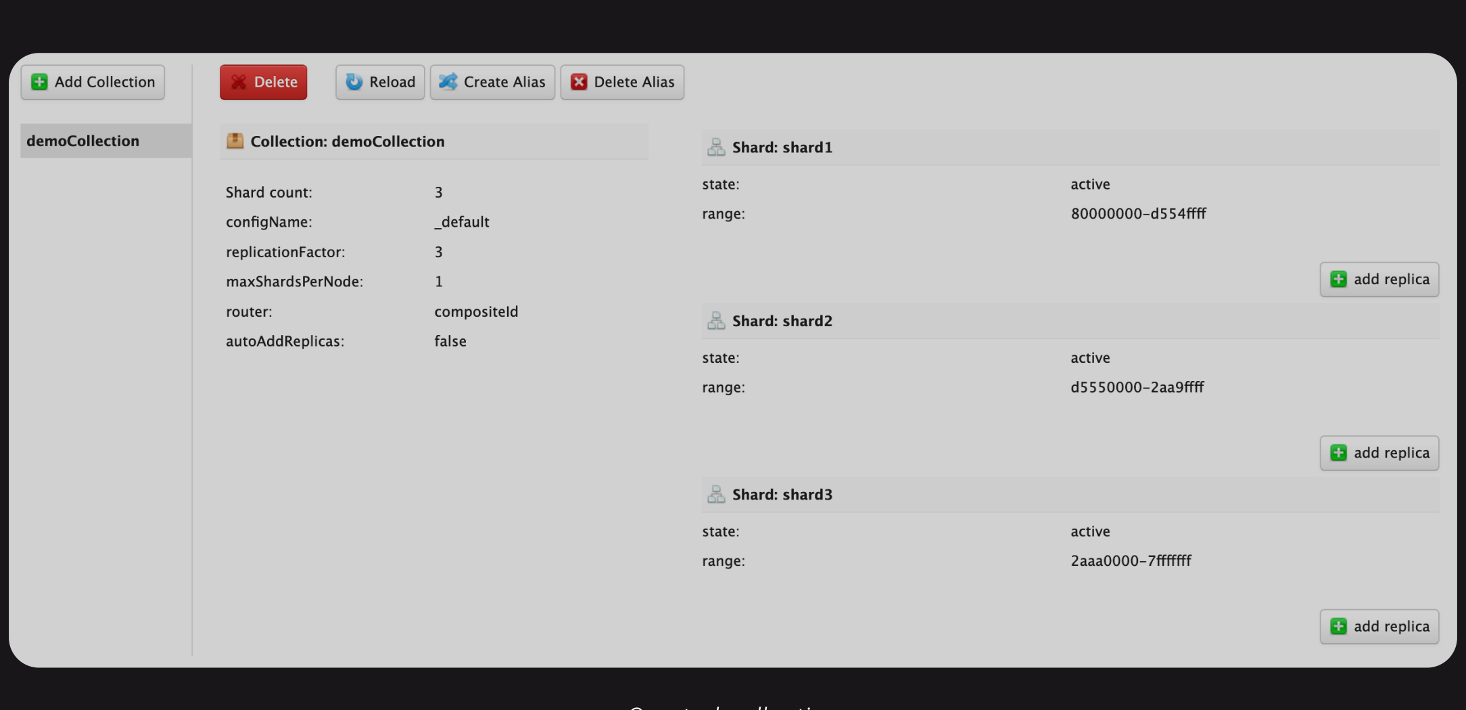


Add Collection
Cancel

name: demoCollection
config set: Select an Option
numShards: 3
replicationFact 3
Show advanced ☑

Create new collection

4. (Optional) Click ***Show advanced*** to see optional Advanced options:
   - ***router***. The name of the router that will be used. A router defines the distribution of documents among the shards. Possible values are `Composite ID` or `Implicit`. The `shards` parameter is required when using the `Implicit` router; when using the `Composite ID` router, the `numShards` parameter is required.
     - ***Implicit***. The documents do not route automatically to different shards. Whichever shard you indicate on the indexing request (or within each document) will be the destination for those documents.
     - ***Composite ID***. This router hashes the value in the `uniqueKey` field. It searches for that hash in the collection's cluster state to determine which shard will receive the document, with the additional ability to direct the routing manually.
   - ***maxShardsPerNode***. When creating Collections, the shards and/or replicas are spread across all available (i.e., live) nodes, and two replicas of the same shard will never be on the same node. If a node is not alive when the `CREATE` operation is called, it will not get any parts of the new collection, leading to too many replicas being created on a single live node. Defining `maxShardsPerNode` sets a limit on the number of replicas `CREATE` will spread to each node. If the entire collection can not fit into the live nodes, it will create no collection at all. The default value is `1`.
   - ***shards***. A comma-separated list of shard names (e.g., `shard-x,shard-y,shard-z`). This is a required parameter when using the `Implicit` router.
   - ***router.field***. If this field is specified, the router looks at the field's value in an input document to compute the hash and identify a shard instead of looking at the `uniqueKey` field. If the field specified is `null` in the document, it will reject the document.
   - ***autoAddReplicas***. When set to `true`, enables automatic addition of replicas on shared file systems. The default value is `false`.
5. After specifying all the parameters, click ***Add Collection*** to create the new collection.



Add Collection
Delete
Reload
Create Alias
Delete Alias
demoCollection
add replica
add replica
add replica

Collection: demoCollection

Shard count: 3
configName: _default
replicationFactor: 3
maxShardsPerNode: 1
router: compositeId
autoAddReplicas: false

Shard: shard1
state: active
range: 80000000-d554ffff

Shard: shard2
state: active
range: d5550000-2aa9ffff

Shard: shard3
state: active
range: 2aaa0000-7fffffff

Created collection

Create collections via CLI

This section shows basic Solr operations using CLI. You can perform operations on Solr collections using the */usr/lib/solr/bin/solr* script and `solrctl` utility.

NOTE

The examples below assume no Kerberos usage in the ADH cluster.

Create a collection

To create a new collection, use the commands:

```
$ /usr/lib/solr/bin/solr create -c <collection_name> -s <shards_num> -rf <replicas_num>
# or
$ solrctl collection --create <collection_name> -s <shards_num> -c <config_set>
```

Where:

- <collection_name> — an alphanumerical name;
- <shards_num> — a number of shards to split the collection into;
- <replicas_num> — a number of copies of each document in the collection.

Example:

```
$ /usr/lib/solr/bin/solr create -c Collection0 -s 2 -rf 2
```

List collections

To view all available collections, run:

```
$ solrctl collection --list
```

The command output looks as follows:

```
Collection0 (5)
demoCollection_1 (5)
...
```

Delete a collection

To delete a collection, use the commands:

```
$ /usr/lib/solr/bin/solr delete -c <collection_name>
# or
$ solrctl collection --delete <collection_name>
```

The command output looks as follows:

```
{
  "responseHeader":{
    "status":0,
    "QTime":236},
  "success":{
    "ka-adh-3.ru-central1.internal:8983_solr":{"responseHeader":{
        "status":0,
        "QTime":26}},
    "ka-adh-1.ru-central1.internal:8983_solr":{"responseHeader":{
        "status":0,
        "QTime":26}},
    "ka-adh-2.ru-central1.internal:8983_solr":{"responseHeader":{
        "status":0,
        "QTime":42}}}}


Deleted collection 'collection1' using command:
http://ka-adh-1.ru-central1.internal:8983/solr/admin/collections?action=DELETE&name=collection1
```

Collection health check

To check the state of a specific collection and get diagnostics information, use the command:

```
$ /usr/lib/solr/bin/solr healthcheck -c <collection_name>
```

Example:

```
$ /usr/lib/solr/bin/solr healthcheck -c Collection0
```

The output looks as follows:

```
{
  "collection":"Collection0",
  "status":"healthy",
  "numDocs":0,
  "numShards":2,
  "shards":[
    {
      "shard":"shard1",
      "status":"healthy",
      "replicas":[
        {
          "name":"core_node3",
          "url":"http://ka-adh-2.ru-central1.internal:8983/solr/Collection0_shard1_replica_n1/",
          "numDocs":0,
          "status":"active",
          "uptime":"2 days, 22 hours, 21 minutes, 49 seconds",
          "memory":"158.9 MB (%32.8) of 485 MB"},
        {
          "name":"core_node5",
          "url":"http://ka-adh-1.ru-central1.internal:8983/solr/Collection0_shard1_replica_n2/",
          "numDocs":0,
          "status":"active",
          "uptime":"2 days, 22 hours, 21 minutes, 49 seconds",
          "memory":"240.4 MB (%49.9) of 481.5 MB",
          "leader":true}]},
    {
      "shard":"shard2",
      "status":"healthy",
      "replicas":[
        {
          "name":"core_node7",
          "url":"http://ka-adh-3.ru-central1.internal:8983/solr/Collection0_shard2_replica_n4/",
          "numDocs":0,
          "status":"active",
          "uptime":"2 days, 22 hours, 21 minutes, 49 seconds",
          "memory":"196 MB (%40) of 489.5 MB",
          "leader":true},
        {
          "name":"core_node8",
          "url":"http://ka-adh-2.ru-central1.internal:8983/solr/Collection0_shard2_replica_n6/",
          "numDocs":0,
          "status":"active",
          "uptime":"2 days, 22 hours, 21 minutes, 49 seconds",
          "memory":"160.8 MB (%33.2) of 485 MB"}]}]}
```

Logging in Solr

Konstantin Alpashkin

Contents

To Table of Contents

Logs location and format

Configure logging

└ Default log4j2.xml

Loglevel API

Log slow queries

Logging UI

Under the hood, Solr uses the Log4j 2 framework for logging. The service generates several types of logs and also provides a Logging UI to view and analyze Solr logs. All the logging settings are defined through Log4j2 configuration files that can be edited via ADCM.

Logs location and format

Solr generates several types of logs to store all the system events. On the file system, these logs are stored as plain text *.log* files under the */var/log/solr/* directory (default location). Most Solr log events have the following structure:

```
2024-10-21 15:04:36.923 INFO  (coreLoadExecutor-27-thread-1-processing-n:ka-adh-3.ru-central1.internal:8983_solr)
[c:test_collection s:shard1 r:core_node2 x:test_collection_shard1_replica_n1] o.a.s.c.CoreContainer Creating SolrCore
'test_collection_shard1_replica_n1' using configuration from configset test_collection, trusted=true
```

The path to the log files is customizable and can be changed by editing the log4j2.xml configuration via ADCM. The sample */var/log/solr/* directory contents is as follows:

```
-rw-r--r-- 1 solr solr    3141 Nov  6 12:50 solr-8983-console.log ①
-rw-r--r-- 1 solr solr  703758 Nov 10 20:11 solr_gc.log.0.current ②
-rw-r--r-- 1 solr solr 4686227 Nov  9 13:00 solr.log ③
-rw-r--r-- 1 solr solr  113261 Oct 21 15:23 solr.log.1
-rw-r--r-- 1 solr solr 9054280 Oct 28 23:32 solr.log.10
-rw-r--r-- 1 solr solr  178374 Oct 21 15:52 solr.log.2
-rw-r--r-- 1 solr solr 1829020 Oct 22 19:02 solr.log.3
-rw-r--r-- 1 solr solr   40809 Oct 22 19:10 solr.log.4
-rw-r--r-- 1 solr solr   53809 Oct 22 19:14 solr.log.5
-rw-r--r-- 1 solr solr  463368 Oct 22 19:43 solr.log.6
-rw-r--r-- 1 solr solr   68754 Oct 22 19:50 solr.log.7
-rw-r--r-- 1 solr solr   57230 Oct 22 19:54 solr.log.8
-rw-r--r-- 1 solr solr  234734 Oct 22 20:31 solr.log.9
-rw-r--r-- 1 solr solr       0 Oct 21 14:21 solr_slow_requests.log ④
```

① Stores Solr's console output redirected to STDOUT.

② Stores details on GC invocations by Solr.

③ The main log files reflecting Solr Server's activity. By default, the *solr.log.{n}* files get rolled over upon reaching certain size.

④ Stores details about slow queries in your system.

Configure logging

With the use of Log4j2, configuring logging in Solr assumes editing the corresponding Log4j2 files:

- *log4j2.xml* — the main configuration file responsible for logging all Solr Server's events.
- *log4j2-console.xml* — the configuration file responsible for logging the console client's activity.

Both files are available at */usr/lib/solr/server/resources*, and Solr reads these files during the startup. ADCM provides an editor where you can modify the contents of these files. To edit the Log4j2 configuration using ADCM:

1. In ADCM, go to ***Clusters*** → ***<clusterName>*** → ***Services*** → ***Solr*** → ***Components*** → ***Solr Server*** → ***Primary configuration***.
2. Click the Log4j2 configuration file you want to edit (for example, ***log4j2.xml***).
3. Update the configuration. Below you can find the default *log4j2.xml* configuration with major configuration parameters highlighted. The detailed information about supported Log4j2 properties can be found in Log4j 2 documentation.
4. Save the Solr service configuration.
5. Restart the Solr service.

Default log4j2.xml

Below you can find the default *log4j2.xml* configuration used by Solr. The major configuration parameters are highlighted with comments.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Configuration for asynchronous logging -->
<Configuration>
<Appenders>
    <Console name="STDOUT" target="SYSTEM_OUT"> ①
      <PatternLayout>
        <Pattern> ②
          %maxLen{%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p (%t) [%X{collection} %X{shard} %X{replica} %X{core}] %c{1.} %m%notEmpty{
=>%ex{short}}}{10240}%n
        </Pattern>
      </PatternLayout>
    </Console>

    <RollingRandomAccessFile ③
        name="MainLogFile"
        fileName="/var/log/solr/solr.log" ④
        filePattern="/var/log/solr/solr.log.%i" > ⑤
      <PatternLayout>
        <Pattern>
          %maxLen{%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p (%t) [%X{collection} %X{shard} %X{replica} %X{core}] %c{1.} %m%notEmpty{
=>%ex{short}}}{10240}%n
        </Pattern>
      </PatternLayout>
      <Policies>
        <OnStartupTriggeringPolicy /> ⑥
        <SizeBasedTriggeringPolicy size="32 MB"/> ⑦
      </Policies>
      <DefaultRolloverStrategy max="10"/> ⑧
    </RollingRandomAccessFile>

    <RollingRandomAccessFile ⑨
        name="SlowLogFile"
        fileName="/var/log/solr/solr_slow_requests.log"
        filePattern="/var/log/solr/solr_slow_requests.log.%i" >
      <PatternLayout>
        <Pattern>
          %maxLen{%d{yyyy-MM-dd HH:mm:ss.SSS} %-5p (%t) [%X{collection} %X{shard} %X{replica} %X{core}] %c{1.} %m%notEmpty{
=>%ex{short}}}{10240}%n
        </Pattern>
      </PatternLayout>
      <Policies>
        <OnStartupTriggeringPolicy />
        <SizeBasedTriggeringPolicy size="32 MB"/>
      </Policies>
      <DefaultRolloverStrategy max="10"/>
    </RollingRandomAccessFile>

  </Appenders>
  <Loggers> ⑩
    <AsyncLogger name="org.apache.hadoop" level="warn"/>
    <AsyncLogger name="org.apache.solr.update.LoggingInfoStream" level="off"/>
    <AsyncLogger name="org.apache.zookeeper" level="warn"/>
    <AsyncLogger name="org.apache.solr.core.SolrCore.SlowRequest" level="info" additivity="false">
      <AppenderRef ref="SlowLogFile"/>
    </AsyncLogger>

    <AsyncRoot level="info">
      <AppenderRef ref="MainLogFile"/>
      <AppenderRef ref="STDOUT"/>
    </AsyncRoot>
  </Loggers>
</Configuration>

<!-- Configuration for synchronous logging
there _may_ be a very small window where log messages will not be flushed
to the log file on abnormal shutdown. If even this risk is unacceptable, use
the configuration below
-->
<!--Configuration>
    ...
</Configuration-->
```

① Defines the console appender named `STDOUT` that writes logs to STDOUT.

② Specifies a pattern for each log record written to the log file by the appender. For details on `PatternLayout` syntax, see Log4j2 layouts.

③ Defines the RollingRandomAccessFile appender that writes main Solr logs to files, which are rotated based on size.

④ Specifies a log file name.

⑤ Specifies a log file name pattern for the rotated log files when a rollover occurs.

⑥ The policy triggers a rollover on every Solr startup.

⑦ The policy triggers a rollover when the file reaches 32 MB.

⑧ Defines the maximum number of archived log files to keep. When this limit is reached, the oldest files are deleted.

⑨ Defines the `RollingRandomAccessFile` appender that writes information about slow requests to files, rotated based on size.

⑩ Defines a list of all loggers that log Solr events.

> **NOTE**
>
> By default, the *log4j2.xml* file has two configurations — for synchronous and asynchronous logging. The synchronous configuration is disabled by default, as the Log4j's asynchronous logging provides better performance by executing all I/O operations in a separate thread. However, in case of an abnormal Solr shutdown, there is a chance that some log messages cannot be flushed to the disk, and thus, may get lost. If this is unacceptable, use the synchronous Logj2 configuration. For this, in *log4j2.xml*, comment the asynchronous configuration part, then uncomment the synchronous configuration block, and restart Solr.

Loglevel API

Solr exposes a REST endpoint to update the logging configuration for Solr components. For example, to set a new logging level for a single Solr component (`org.apache.solr.security`), submit the following request:

```
$ curl http://ka-adh-3.ru-central1.internal:8983/solr/admin/info/logging --data-binary "set=org.apache.solr.security:WARN"
```

A request to set a logging level for several Solr components at once (for example, all `org.apache.zookeeper` children) looks as follows:

```
$ curl http://ka-adh-3.ru-central1.internal:8983/solr/admin/info/logging --data-binary "set=org.apache.zookeeper:WARN"
```

As a response, Solr returns a JSON document, reflecting the current Log4j2 levels for each Solr component. A sample response is below.

Sample response

```
{
  "responseHeader":{
    "status":0,
    "QTime":1},
  "watcher":"Log4j2",
  "levels":["ALL",
    "TRACE",
    "DEBUG",
    "INFO",
    "WARN",
    "ERROR",
    "FATAL",
    "OFF"],
  "loggers":[{
      "name":"root",
      "level":"INFO",
      "set":true},
    {
      "name":"com",
      "level":null,
      "set":false},
    {
      "name":"com.codahale",
      "level":null,
      "set":false},
    {
      "name":"com.codahale.metrics",
      "level":null,
      "set":false},
    {
      "name":"com.codahale.metrics.jmx",
      "level":null,
      "set":false},
    {
      "name":"com.codahale.metrics.jmx.JmxReporter",
      "level":"INFO",
      "set":true},
    {
      "name":"org",
      "level":"INFO",
      "set":true},
    {
      "name":"org.apache",
      "level":null,
      "set":false},
    {
      "name":"org.apache.calcite",
      "level":null,
      "set":false},
    {
      "name":"org.apache.calcite.avatica",
      "level":null,
      "set":false},
    {
      "name":"org.apache.calcite.avatica.remote",
      "level":null,
      "set":false},
    {
      "name":"org.apache.calcite.avatica.remote.Driver",
      "level":"INFO",
      "set":true},
    {
      "name":"org.apache.hadoop",
      "level":"WARN",
      "set":true},
    ....
```

Log slow queries

For search-intensive applications, logging every single query can generate a large amount of excessive logs, and depending on the volume, may potentially impact the performance. If you need the detailed logs (for example, for automatic log analysis or early detection of possible issues), then logging every query may be useful. On the other hand, if you are only concerned about warnings and errors related to requests, then you can set the log verbosity to `WARN`. However, this brings a potential problem — Solr will not raise alerts for queries that take long, since such queries are still logged at the `INFO` level.

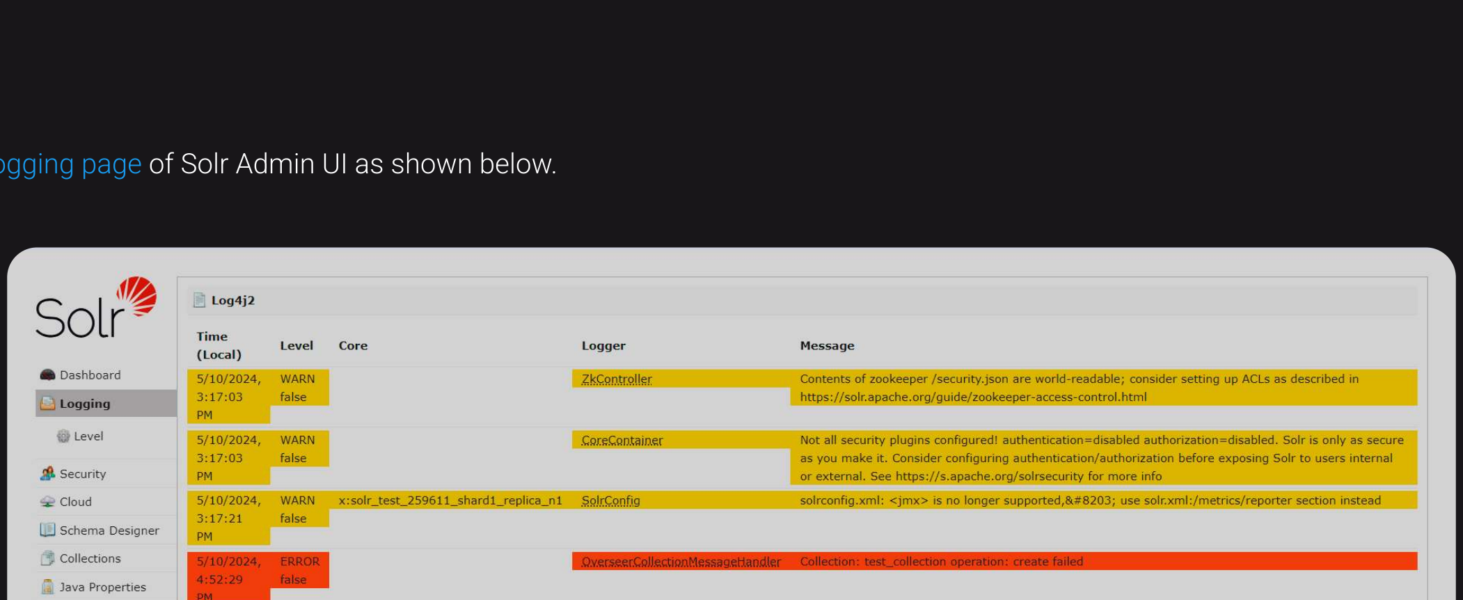
If you want to use the `WARN` verbosity level, but still be able to detect slow queries, Solr provides a way to set a latency threshold above which a request is considered slow. Such long-running requests will be logged at the `WARN` level to help you identify slow queries in your application. To enable this feature, add the slowQueryThresholdMillis parameter in the query section of *solrconfig.xml*. For example:

```
<requestHandler name="/select" class="solr.SearchHandler">
  <int name="slowQueryThresholdMillis">3000</int>
</requestHandler>
```

Any queries that take longer than the specified threshold will be logged as slow at the `WARN` level. These logs are written to a separate log file, which defaults to *solr_slow_requests.log*.

Logging UI

You can view Solr logs on the Logging page of Solr Admin UI as shown below.

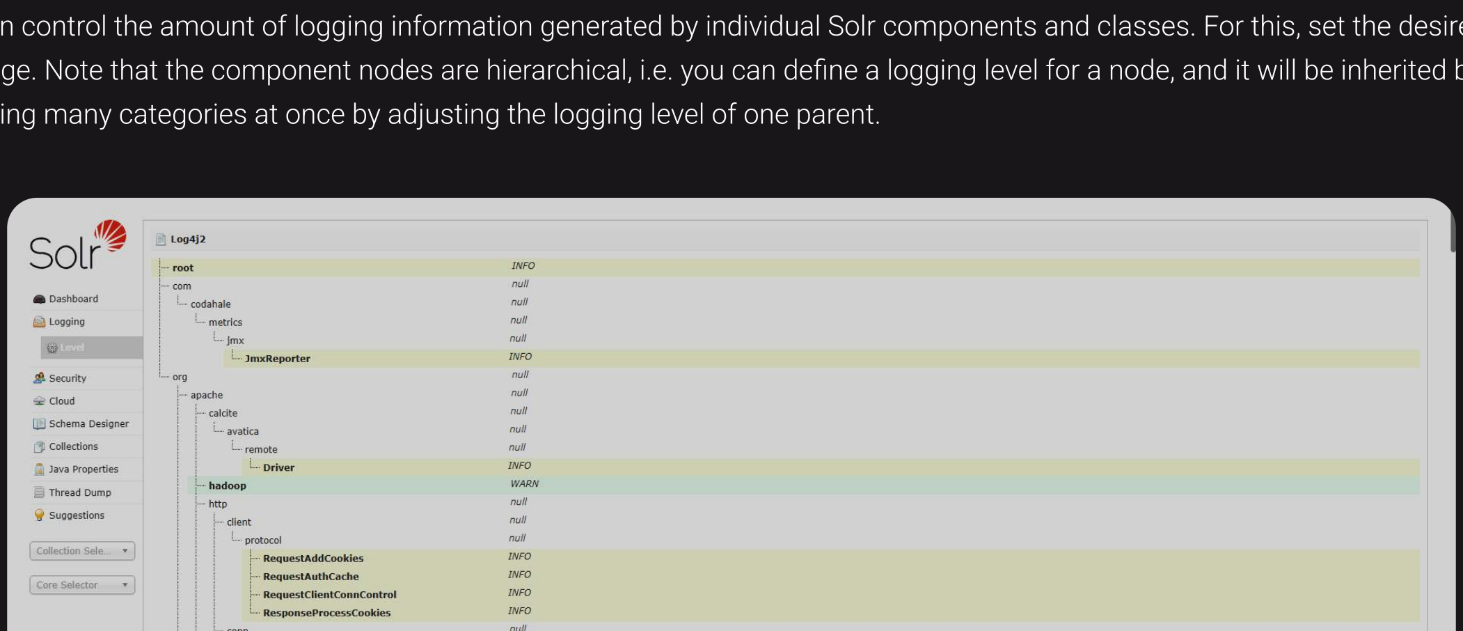


Time (Local)	Level	Core	Logger	Message
5/10/2024, 3:17:03 PM	WARN false		ZkController	Contents of zookeeper /security.json are world-readable; consider setting up ACLs as described in https://solr.apache.org/guide/zookeeper-access-control.html
5/10/2024, 3:17:03 PM	WARN false		CoreContainer	Not all security plugins configured! authentication=disabled authorization=disabled. Solr is only as secure as you make it. Consider configuring authentication/authorization before exposing Solr to users internal or external. See https://s.apache.org/solrsecurity for more info
5/10/2024, 3:17:21 PM	WARN false	x:solr_test_259611_shard1_replica_n1	SolrConfig	solrconfig.xml: <jmx> is no longer supported,&#8203; use solr.xml:/metrics/reporter section instead
5/10/2024, 4:52:29 PM	ERROR false		OverseerCollectionMessageHandler	Collection: test_collection operation: create failed
5/10/2024, 4:52:37 PM	WARN false	x:test_collection_shard1_replica_n1	SolrConfig	solrconfig.xml: <jmx> is no longer supported,&#8203; use solr.xml:/metrics/reporter section instead

Last Check: 5/10/2024, 5:52:29 PM

The Logging page

Also, using the ***Level*** tab, you can control the amount of logging information generated by individual Solr components and classes. For this, set the desired logging level for the Solr components available on the page. Note that the component nodes are hierarchical, i.e. you can define a logging level for a node, and it will be inherited by all of its children components. This allows changing many categories at once by adjusting the logging level of one parent.



```
root                                  INFO
com                                   null
  codahale                            null
    metrics                           null
      jmx                             null
        JmxReporter                   INFO
org                                   null
  apache                              null
    calcite                           null
      avatica                         null
        remote                        null
          Driver                      INFO
    hadoop                            WARN
    http                              null
      client                          null
        protocol                      null
          RequestAddCookies           INFO
          RequestAuthCache            INFO
          RequestClientConnControl    INFO
          ResponseProcessCookies      INFO
      conn                            null
        ssl                           null
          AllowAllHostnameVerifier    INFO
          BrowserCompatHostnameVerifier INFO
          DefaultHostnameVerifier     INFO
          SSLConnectionSocketFactory  INFO
          StrictHostnameVerifier      INFO
      headers                         INFO
      impl                            null
        auth                          null
          HttpAuthenticator           INFO
        client                        null
          DefaultRedirectStrategy     INFO
          InternalHttpClient          INFO
          ProxyAuthenticationStrategy INFO
          TargetAuthenticationStrategy INFO
```

Logging levels

> **NOTE**
>
> Using this page, you can only change logging settings for a running Solr instance. These setting are not persisted for the next Solr run.

Solr performance tuning

Konstantin Alpashkin

Contents

To Table of Contents

- Overview
- Index configuration
  - Schema design
  - Index size
  - Sharding and replication
- Memory and JVM parameters
- Commit parameters
- Query optimization

Overview

This article provides optimization techniques and best practices that can improve your Solr performance.

> There is no definitive guide on improving Solr performance that would work for all deployments. Solr has a lot of moving parts that affect each other, so changing one configuration parameter can affect the work of others. When tuning your Solr performance, it is very important to iteratively test your Solr setup and make only a few configuration updates between each test. This way you can be confident that you are moving in the right direction and can identify the bottlenecks earlier.

All the recommendations and optimizations described further can be divided into groups:

- Index configuration
- Memory and JVM parameters
- Commit parameters
- Query optimization

Index configuration

A proper indexing configuration allows Solr to store documents more efficiently in the Lucene index files and results in faster queries when the documents are retrieved from the index.

Schema design

- Choose the field types that best match the data type stored in the fields. For example, storing dates in `DatePointField` fields rather than `StrField` can improve the performance both at index and query time.
- Minimize the use of `text_general` (`TextField`) fields to store simple strings. Indexing and searching using fields of this type triggers tokenization/analysis processing chains for each request, which consumes computing power. Instead, prefer field-specific types like `string` (`StrField`) to store strings without any additional processing.
- Minimize the use of dynamic fields. Dynamic fields allow Solr to index data types that were not explicitly defined in the schema. However, such flexibility brings additional overhead. For example, using dynamic fields is a good choice for sparsely populated fields or when the field type can vary.
- Use docValues for fields that are frequently used in sorting or faceting.
- When using custom fields, specify the field properties taking into account how the field will be used in future for querying. The basic recommendations are as follows:
  - Avoid unnecessary `multiValued` fields. This parameter indicates that a document might contain multiple values for a field type.
  - Do not use `stored=true` unless the field is needed for retrieval.
  - Use `indexed=false` for fields that are not part of the search or filtering process.

Index size

- Avoid adding and storing redundant and duplicate information to the index. For more information on catching duplicates, see Deduplication in Solr.
- Change the merging policy that defines a strategy for merging smaller Lucene segment files into one. By default, `TieredMergePolicy` merges segments of approximately equal size, subject to an allowed number of segments per tier.

Sharding and replication

- Distribute your data across multiple shards residing on different ADH hosts to handle more queries simultaneously. In a multi-sharded Solr index, the documents are distributed among shards so that every document is contained in exactly one shard.
- Use replication for high availability and better fault tolerance.

Memory and JVM parameters

One of the crucial factors affecting Solr performance is RAM. Solr requires sufficient memory for Java heap and additionally needs the free memory for the OS-level page cache operations.

- Allocate sufficient Java heap memory for every Solr Server node. For this, use the ***Solr Server Heap Memory*** parameter in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Solr*** → ***Primary configuration***). However, simply allocating a larger Java heap does not guarantee speeding up your Solr setup. Your heap should be able to accommodate all caches and internal objects while leaving enough space to avoid frequent GC runs. Also, there must be host memory left to handle OS page caching activities to cache Solr index data.
- Allocate JVM heap size equal to 40-70% of the host RAM, a portion of memory should always remain free. Apart from heap, Solr utilizes free direct memory (off-heap) for faster caching/flushing document data to index segment files. Too large heap size can leave little memory for these operations and make them a bottleneck.
- Use the garbage collection logs to detect abnormalities and monitor memory usage. By default, Solr stores garbage collection logs in */var/log/solr/solr_gc.log.** files. For more information, see Logging in Solr.

Commit parameters

In Solr, commits are used to make updated documents visible for search queries. Until a commit is made, updated documents are unavailable for searching by other Solr clients. Solr performs two types of commits:

- **Hard commit**. Makes updates available for searching by calling `fsync` on Lucene index files and flushing bits to disk. After a hard commit, Solr truncates the transaction log and opens a new one.
- **Soft commit**. Makes index changes visible without calling `fsync` on the Lucene segment files. After a soft commit, updated document data is written to a transaction log but is not yet stored in a segment file.

Soft commits are a good choice for achieving near-real-time search results, as they make documents visible without waiting for a costly hard commit. Hard commits are essential for durability and ensuring that documents are not lost when a node goes down unexpectedly. You can change the commit settings in *solrconfig.xml* (`<autoCommit>` and `<autoSoftCommit>` sections).

Consider the following tips on tuning the commit mechanism:

- Avoid too frequent hard commits as their massive load on CPU and disk I/O can heavily impact the overall performance. Disk I/O can be the major bottleneck during hard commits if enough memory is not available for OS page caching operations. When choosing a value for automatic hard commits, you should balance between as infrequent commits as possible and meeting the time SLA for any updates to be visible to Solr clients.
- Although Solr soft commits are cheaper than hard commits, they are still not CPU/RAM-free. It is recommended to auto-run soft commits more frequently than hard commits, but still balance between near real-time update delay and indexing speed.

Query optimization

- Complex queries can slow down Solr. If your query is complex and runs for too long, try splitting the query in two shorter ones.
- Change the query caching behavior. Solr automatically pre-populates caches for frequent and similar queries. The time taken for this is called cache warm-up time. However, when data is written to Solr, all caches get invalidated after each commit and have to be warmed up again, which consumes resources. Use the `autowarmCount=0` parameter for `<filterCache>`, `<queryResultCache>`, and `<documentCache>` settings in your *solrconfig.xml* to reduce cache warmup cycles.
- Do not use too many filters within one search request.
- Avoid using expensive operations like wildcard searches at the beginning of a term.
- Use lazy field loading (the `<enableLazyFieldLoading>` parameter) to allow Solr to load fields that are not directly requested later as needed. This can improve the performance if most queries request only a small subset of fields.

Solr service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Solr service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Solr service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Solr components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM
Manage Ranger plugin	Enables or disables Ranger plugin for Solr
Update collection expiration	Updates the expiration time (the `_expire_at_` field) for a specific Solr collection. When you run this action, the following options are available: ■ ***Collection name*** — the Solr collection to update. ■ ***Expire at*** — the timestamp string that sets the expiration time for documents indexed in Solr. For example, `2023-07-28T18:05:03.371Z`. ■ ***Batch size*** — the number of documents to process per one batch.
Validate configuration	Validates Solr configuration (OS family compatibility, port conflicts) on all hosts with Solr components

spark3-submit & spark3-shell

Konstantin Alpashkin

Contents

To Table of Contents

spark3-submit

spark3-shell

There are two ways to launch Spark jobs on your cluster. You can run it on any host with the Spark3 client:

- `spark3-submit`. Submit interactive statements through the Scala, Python, or R shell.
- `spark3-shell`. Create a Spark application that runs interactively or in batch mode, using Scala, Python, or R.

Alternatively, you can use Livy to submit and manage Spark applications on a cluster. Livy is a Spark service that allows local and remote applications to interact with Apache Spark over an open source REST interface. Livy offers additional multi-tenancy and security functionality.

spark3-submit

To launch Spark applications on a cluster, you can use the *spark3-submit* script in the Spark *bin* directory. It can use all of the Spark’s supported cluster managers through a uniform interface so you don’t have to configure your application specially for each cluster manager.

If your code depends on other projects, you have to package the dependencies alongside with your application to run the application on a cluster with Spark. To do this, create an assembly jar (or uber-jar) containing your code and its dependencies. You can use *sbt* or *Maven* assembly plugins. When creating assembly jars, list Spark and Hadoop as provided dependencies; those need not be bundled since the cluster manager provides them during the runtime.

When you have built the uber-jar, you can run it using the *bin/spark3-submit* script. This script takes care of setting up the classpath with Spark and its dependencies, and can support different cluster managers and deploy modes that Spark supports.

You can find more information at the Submitting Applications page of the Spark documentation.

spark3-shell

Spark shell provides a simple way to learn the API, as well as a powerful tool to analyze the data interactively. You can use the API interactively by launching a shell for Scala (*spark3-shell* script). Note that each interactive shell automatically creates `SparkContext` in a variable called `sc`.

It is available in Scala, that runs on the Java VM and is a good way to use existing Java libraries. Start it by running the following command in the Spark directory:

```
$ ./bin/spark3-shell
```

Example of launching jobs with a Dataset

Konstantin Alpashkin

Contents

To Table of Contents

Dataset
Preparations
Launching calculation
View results

You can use this sample program to validate your Spark framework and to explore how to run Spark jobs using `spark3-submit`. In this sample program, we collect statistics on the criminal situation in different areas of Boston. The dataset used in the example can be downloaded from this dataset source.

Dataset

The dataset includes records starting from June 14, 2015 up to September 3, 2018. Each row represents a crime report including the type of crime, date and time, and location.

We have two files for processing:

- *crime.csv* — the main file with crime records;
- *offense_codes.csv* — the data file that contains crime codes.

After the processing we get the following information:

- the most common types of crime;
- the distribution of crimes by districts;
- the frequency of different types of crimes.

Preparations

You can read Scala documentation on the creation of a Scala project using the Scala build tool. You can build the project on your cluster or on any other machine.

> **NOTE**
>
> If you create the project locally, do not forget to copy the jar file to the Spark master host.

Using Spark, we aggregate around all districts (`district` field) with the following metrics:

- `crimes_total` — the total crimes number in the specified district.
- `crimes_monthly` — the median crimes number per month in the specified district.
- `frequent_crime_types` — the three most frequent crime types in the entire history of observations in the specified district, separated by a comma with a single space (`, `), arranged in the descending order of frequency.
- `crime_type` — the first part of the `NAME` from the *offense_codes.csv* table, split by the `-` separator (for example, if `NAME` is `BURGLARY - COMMERICAL - ATTEMPT`, then `crime_type` is `BURGLARY`).
- `lat` — the latitude of the district coordinates, average for all incidents.
- `lng` — the longitude of the district coordinates, average for all incidents.

You can use the following code samples to build your own Scala project.

Scala main code sample

```scala
package com.example

import com.example.entity.{Crime, OffenseCode}
import org.apache.spark.sql._
import org.apache.spark.sql.functions._

object BostonCrimesMap extends App {

    val crimeFilepath = args(0)
    val offenseCodesFilePath = args(1)
    val resultFilePath = args(2)

  val spark = SparkSession
    .builder()
    .master("local[*]")
    .appName("boston_crimes")
    .getOrCreate()

  import spark.implicits._

  val crimes = spark
    .read
    .option("header", "true")
    .option("inferSchema", "true")
    .csv(crimeFilepath)
    .as[Crime]


  val offense_codes = spark
    .read
    .option("header", "true")
    .option("inferSchema", "true")
    .csv(offenseCodesFilePath)
    .withColumn("CRIME_TYPE", trim(substring_index($"NAME", "-", 1)))
    .as[OffenseCode]


  val offense_codes_br = spark.sparkContext.broadcast(offense_codes)

  val filteredCrimes = crimes
    .filter($"DISTRICT".isNotNull).cache()

  val crimesWithOffenceCodes = filteredCrimes
    .join(offense_codes_br.value, filteredCrimes("OFFENSE_CODE") === offense_codes_br.value("CODE"))
    .select("INCIDENT_NUMBER", "DISTRICT", "MONTH", "Lat", "Long", "CRIME_TYPE").cache()

  val crimesDistrictAnalytics = filteredCrimes
    .groupBy($"DISTRICT")
    .agg(expr("COUNT(INCIDENT_NUMBER) as crimes_total"),
      expr("AVG(Lat) as lat"),
      expr("AVG(Long) as lng")
    )


  val crimesByDistrictByMonth = filteredCrimes
    .groupBy($"DISTRICT", $"MONTH")
    .agg(expr("count(INCIDENT_NUMBER) as CRIMES_CNT")).createOrReplaceTempView("crimesByDistrictByMonth")

  val crimesDistrictMedian = spark.sql(
    "select " +
      " DISTRICT" +
      " ,percentile(CRIMES_CNT, 0.5) as crimes_monthly " +
      " from crimesByDistrictByMonth" +
      " group by DISTRICT")


  val crimesByDistrictByCrimeTypes = crimesWithOffenceCodes
    .groupBy($"DISTRICT", $"CRIME_TYPE")
    .agg(expr("count(INCIDENT_NUMBER) as CRIMES_CNT"))
    .selectExpr("*", "row_number() over(partition by DISTRICT order by CRIMES_CNT desc) as rn")
    .filter($"rn" <= 3)
    .drop($"rn")
    .drop($"CRIMES_CNT")
    .groupBy($"DISTRICT")
    .agg(concat_ws(", ", collect_list($"CRIME_TYPE")).alias("frequent_crime_types"))


  val finalResult =
    crimesDistrictAnalytics
      .join(crimesDistrictMedian, "DISTRICT")
      .join(crimesByDistrictByCrimeTypes, "DISTRICT")
      .select($"DISTRICT", $"crimes_total", $"crimes_monthly", $"frequent_crime_types", $"lat", $"lng")

  finalResult.repartition(1).write.mode("OVERWRITE").parquet(resultFilePath)


}
```

build.sbt sample

```scala
lazy val root = (project in file(".")).

  settings(
    inThisBuild(List(
      organization := "com.example",
      scalaVersion := "2.11.8"
    )),
    name := "boston_crimes",
    version := "0.0.1",

    sparkVersion := "2.3.0",
    sparkComponents := Seq(),

    javacOptions ++= Seq("-source", "1.8", "-target", "1.8"),
    javaOptions ++= Seq("-Xms512M", "-Xmx2048M", "-XX:MaxPermSize=2048M", "-XX:+CMSClassUnloadingEnabled"),
    scalacOptions ++= Seq("-deprecation", "-unchecked"),
    parallelExecution in Test := false,
    fork := true,

    coverageHighlighting := true,

    libraryDependencies ++= Seq(
      "org.apache.spark" %% "spark-streaming" % "2.3.0" % "provided",
      "org.apache.spark" %% "spark-sql" % "2.3.0" % "provided",

      "org.scalatest" %% "scalatest" % "3.0.1" % "test",
      "org.scalacheck" %% "scalacheck" % "1.13.4" % "test",
      "com.holdenkarau" %% "spark-testing-base" % "2.3.0_0.9.0" % "test"
    ),

    // uses compile classpath for the run task, including "provided" jar (cf http://stackoverflow.com/a/21803413/3827)
    run in Compile := Defaults.runTask(fullClasspath in Compile, mainClass in (Compile, run), runner in (Compile, run)).
evaluated,

    scalacOptions ++= Seq("-deprecation", "-unchecked"),
    pomIncludeRepository := { x => false },

   resolvers ++= Seq(
      "sonatype-releases" at "https://oss.sonatype.org/content/repositories/releases/",
      "Typesafe repository" at "http://repo.typesafe.com/typesafe/releases/",
      "Second Typesafe repo" at "http://repo.typesafe.com/typesafe/maven-releases/",
      Resolver.sonatypeRepo("public")
    ),

    pomIncludeRepository := { x => false },

    // publish settings
    publishTo := {
      val nexus = "https://oss.sonatype.org/"
      if (isSnapshot.value)
        Some("snapshots" at nexus + "content/repositories/snapshots")
      else
        Some("releases"  at nexus + "service/local/staging/deploy/maven2")
    }
  )
```

Entity Crime.scala sample

```scala
package com.example.entity

case class Crime(
                  INCIDENT_NUMBER: Option[String],
                  OFFENSE_CODE: Option[Int],
                  OFFENSE_CODE_GROUP: Option[String],
                  OFFENSE_DESCRIPTION: Option[String],
                  DISTRICT: Option[String],
                  REPORTING_AREA: Option[String],
                  SHOOTING: Option[String],
                  OCCURRED_ON_DATE: Option[String],
                  YEAR: Option[Int],
                  MONTH: Option[Int],
                  DAY_OF_WEEK: Option[String],
                  HOUR: Option[Int],
                  UCR_PART: Option[String],
                  STREET: Option[String],
                  Lat: Option[Double],
                  Long: Option[Double],
                  Location: Option[String]
                )
```

Entity OffenseCode.scala sample

```scala
package com.example.entity

case class OffenseCode(
                      CODE: Option[Int],
                      NAME: Option[String],
                      CRIME_TYPE: Option[String]
                      )
```

Build the uber-jar in the project folder using the command:

```
$ sbt clean assembly
```

Launching calculation

After you get the uber-jar, ensure that you have placed this file to your cluster using the Spark server. In the project directory, launch the calculation with the following command:

```
$ spark3-submit --master local --class com.example.BostonCrimesMap path-to/boston_crimes.jar path-to/crime.csv path-to/offense_codes.csv path-to/result
```

View results

All results are written to the directory that you have specified in the `spark3-submit` command. It is represented in a parquet format. Use regular commands to see the result.

Results sample

```
{"DISTRICT":"C6","crimes_total":23460,"crimes_monthly":1870,"frequent_crime_types":"SICK/INJURED/MEDICAL, DRUGS, INVESTIGATE PERSON","lat":42.21212258445548,"lng":-70.85561011772236}
{"DISTRICT":"B2","crimes_total":49945,"crimes_monthly":3986,"frequent_crime_types":"VERBAL DISPUTE, SICK/INJURED/MEDICAL, INVESTIGATE PERSON","lat":42.31600367732765,"lng":-71.0756993065433}
{"DISTRICT":"C11","crimes_total":42530,"crimes_monthly":3284,"frequent_crime_types":"SICK/INJURED/MEDICAL, INVESTIGATE PERSON, VERBAL DISPUTE","lat":42.29263740900067,"lng":-71.05125995734362}
{"DISTRICT":"E13","crimes_total":17536,"crimes_monthly":1368.5,"frequent_crime_types":"SICK/INJURED/MEDICAL, INVESTIGATE PERSON, DRUGS","lat":42.309803655709956,"lng":-71.09800478878388}
{"DISTRICT":"B3","crimes_total":35442,"crimes_monthly":2764,"frequent_crime_types":"VERBAL DISPUTE, INVESTIGATE PERSON, MISSING PERSON","lat":42.283059445201054,"lng":-71.07894914185492}
{"DISTRICT":"E5","crimes_total":13239,"crimes_monthly":1043.5,"frequent_crime_types":"SICK/INJURED/MEDICAL, INVESTIGATE PERSON, PROPERTY","lat":42.19796999447013,"lng":-71.00440862434752}
{"DISTRICT":"A15","crimes_total":6505,"crimes_monthly":499,"frequent_crime_types":"INVESTIGATE PERSON, VANDALISM, SICK/INJURED/MEDICAL","lat":42.17915525091085,"lng":-70.74472508958515}
{"DISTRICT":"A7","crimes_total":13544,"crimes_monthly":1092,"frequent_crime_types":"SICK/INJURED/MEDICAL, INVESTIGATE PERSON, VANDALISM","lat":42.36070260499386,"lng":-71.00394833039842}
{"DISTRICT":"D14","crimes_total":20127,"crimes_monthly":1607.5,"frequent_crime_types":"TOWED MOTOR VEHICLE, SICK/INJURED/MEDICAL, INVESTIGATE PERSON","lat":42.343507245109336,"lng":-71.13125461726499}
{"DISTRICT":"D4","crimes_total":41915,"crimes_monthly":3297.5,"frequent_crime_types":"PROPERTY, INVESTIGATE PERSON, SICK/INJURED/MEDICAL","lat":42.34124251790852,"lng":-71.07725024946983}
{"DISTRICT":"E18","crimes_total":17348,"crimes_monthly":1313,"frequent_crime_types":"SICK/INJURED/MEDICAL, INVESTIGATE PERSON, VERBAL DISPUTE","lat":42.26268061122604,"lng":-71.11891998757692}
{"DISTRICT":"A1","crimes_total":35717,"crimes_monthly":2775,"frequent_crime_types":"PROPERTY, SICK/INJURED/MEDICAL, WARRANT ARREST","lat":42.33123077259828,"lng":-71.01991881362002}
```

Example of launching jobs with an RDD

Konstantin Alpashkin

Contents

To Table of Contents

- RDD operations
  - Create
  - Modify
  - View contents
- Example

Spark resilient distributed dataset (RDD) is a fault-tolerant collection of elements distributed across the nodes of a cluster that can be operated on in parallel.

This section provides basic RDD usage examples in Scala that can run in spark3-shell. Also, Spark supports other programming languages like Java, Python, and R.

RDD operations

Create

There are two ways to create a new RDD, see the examples below.

Parallelize a collection

To create a new RDD from a collection, you can use `sparkContext.parallelize()` in the driver program code.

```
val dataSeq = Seq(("foo", 0), ("bar", 1), ("buzz", 2))
val rdd=sc.parallelize(dataSeq)


val dataArr = Array(0, 1, 2, 3)
val rdd1=sc.parallelize(dataArr)


val emptyRDD = sc.parallelize(Seq.empty[String])
```

From external storage, like HDFS

You can create an RDD from an external storage like HDFS, HBase, or any other data source implementing Hadoop InputFormat. A way to read data from HDFS is below.

```
val rdd2 = sc.textFile("hdfs://adh/user/admin/file_test.csv")
```

Once an RDD object is created, you can perform main Spark operations on the RDD.

Modify

All RDD operations are divided into two types:

- **Transformations** that return a new modified RDD. For example, `map()`, `flatMap()`, `union()`, etc.
- **Actions** that return a value to the driver program after running a computation on the RDD. For example, `collect()`, `count()`, etc.

All RDD operations, including creation and modification, are lazy, i.e. they do not compute right away. The computation is done only when some result must be returned to the driver program, for example `rdd.take(10)`.

For a complete reference on supported operations, see Spark RDD API.

View contents

To view the contents of an RDD, use `rdd.take(x)` that returns first `x` elements of the RDD.

> **NOTE**
>
> Be careful with using constructs like `rdd.collect().foreach(println)` as this may cause the Spark driver machine to run out of memory, since `collect()` fetches the entire RDD from all the cluster nodes to a single machine.

Example

The snippet below summarizes basic RDD operations. To run the example without errors, create the corresponding test file (*test_lorem.txt*) in HDFS. To create the test file:

1. On your local file system, create the *test_lorem.txt* file with dummy contents:

```
echo "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque
penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis,
sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus
ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus
elementum semper nisi. Aenean vulputate eleifend tellus." > test_lorem.txt
```

2. Create a test directory in HDFS, provide access permissions, and copy the test file to HDFS:

```
$ sudo -u hdfs hdfs dfs -mkdir /user/admin
$ sudo -u hdfs hdfs dfs -chown admin:admin /user/admin
$ sudo -u hdfs hdfs dfs -put /home/admin/spark-demo/test_lorem.txt /user/admin/test_lorem.txt
```

Please ensure that the `hdfs` user has access to the *test_lorem.txt* file.

Once the test data is loaded to HDFS, you can run the following Scala code in spark3-shell (*/bin/spark-shell*).

```
val rdd = sc.textFile("hdfs://adh/user/admin/test_lorem.txt")
println("Created RDD:")
rdd.take(10)

val resultRDD = rdd.flatMap(line => line.split(" ")) ①
      .map(word => (word,1)) ②
      .reduceByKey(_+_) ③
      .sortBy(_._2, false) ④

resultRDD.take(100).foreach(println)

resultRDD.persist() ⑤
println("Writing RDD to a file: /user/admin/lorem_demo_result")
resultRDD.saveAsTextFile("hdfs://adh/user/admin/lorem_demo_result") ⑥
```

① For each line of the original text file, split the string into words.

② For each word, create a tuple `(<word>,1)`.

③ Sum up repetitive words and leave only unique words with number of occurrences.

④ Sort words by occurrence count.

⑤ Cache RDD in RAM. This synchronizes the RDD contents among all cluster nodes.

⑥ Store RDD contents to HDFS. This creates an HDFS directory with parts of RDD data.

Running the example above outputs the following:

```
Created RDD:
val res0: Array[String] = Array(Lorem ipsum dolor sit amet, ...)

val resultRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[9] at sortBy at <console>:4

(Aenean,3)
(vulputate,2)
(pede,2)
(nec,,2)
(Donec,2)
(justo,,2)
(montes,,1)
...

Writing RDD to a file: /user/admin/lorem_demo_result
```

After running the example, check the RDD writes to HDFS:

```
$ sudo -u hdfs hdfs dfs -ls /user/admin/lorem_demo_result
```

This should return a similar output:

```
Found 3 items
-rw-r--r--   3 admin admin          0 2023-07-25 20:55 /user/admin/lorem_demo_result/_SUCCESS
-rw-r--r--   3 admin admin         64 2023-07-25 20:55 /user/admin/lorem_demo_result/part-00000
-rw-r--r--   3 admin admin        753 2023-07-25 20:55 /user/admin/lorem_demo_result/part-00001
```

Example of launching jobs with a DataFrame

Konstantin Alpashkin

Contents

To Table of Contents

- DataFrame operations
  - Create
  - Modify
  - View contents
  - Run SQL queries
- Example

Spark DataFrame (DF) is a data structure organized into named columns and distributed across a cluster. It is conceptually similar to a table in a relational database, but with smart optimizations under the hood. In terms of Spark, a DataFrame is technically a DataSet of Rows.

This section provides basic DataFrame usage examples that can run in spark3-shell, however Spark supports other languages like Java, Python, and R.

DataFrame operations

Create

A DataFrame object can be constructed from a variety of sources, most common ways are presented below.

From structured data files

The following snippet shows how to create a DF from a JSON file stored on HDFS.

```
val df = spark
    .read
    .json("hdfs://adh/user/admin/testload/people.json")

// read multiline JSON
val multiline_df = spark
    .read
    .option("multiline","true")
    .json("hdfs://adh/user/admin/testload/people_multiline.json")
```

In this case the DF schema is inferred automatically based on the structure of the JSON file. For more information on creating DFs from other sources, like Avro, Parquet, ORC, and many others, see available data sources.

From RDD

You can convert an RDD to DF using the `rdd.toDF()` method. By default, Spark generates the schema automatically based on the RDD structure, though you can also specify the schema explicitly.

```
val rdd = sc.textFile("hdfs://adh/user/admin/testload/test_data_rdd.txt")
val df = rdd.toDF()
```

From external database via JDBC

The example below shows getting a DF from a MySQL table. The MySQL driver JAR must be available in the classpath for this to work.

```
val df_mysql = spark.read.format("jdbc")
   .option("url", "jdbc:mysql://<host>:<port>/db_name")
   .option("driver", "com.mysql.jdbc.Driver")
   .option("dbtable", "<table_name>")
   .option("user", "<username>")
   .option("password", "<password>")
   .load()
```

Getting a DF from other DB engines is conceptually similar.

Apart from these, there are many other ways to get a DataFrame object, like using `SparkSession.sql()` / `SparkSession.createDataFrame()`, querying a Hive/HBase table, and so on. More details on using these methods are available in Spark documentation.

Modify

All DataFrame operations are divided into two types:

- **Transformations** that return a new modified DF. For example, `map()`, `filter()`, `agg()`, `col()`, `join()`, etc.
- **Actions** that return a value to the driver program. For example, `collect()`, `count()`, etc.

All DF operations are lazy, i.e. they do not compute right away. The computation is done only when some result must be returned to the driver program, for example `df.show()`.

For a complete reference on supported operations, see Spark DataSet API.

View contents

To view a DF contents, use `df.show(x)` that returns first `x` rows of the DF.

> **NOTE**
>
> Be careful with using constructs like `df.collect().foreach(println)` as this may cause the Spark driver machine to run out of memory, since `collect()` fetches the entire DF from all the cluster nodes to a single machine.

Use `df.printSchema()` to print the DF schema as shown below.

```
root
 |-- age: long (nullable = true)
 |-- dept: string (nullable = true)
 |-- id: long (nullable = true)
 |-- lastname: string (nullable = true)
 |-- name: string (nullable = true)
```

Run SQL queries

The Spark SQL module allows you to run traditional SQL queries on Dataframes. For this, you should create a temporary view from a DataFrame using `createOrReplaceTempView()`. Then, you can submit SQL queries to this view using the `SparkSession.sql()` method. An example is below.

```
val df = spark
    .read
    .csv("hdfs://adh/user/admin/testload/employees.json")

df.createOrReplaceTempView("employees")

val df_sql = spark.sql("SELECT name, lastname, email FROM employees")
df_sql.show()
```

Once created, the temporary view behaves like an ordinary SQL table, but it is not persisted anywhere as a file. The view is attached to the SparkSession where it was created, and remains accessible until the session dies. For more details on running SQL queries, see SparkSession.sql() API.

Example

The snippet below summarizes basic DF operations. To run the example without errors, create the corresponding test file (*employees.json*) in HDFS. To create the test file:

1. On your local file system, create the *employees.json* file with dummy contents:

```
echo -e
'{"id":1,"name":"Sarah","lastname":"Connor","dept":"hr","age":34}\n{"id":2,"name":"Michael","lastname":"Scott","dept":"sales","age
":47}\n{"id":3,"name":"Luke","lastname":"Skywalker","dept":"it","age":45}\n{"id":4,"name":"James","lastname":"Hetfield","dept":"sa
les","age":50}' > employees.json
```

2. Create a test directory in HDFS, provide access permissions, and copy the test file to HDFS:

```
$ sudo -u hdfs hdfs dfs -mkdir /user/admin
$ sudo -u hdfs hdfs dfs -chown admin:admin /user/admin
$ sudo -u hdfs hdfs dfs -put /home/admin/spark-demo/employees.json /user/admin/employees.json
```

   Please ensure that the `hdfs` user has access to the *employees.json* file.

Once the test data is loaded to HDFS, you can run the following Scala code in spark3-shell (*/bin/spark-shell*).

```
val df_employees = spark.read.json("hdfs://adh/user/admin/employees.json") ①

println("DataFrame created:")
df_employees.show()

println("DataFrame schema:")
df_employees.printSchema()

println("Show name-age selection from all employees:")
df_employees.select("name","age").show()

println("Get all employees older than 40")
df_employees.filter("age > 40").show()


val depts_data = Seq((1,"it"),(2,"sales"),(3,"hr"))
val rdd = sc.parallelize(depts_data)
val df_department = rdd.toDF("id","department") ②

println("DataFrame created:")
df_department.show()

val joined = df_employees.filter("age > 30").join(df_department, df_employees("dept") === df_department("department")) ③

println("Get number of employees per department")
joined.groupBy("department").count().show()

println("Get average age per department")
joined.groupBy("department").mean("age").show() ④
```

① Create a DF from a JSON file in HDFS.

② Create a DF from RDD.

③ Filter DF data by a column and apply `JOIN` for two DFs.

④ Use `GROUP BY` with aggregation by a column.

Running the example outputs the following:

```
DataFrame created:
+---+-----+---+---------+-------+
|age| dept| id| lastname|   name|
+---+-----+---+---------+-------+
| 34|   hr|  1|   Connor|  Sarah|
| 47|sales|  2|    Scott|Michael|
| 45|   it|  3|Skywalker|   Luke|
| 50|sales|  4| Hetfield|  James|
+---+-----+---+---------+-------+

DataFrame schema:
root
 |-- age: long (nullable = true)
 |-- dept: string (nullable = true)
 |-- id: long (nullable = true)
 |-- lastname: string (nullable = true)
 |-- name: string (nullable = true)

Show name-age selection from all employees:
+-------+---+
|   name|age|
+-------+---+
|  Sarah| 34|
|Michael| 47|
|   Luke| 45|
|  James| 50|
+-------+---+

Get all employees older than 40:
+---+-----+---+---------+-------+
|age| dept| id| lastname|   name|
+---+-----+---+---------+-------+
| 47|sales|  2|    Scott|Michael|
| 45|   it|  3|Skywalker|   Luke|
| 50|sales|  4| Hetfield|  James|
+---+-----+---+---------+-------+

DataFrame created:
+---+----------+
| id|department|
+---+----------+
|  1|        it|
|  2|     sales|
|  3|        hr|
+---+----------+

Get number of employees per department:
+----------+-----+
|department|count|
+----------+-----+
|     sales|    2|
|        hr|    1|
|        it|    1|
+----------+-----+

Get average age per department:
+----------+--------+
|department|avg(age)|
+----------+--------+
|     sales|    48.5|
|        hr|    34.0|
|        it|    45.0|
+----------+--------+
```

Spark and Hive

Konstantin Alpashkin

Contents

To Table of Contents

Custom configuration

By default, Spark is already configured to work with Hive. Hive settings for Spark are located in the home directory — */etc/spark3/conf*. If your Spark application is interacting with Hadoop or both with Hive, you need to put Hadoop configuration files in the Spark's classpath.

Multiple running applications might require different Hadoop/Hive client side configurations. You can copy and modify *hdfs-site.xml*, *core-site.xml*, *yarn-site.xml* in Spark's classpath for each application.

NOTE

In a Spark cluster running on YARN, these configuration files are set cluster-wide and cannot safely be changed by the application.

The best choice is to use Spark Hadoop properties in the form of `spark.hadoop.*`, and Spark Hive properties in the form of `spark.hive.*`. For example, adding `spark.hadoop.abc.def=xyz` is the same as adding the `abc.def=xyz` Hadoop property; adding `spark.hive.abc=xyz` is equivalent to the `hive.abc=xyz` Hive property. They can be considered as same as normal Spark properties which can be set in *$SPARK_HOME/conf/spark-defaults.conf*. Default home directory for Spark is */etc/spark3/conf*. Here you can keep all Spark configurations.

In some cases, you may want to avoid hard-coding certain configurations in a SparkConf. For instance, Spark allows you to simply modify or add configurations at runtime:

```
./bin/spark3-submit \
  --name "My app" \
  --master local[4] \
  --conf spark.eventLog.enabled=false \
  --conf "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" \
  --conf spark.hadoop.abc.def=xyz \
  --conf spark.hive.abc=xyz
  myApp.jar
```

You can find all Spark parameters description in the Spark documentation.

Custom configuration

If you need to make some custom updates in Spark applications for Hive, there are two ways:

- One way is by adding custom properties into the *spark-defaults.conf* file and adding this file to the Hive classpath.
- The other way is to set configuration properties in the *hive-site.xml* Hive configuration file. All configuration files are stored in the Spark home directory */etc/spark3/conf/hive-site.xml*.

For more information about Hive parameters for Spark, please, refer to Hive on Spark parameters.

ADB Spark 3 Connector

Daria Barysheva

To Table of Contents

Arenadata DB (ADB) is a distributed analytical data storage built on the open-source massively parallel Greenplum DBMS.

Apache Spark 3 is an open-source framework that is designed for distributed processing of unstructured or weakly structured data.

ADB Spark 3 Connector provides the possibility of high-speed, parallel data exchange between Spark 3 and Arenadata DB. The following articles contain a full description of ADB Spark 3 Connector:

- Overview
- Prerequisites
- Usage examples
- Options

IMPORTANT

Starting with ADH 3.1.2.1.b1, the connector for Spark 3 comes with the ADH bundle. You should install the Spark 3 service to use the connector. For information on the Spark 2 connector, see ADB Spark Connector.

ADB Spark 3 Connector overview

Mikhail Serov

Contents

To Table of Contents

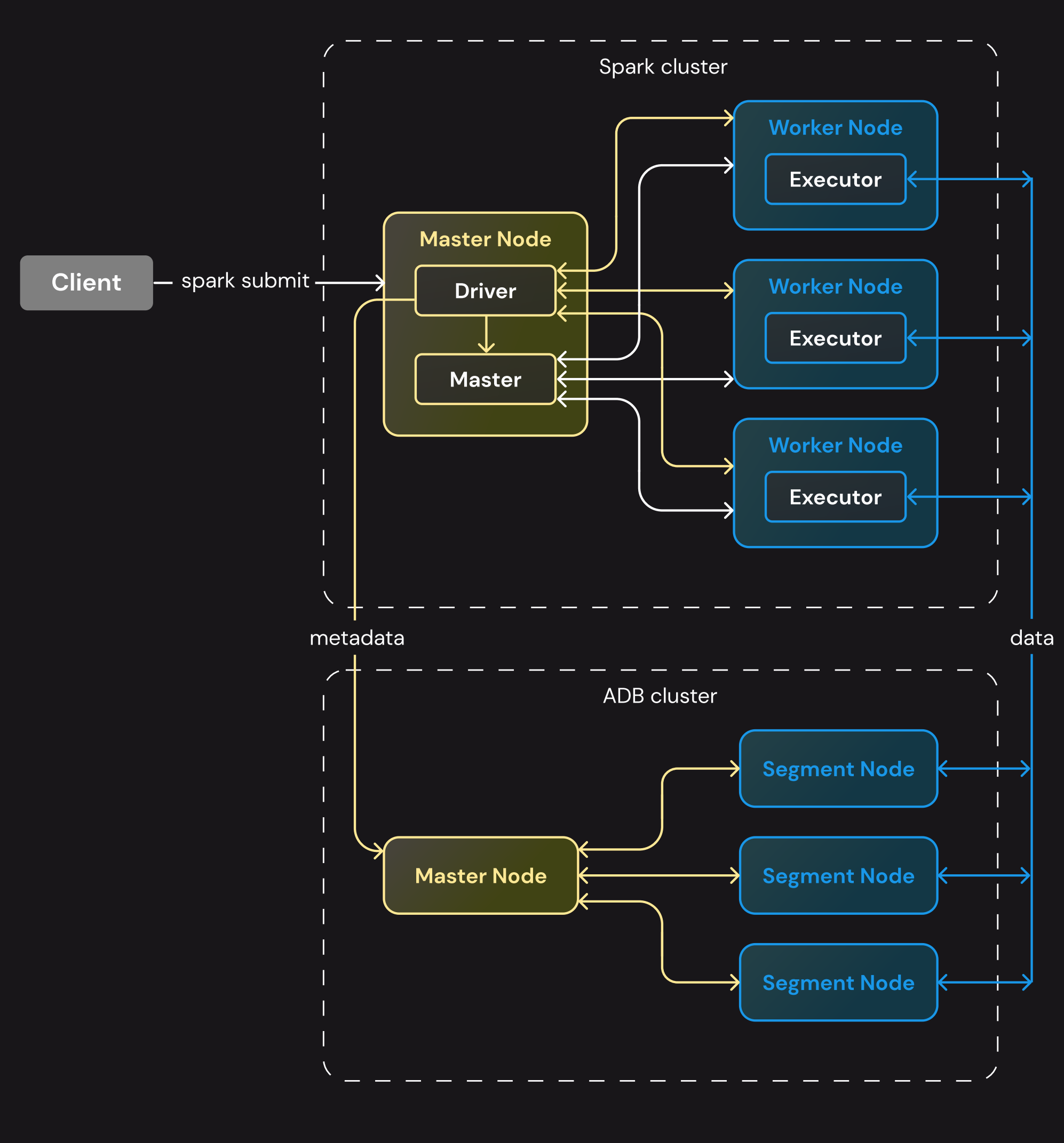
Architecture
Work algorithms
- Data reading
- Data writing

Supported data types
Supported JDBC drivers
Connector usage

ADB Spark 3 Connector provides the ability of high speed parallel data exchange between Apache Spark 3 and Arenadata DB.

Architecture

Each Spark 3 application consists of a controlling process (driver) and a number of distributed worker processes (executors). The interaction scheme is provided below.



ADB Spark 3 Connector architecture

Work algorithms

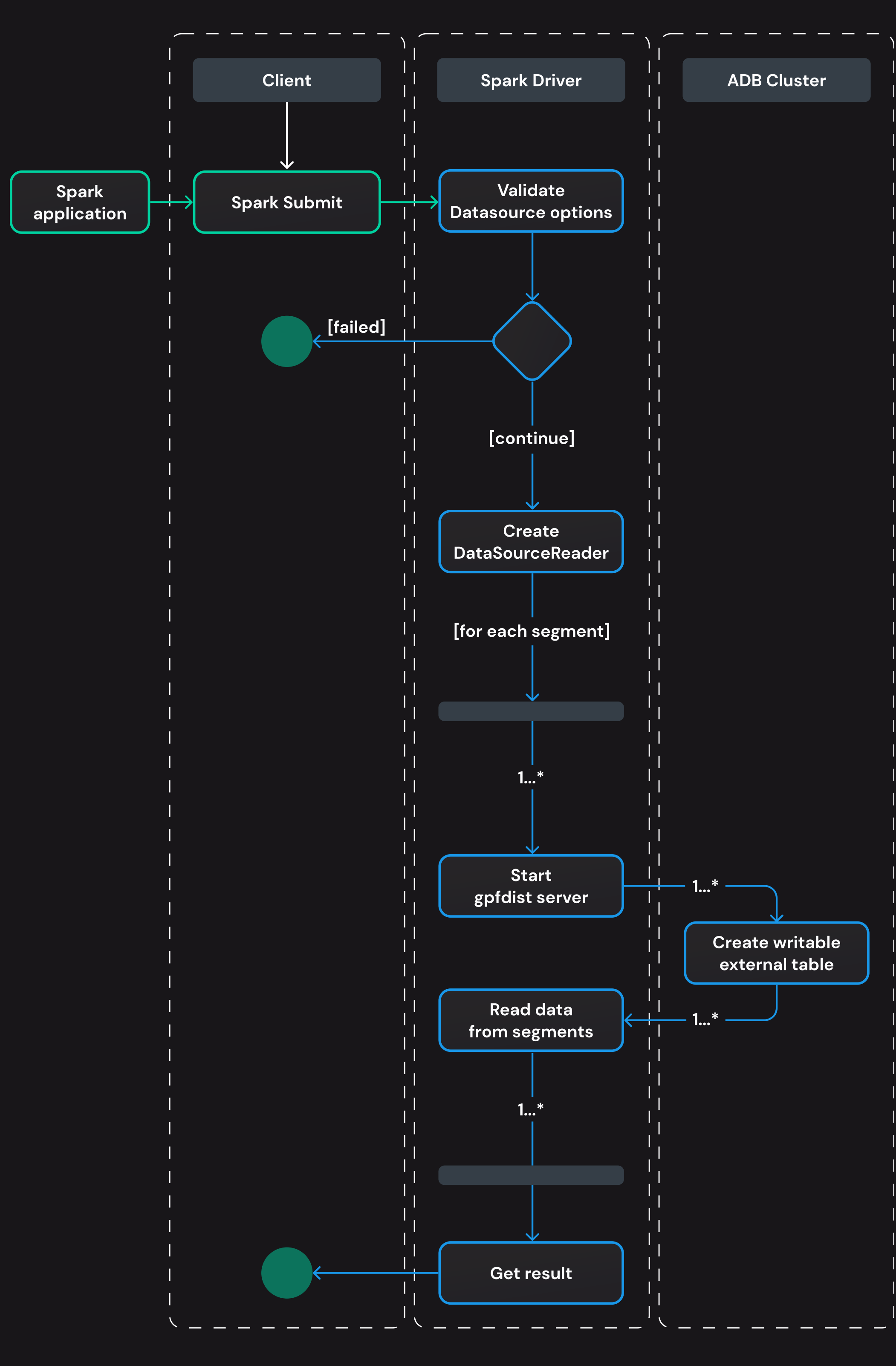
Data reading

ADB Spark 3 Connector initializes a driver in order to load an ADB table into Spark 3. That driver establishes a connection to the ADB master via JDBC to obtain the necessary metadata about the table. The metadata contains information on type, tables structure, and distribution key. The distribution key allows you to distribute the data from a table to the existing Spark 3 performers.

The data is stored at ADB segments. A Spark 3 application that uses a connector can transfer data from each segment into one or more Spark 3 partitions. There are five partition modes:

- **According to the gp_segment_id**. The data from ADB tables is read and distributed across Spark partitions depending on a segment. The number of Spark 3 partitions corresponds to the number of active ADB segments. This partition mode does not require additional parameters and is used by default.
- **According to the specified column and the number of partitions**. Data is divided into Spark 3 partitions inside the column range by the specified partitions number. The number of Spark 3 partitions corresponds to the number of ranges. It is necessary to set the appropriate parameters (see ADB Spark 3 Connector options). This mode has limitations: only the `integer` and the `date/time` table fields can be used as a column.
- **According to the specified column**. Data is divided into Spark 3 partitions depending on the unique values of the specified column. The number of Spark 3 partitions corresponds to the number of unique values of the specified column. It is necessary to set the appropriate parameters (see ADB Spark 3 Connector options). This mode is recommended for the case of a small and limited set of values.
- **According to the specified number of partitions**. Data is divided into Spark 3 partitions depending on the specified number of partitions. It is necessary to set the appropriate parameters (see ADB Spark 3 Connector options).
- **According to the specified hash function and the number of partitions**. Data is divided into Spark 3 partitions depending on the specified hash function and the specified number of partitions. It is necessary to set the appropriate parameters (see ADB Spark 3 Connector options). This mode has limitations: only the expression that returns nothing but an integer can be used as a hash function.

Some data processing tasks are spread across all Spark 3 executors along with the corresponding partition that was created before. The data exchange is performed in parallel via external writable tables for every partition.



Data reading algorithm

Data writing

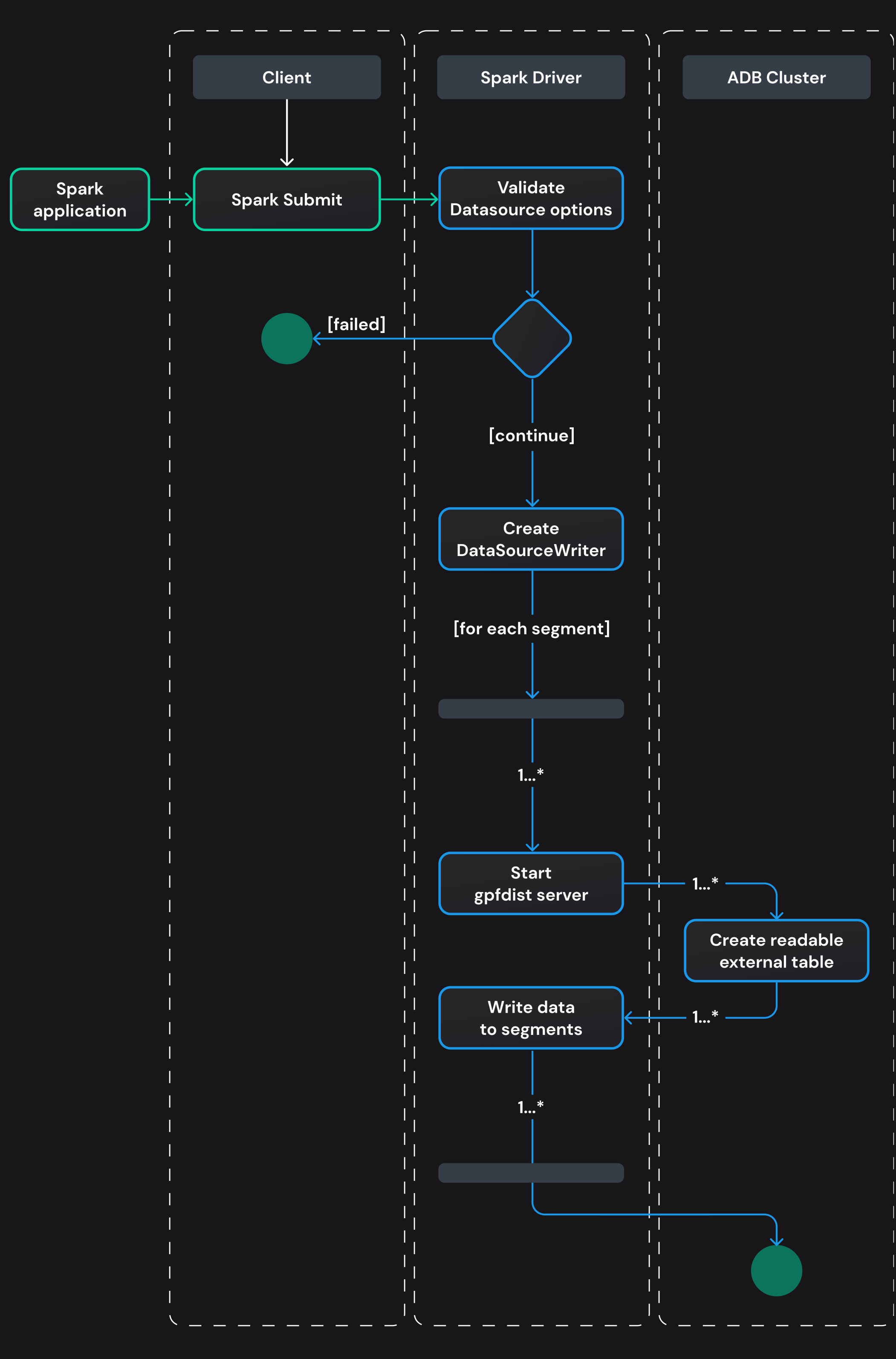
ADB Spark 3 Connector initializes a driver to load the table from Spark 3 to ADB. After that, the driver establishes a connection to the ADB master via JDBC and prepares the load of ADB. The load method depends on the writing mode.

The following writing modes are supported:

- `overwrite`. Depending on `spark.db.table.truncate`, the target table is either completely deleted, or all data is cleaned up before the writing process.
- `append`. The additional data is written to the target table.
- `errorIfExists`. If the target table exists, the error appears.

A number of the `spark.db.create.table` options are also available. This allows you to specify additional settings when you create a target table (see ADB Spark 3 Connector options).

A task to load the data is created for each Spark 3 partition. The data exchange is performed in parallel via external readable tables for every segment.



Data writing algorithm

Supported data types

Matching of data types during a transfer is shown in the tables below.

Transfer from ADB to Spark 3

ADB	Spark 3
bigint	LongType
bigSerial	LongType
bit	StringType
bytea	BinaryType
boolean	BooleanType
char	StringType
date	DateType
decimal	DecimalType
float4	FloatType
float8	DoubleType
int	IntegerType
interval	CalendarIntervalType
serial	IntegerType
smallInt	ShortType
text	StringType
time	TimeStampType
timestamp	TimeStampType
timestamptz	TimeStampType
timetz	TimeStampType
varchar	StringType

Transfer from Spark 3 to ADB

Spark 3	ADB
BinaryType	byte
BooleanType	boolean
CalendarIntervalType	interval
DateType	date
DecimalType	numeric
DoubleType	float8
FloatType	float4
IntegerType	int
LongType	bigint
ShortType	smallint
StringType	text
TimeStampType	timestamp

Supported JDBC drivers

ADB Spark 3 Connector communicates with ADB via a JDBC connection. The PostgreSQL JDBC driver comes with the connector.

A JDBC connection string for the standard driver has the following structure:

```
jdbc:postgresql://<master_ip>[:<port>]/<database_name>
```

where:

- `<master_ip>` — an IP address of the ADB master host.
- `<port>` — a connection port (`5432` by default).
- `<database_name>` — a database name in ADB.

Additionally, ADB Spark 3 Connector supports the use of third-party JDBC drivers. In order to use a third-party JDBC driver, do the following:

1. Prepare an ODBC connection string for the ADB master.
2. Submit a JAR file with a third-party JDBC driver via one of the following ways:
    - Use the `--jars` environment variable when calling spark3-shell or spark3-submit.
    - Build the *uber-jar* file with all dependencies.
    - Install a JAR file with a third-party driver on all Spark 3 nodes.
3. Set the full path to the driver Java class in the connector option `spark.adb.driver`.

Connector usage

To transfer data between Spark 3 and ADB via ADB Spark 3 Connector, you can use the standard utilities spark3-shell and spark3-submit. Examples of the spark3-shell usage can be found in the ADB Spark 3 Connector usage examples article.

When developing a standalone Spark application, it is necessary to combine the connector JAR file (*/usr/lib/spark3/jars/adb-spark-connector-assembly-release-<connector_version>-spark-<spark_version>.jar*) with other application dependencies and put the result into the *uber-jar* file. For more information, see Submitting Applications in the Spark documentation.

ADB Spark 3 Connector prerequisites

Mikhail Serov

Contents

To Table of Contents

Checklist
Memory
Network

Checklist

Before installing and using ADB Spark 3 Connector, you have to make sure that the following requirements are met:

- There is access to the ADB cluster.
- There is access to the Spark 3 cluster.
- There is network connection between the ADB master node and the Spark 3 driver.
- There is network connection between the ADB master node and each Spark 3 executor node.
- There is network connection between each ADB segment node and each Spark 3 executor node.

You need to grant special rights to the user who will access ADB tables from Spark 3. Those rights include the right to create external tables. You can use the following sample:

```
ALTER ROLE <role_name>
WITH    CREATEEXTTABLE(protocol='gpfdist',type='readable')
        CREATEEXTTABLE(protocol='gpfdist',type='writable');
```

where `<role_name>` is a user name in ADB.

You also need to configure the selected user’s access to ADB tables from Spark 3 hosts in the *pg_hba.conf* file (see ADB Spark 3 Connector usage examples).

Memory

In general, Spark 3 runs fine using any memory amount between 8 GB and hundreds of gigabytes per machine. We recommend to allocate 75% of the memory for Spark 3 at most — leave the rest for the operating system and buffer cache.

The memory amount that you need depends on your application. To determine the memory amount that your application uses for a certain dataset size, load a part of your dataset into the Spark 3 RDD, then use the ***Storage*** tab of the Spark 3 monitoring UI (*http://<driver-node>:4040*) to see the memory size for that part. Memory usage is affected by storage level and serialization format. See the tuning guide for tips on how to reduce the memory usage.

NOTE

Java VM does not always behave well if there is more than 200 GB of RAM. If you purchase machines with more RAM, you can run multiple worker Java VMs per node. In Spark 3 standalone mode, you can set the number of worker machines per node via the `SPARK_WORKER_INSTANCES` variable in the *conf/spark-env.sh* script. You can also set the number of cores per worker machine via the `SPARK_WORKER_CORES` variable.

Network

ADB Spark 3 Connector uses the `gpfdist` protocol to transfer data between ADB segments and Spark 3 execution nodes. By default, the connector starts an instance of the `gpfdist` server inside the `[8080 .. 65536]` port range.

IMPORTANT

Make sure there are open ports in the range specified above. The ports should be accessible from the ADB master.

ADB Spark 3 Connector usage examples

Daria Barysheva

Contents

To Table of Contents

- Requirements
- Usage examples: Scala
  - Load data from ADB to Spark
  - Run any SQL query in ADB
  - Write data from Spark to ADB
- Usage examples: PySpark
  - Load data from ADB to Spark
  - Write data from Spark to ADB

Requirements

This article describes how to transfer data between ADB and Spark 3 via ADB Spark 3 Connector. The following prerequisites are met:

- The ADB cluster is installed according to the Online installation guide.
- The ADH cluster is installed according to the Online installation guide. The minimal ADH version is 3.1.2.1.b1.
- The Spark 3 service is added to the ADH cluster.
- IP address of the ADB master host is `10.92.40.38`. The default port number for income PostgreSQL connections is `5432`.
- The `adb` database exists in the ADB cluster.
- The `adb_to_spark` user with the `CREATEEXTTABLE` privileges and `123` password exists in the ADB cluster. To add a user, you can run the following query:

```
CREATE ROLE adb_to_spark
WITH  CREATEEXTTABLE(protocol='gpfdist',type='readable')
      CREATEEXTTABLE(protocol='gpfdist',type='writable')
      LOGIN
      PASSWORD '123';
```

- The `adb_to_spark` user can access the `adb` database from the ADH cluster. For this purpose, you can change the *pg_hba.conf* file as follows:

```
host    all  adb_to_spark       0.0.0.0/0     md5
```

Note that it is not a production-ready configuration. Use it only for test needs. Instead `0.0.0.0/0`, you can write IP addresses of the hosts in the ADH cluster where Spark 3 is installed (with a subnet number).

> TIP
>
> You can modify the *pg_hba.conf* file via the ADCM web interface. To do this, fill in the ***Custom pg_hba section*** parameter on the ***Configuration*** tab of the ADB service in the ADB cluster. To apply changes, click ***Save*** and run the service ***Reconfigure*** action.

- The `author` table exists in the `adb` database. To create and fill in a table, use the following queries:

```
CREATE TABLE author(id INT NOT NULL, name TEXT NOT NULL)
WITH (appendoptimized=true)
DISTRIBUTED BY(id);
```

```
INSERT INTO author(id, name) VALUES
(1,'Virginia Woolf'),
(2,'J.R.R. Tolkien'),
(3,'Harper Lee'),
(4,'J.D. Salinger'),
(5,'George Orwell'),
(6,'John Steinbeck'),
(7,'Margaret Mitchell'),
(8,'Alan Moore'),
(9,'Jack Kerouac'),
(10,'Ernest Hemingway');
```

- The `adb_to_spark` user has necessary permissions on the `author` table:

```
GRANT SELECT, INSERT ON public.author TO adb_to_spark;
```

ADB Spark 3 Connector communicates with ADB via a JDBC connection. The PostgreSQL JDBC driver comes with the connector.

Usage examples: Scala

The examples of Scala code presented below are intended for execution in the interactive shell for Scala — spark3-shell.

Load data from ADB to Spark

1. Open spark3-shell on the host where Spark 3 is installed:

```
$ sudo -u hdfs spark3-shell
```

2. Specify the connector settings for reading data from the ADB `author` table:

```
val options = Map(
   "spark.adb.url" -> "jdbc:postgresql://10.92.40.38:5432/adb",
   "spark.adb.user" -> "adb_to_spark",
   "spark.adb.password" -> "123",
   "spark.adb.dbschema" -> "public",
   "spark.adb.dbtable" -> "author"
   )
```

The result:

```
val options: scala.collection.immutable.Map[String,String] = HashMap(spark.adb.password -> 123, spark.adb.dbtable -> author,
spark.adb.user -> adb_to_spark, spark.adb.url -> jdbc:postgresql://10.92.40.38:5432/adb, spark.adb.dbschema -> public)
```

3. Register a DataFrame:

```
val adb_author  = spark.read.format("adb").options(options).load()
```

The result:

```
val adb_author: org.apache.spark.sql.DataFrame = [id: int, name: string]
```

4. Check the DataFrame content:

```
adb_author.show()
```

The result:

```
+---+-----------------+
| id|             name|
+---+-----------------+
|  4|    J.D. Salinger|
|  8|       Alan Moore|
|  5|    George Orwell|
| 10| Ernest Hemingway|
|  2|   J.R.R. Tolkien|
|  1|   Virginia Woolf|
|  7|Margaret Mitchell|
|  6|   John Steinbeck|
|  3|       Harper Lee|
|  9|     Jack Kerouac|
+---+-----------------+
```

5. Get the table schema:

```
adb_author.printSchema()
```

The result:

```
root
 |-- id: integer (nullable = false)
 |-- name: string (nullable = false)
```

6. Calculate a number of the table rows:

```
adb_author.count()
```

The result:

```
val res9: Long = 10
```

Run any SQL query in ADB

You can execute any SQL query through the connector as well. To do this, run the following commands:

1. Set additional Spark session settings and enable auxiliary functions:

```
spark.stop
val spark1 = org.apache.spark.sql.SparkSession.
   builder().
   master("local[*]").
   appName("spark_example").
   config("spark.adb.url","jdbc:postgresql://10.92.40.38:5432/adb").
   config("spark.adb.driver","org.postgresql.Driver").
   config("spark.adb.user","adb_to_spark").
   config("spark.adb.password","123").
   getOrCreate()
import io.arenadata.spark.adb.implicits._
```

The result:

```
val spark1: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@20e87aa3
```

2. Create an arbitrary table (`test_spark`) in ADB and fill it with some test data:

```
spark1.executeAdbQueryOnMaster("CREATE TABLE public.test_spark(id INT);")
spark1.executeAdbQueryOnMaster("INSERT INTO public.test_spark VALUES(1);")
spark1.executeAdbQueryOnMaster("INSERT INTO public.test_spark VALUES(2);")
spark1.executeAdbQueryOnMaster("INSERT INTO public.test_spark VALUES(3);")
```

3. Create a DataFrame based on the new ADB table (`test_spark`):

```
val res = spark1.executeAdbSelectQueryOnMaster("SELECT * FROM public.test_spark;")
```

The result:

```
val res: org.apache.spark.sql.DataFrame = [id: int]
```

4. Check the DataFrame content:

```
res.show()
```

The result:

```
+---+
| id|
+---+
|  2|
|  3|
|  1|
+---+
```

Write data from Spark to ADB

1. Specify the connector settings for writing data to ADB:

```
val options_write = Map(
   "spark.adb.url" -> "jdbc:postgresql://10.92.40.38:5432/adb",
   "spark.adb.user" -> "adb_to_spark",
   "spark.adb.password" -> "123",
   "spark.adb.dbschema" -> "public",
   "spark.adb.dbtable" -> "test_spark_write")
```

The result:

```
val options_write: scala.collection.immutable.Map[String,String] = HashMap(spark.adb.password -> 123, spark.adb.dbtable ->
test_spark_write, spark.adb.user -> adb_to_spark, spark.adb.url -> jdbc:postgresql://10.92.40.38:5432/adb, spark.adb.dbschema ->
public)
```

2. Run the following command to create a `test_spark_write` table in ADB and write data from the `res` DataFrame created in the previous example into it:

```
res.write.format("adb").options(options_write).mode(org.apache.spark.sql.SaveMode.Overwrite).save()
```

3. Check the data availability in ADB (for example, via psql):

   - Table structure:

```
\d+ public.test_spark_write
```

```
                   Table "public.test_spark_write"
 Column |  Type   | Modifiers | Storage | Stats target | Description
--------+---------+-----------+---------+--------------+-------------
 id     | integer |           | plain   |              |
Distributed by: (id)
```

   - Table data:

```
SELECT * FROM public.test_spark_write;
```

```
 id
----
  2
  1
  3
(3 rows)
```

Usage examples: PySpark

The examples of Python code presented below are intended for execution in the PySpark shell, which is available on the hosts where Spark3 is installed. The shell process must be started under the `hdfs`/`spark` user to have sufficient permissions for interacting with a remote ADB cluster. The following commands show how to start the PySpark shell appropriately.

1. Create a new virtual environment using Python from */opt/pyspark3-python/*:

```
$ cd ~
$ mkdir pyspark-demo
$ /opt/pyspark3-python/bin/python3 -m venv pyspark-demo/venv
```

2. Activate the virtual environment:

```
$ source pyspark-demo/venv/bin/activate
```

3. In the activated virtual environment, run the PySpark shell:

```
$ sudo -u hdfs pyspark3
```

You should see a similar output:

```
(venv) [admin@ka-adh-1 ~]$ sudo -u hdfs pyspark3
Python 3.10.4 (main, Jun  1 2024, 01:11:04) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Picked up _JAVA_OPTIONS: -Djava.io.tmpdir=/var/tmp
Picked up _JAVA_OPTIONS: -Djava.io.tmpdir=/var/tmp
...
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.4.3
      /_/

Using Python version 3.10.4 (main, Jun  1 2024 01:11:04)
Spark context Web UI available at http://ka-adh-1.ru-central1.internal:4142
Spark context available as 'sc' (master = yarn, app id = application_1724163954275_0023).
SparkSession available as 'spark'.
```

Load data from ADB to Spark

1. In the PySpark shell, define a dict with connector settings for reading data from the ADB `author` table:

```
opts_read = {
   "spark.adb.url": "jdbc:postgresql://10.92.40.38/adb",
   "spark.adb.user": "adb_to_spark",
   "spark.adb.password": "123",
   "spark.adb.dbschema": "public",
   "spark.adb.dbtable": "author"
}
```

2. Create a DataFrame:

```
adb_author = spark.read.format("adb") \
            .options(**opts_read) \
            .load()
```

The result:

```
24/08/28 19:58:59 INFO HikariDataSource: HikariPool-1 - Starting...
24/08/28 19:58:59 INFO HikariDataSource: HikariPool-1 - Start completed.
```

3. Check the DataFrame content:

```
adb_author.show()
```

The result:

```
+---+-----------------+
| id|             name|
+---+-----------------+
|  4|    J.D. Salinger|
|  8|       Alan Moore|
|  5|    George Orwell|
| 10| Ernest Hemingway|
|  2|   J.R.R. Tolkien|
|  1|   Virginia Woolf|
|  7|Margaret Mitchell|
|  6|   John Steinbeck|
|  3|       Harper Lee|
|  9|     Jack Kerouac|
+---+-----------------+
```

4. Get the table schema:

```
adb_author.printSchema()
```

The result:

```
root
 |-- id: integer (nullable = false)
 |-- name: string (nullable = false)
```

5. Get the number of the table rows:

```
adb_author.count()
```

The output:

```
10
```

Write data from Spark to ADB

1. In the PySpark shell, define a dict with connector settings for writing data to the ADB `test_pyspark_write` table:

```
opts_write = {
   "spark.adb.url": "jdbc:postgresql://10.92.40.38:5432/adb",
   "spark.adb.user": "adb_to_spark",
   "spark.adb.password": "123",
   "spark.adb.dbschema": "public",
   "spark.adb.dbtable": "test_pyspark_write"
}
```

2. Create a sample DataFrame and write its content to the test ADB table:

```
data = [
    (1, "foo"),
    (2, "bar"),
    (3, "buzz")
]
columns = ["id", "value"]
test_df = spark.createDataFrame(data=data, schema=columns)

test_df.write.format("adb") \
    .options(**opts_write) \
    .mode("overwrite") \
    .save()
```

The output:

```
24/08/28 17:34:29 INFO HikariDataSource: HikariPool-3 - Starting...
24/08/28 17:34:29 INFO HikariDataSource: HikariPool-3 - Start completed.
...
24/08/28 17:34:31 INFO OverwriteByExpressionExec: Data source write support io.arenadata.spark.adb.spark3.writer.
AdbBatchWrite@26b15eff is committing.
24/08/28 17:34:31 INFO OverwriteByExpressionExec: Data source write support io.arenadata.spark.adb.spark3.writer.
AdbBatchWrite@26b15eff committed.
```

3. Check the data availability in ADB (for example, via psql):

   - Table structure:

```
\d+ public.test_pyspark_write
```

```
                 Table "public.test_pyspark_write"
 Column |  Type  | Modifiers | Storage  | Stats target | Description
--------+--------+-----------+----------+--------------+-------------
 id     | bigint |           | plain    |              |
 value  | text   |           | extended |              |
Distributed by: (id)
```

   - Table data:

```
SELECT * FROM public.test_pyspark_write;
```

```
 id | value
----+-------
  3 | buzz
  1 | foo
  2 | bar
(3 rows)
```

ADB Spark 3 Connector options

Mikhail Serov

To Table of Contents

Read/write parameters and parameters that are necessary to connect to the ADB database are set via key/value pairs.

The connector supports the following options.

ADB Spark 3 Connector options

Key	Description	Type	Required	Default value
spark.adb.url	Database connection string	Read/write	Yes	—
spark.adb.dbschema	Name of the database schema to which the table belongs	Read/write	Yes	public
spark.adb.dbtable	Name of the database table	Read/write	Yes	—
spark.adb.driver	The full path to the JDBC driver when using a custom driver	Read/write	No	org.postgresql.Driver
spark.adb.user	User/ADB role	Read/write	Yes	—
spark.adb.password	User password in ADB	Read/write	No	—
spark.adb.server.usehostname	Allows to use the Spark 3 executor node name as a `gpfdist` server address	Read/write	No	false
spark.adb.server.env.name	Name of the environment variable whose value determines the name of the Spark 3 executor node or the IP address on which the `gpfdist` server process runs	Read/write	No	—
spark.adb.server.port	A port number or a port range of the `gpfdist` server process on the Spark 3 executor node	Read/write	No	—
spark.adb.server.nic	Name of the network interface to be used for resolving `gpfdist` server's FQDN. If this parameter is not set, the connector uses the `hostname -f` command result for FQDN resolution	Read/write	No	—
spark.adb.pool.maxsize	Maximum number of connections in the connection pool	Read/write	No	4
spark.adb.pool.timeoutms	Time in milliseconds after which an inactive connection is considered idle	Read/write	No	10000
spark.adb.pool.minidle	Minimum number of idle connections supported in the connection pool	Read/write	No	0
spark.adb.debugmode	Enables event logging mode in the `adb_spark_debug_query_log` ADB table	Read/write	No	false
spark.adb.partition.column	Name of the table column used for partitioning in Spark 3. This column should be of the `integer` or `date/time` data types	Read	No	—
spark.adb.partition.count	Number of partitions in Spark 3. Can be specified either independently or simulataneously with `spark.add.partition.column` or `spark.adb.partition.hash`	Read	No	—
spark.adb.partition.hash	Expression used as partitioning key when reading data in Spark 3. Specified simultaneously with `spark.adb.partition.count`. This expression should return a value of the `integer` data type	Read	No	—
spark.adb.table.truncate	Used when writing in the `Overwrite` mode. Performs the truncate table operation, if the value is `true`, otherwise drops the table	Write	No	false
spark.adb.create.table.with	Used when writing in the `Overwrite` and `errorIfExists` modes. Stores parameters when creating a table via the `WITH` expression	Write	No	—
spark.adb.create.table.distributedby	Used when writing in the `Overwrite` and `errorIfExists` modes. Works as distribution key when creating the target table via the `DISTRIBUTED BY` expression	Write	No	RANDOMLY
spark.adb.read.mode	Sets the mode for reading data from ADB. The possible values: ▪ `GPFDIST` — Spark fetches data from ADB via the `gpfdist` utility. ▪ `PARALLEL_CURSOR` — Spark reads data directly from ADB segments using the ADB `gp_parallel_retrieve_cursor` cursor. Using this cursor implementation is effective for retrieving large data sets in parallel, as it assumes fewer data transformations.	Read	No	GPFDIST

ADQM Spark 3 Connector

Elena Dvoryadkina

To Table of Contents

ADQM Spark 3 Connector is a multifunctional connector with support for parallel read/write operations between Apache Spark 3 and Arenadata QuickMarts (ADQM).

It has great flexibility in configuration and a multitude of features, including:

- high speed of data transmission;
- automatic data schema generation;
- flexible partitioning;
- support for push-down operators;
- support for batch operations.

IMPORTANT

Starting with ADH 3.2.4.2.b1, the connector for Spark 3 comes with the ADH bundle. You should install the Spark 3 service to use the connector. For information on the Spark 2 connector, see ADQM Spark Connector.

ADQM Spark 3 Connector overview

Elena Dvoryadkina

Contents

To Table of Contents

Architecture
Read data
Write data
Supported table engines
Supported data types

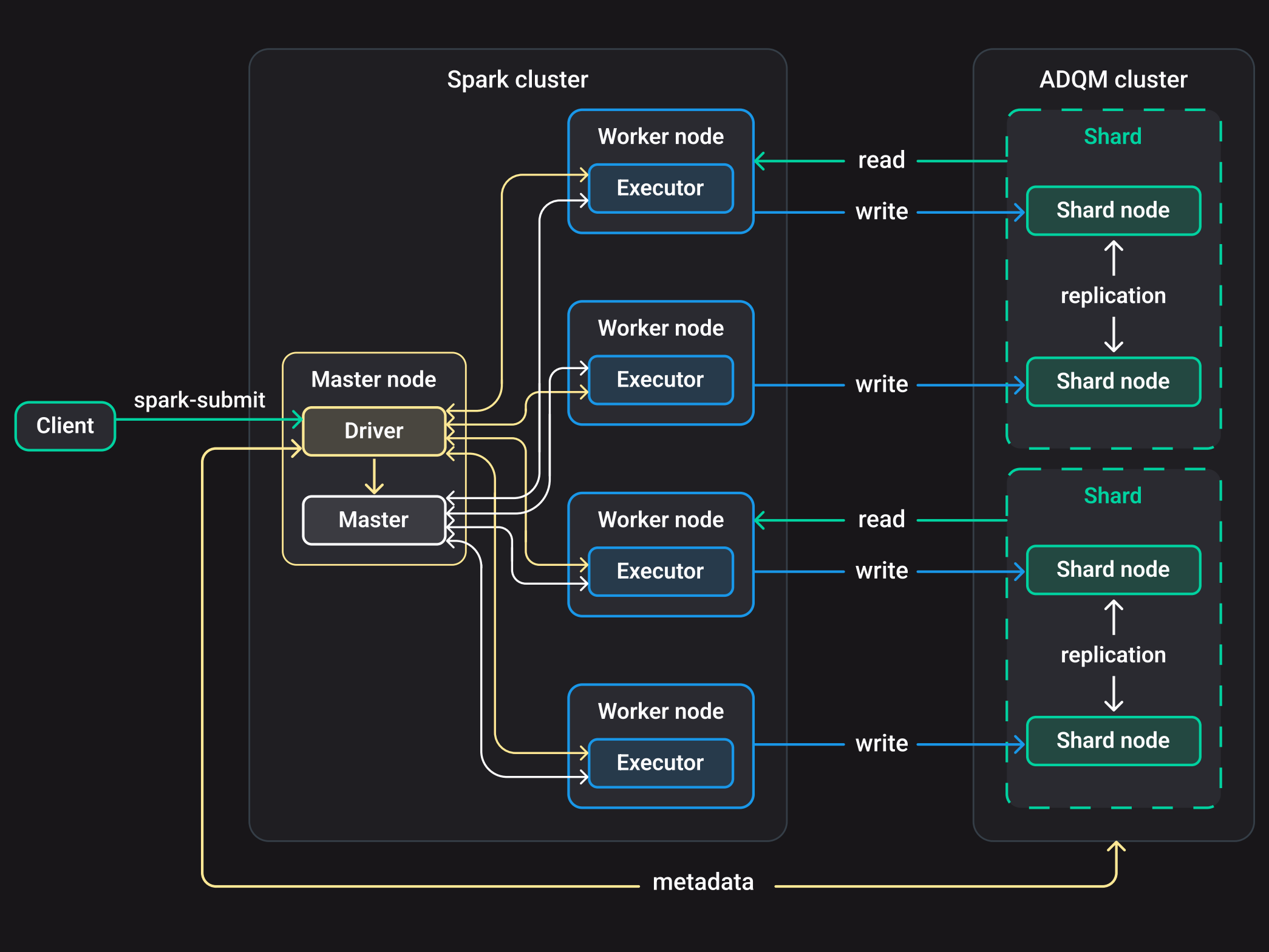
Arenadata QuickMarts (ADQM) is a column-based database management system for online query processing, which is based on the open-source ClickHouse DBMS.

Apache Spark 3 is an open-source framework for distributed processing of unstructured and semi-structured data, a part of the ecosystem of Hadoop projects.

ADQM Spark 3 Connector provides high-speed parallel data exchange between Apache Spark 3 and Arenadata QuickMarts.

Architecture

Each Spark 3 application consists of a controlling process (driver) and a number of distributed worker processes (executors). The interaction scheme is provided below.


Spark cluster
ADQM cluster
Client
spark-submit
Master node
Driver
Master
Worker node
Executor
Worker node
Executor
Worker node
Executor
Worker node
Executor
read
write
write
read
write
write
Shard
Shard node
replication
Shard node
Shard
Shard node
replication
Shard node
metadata

ADQM Spark 3 Connector architecture

Read data

To load an ADQM table into Spark 3, ADQM Spark 3 Connector initializes a driver via a connection string specified by the `spark.adqm.url` option. If the `spark.adqm.cluster.name` option is specified, the driver, connecting to `spark.adqm.url` via JDBC, also reads metadata that contains information about all shards and replicas of the specified ADQM cluster. This allows Spark 3 executors to effective read data from shard tables in parallel.

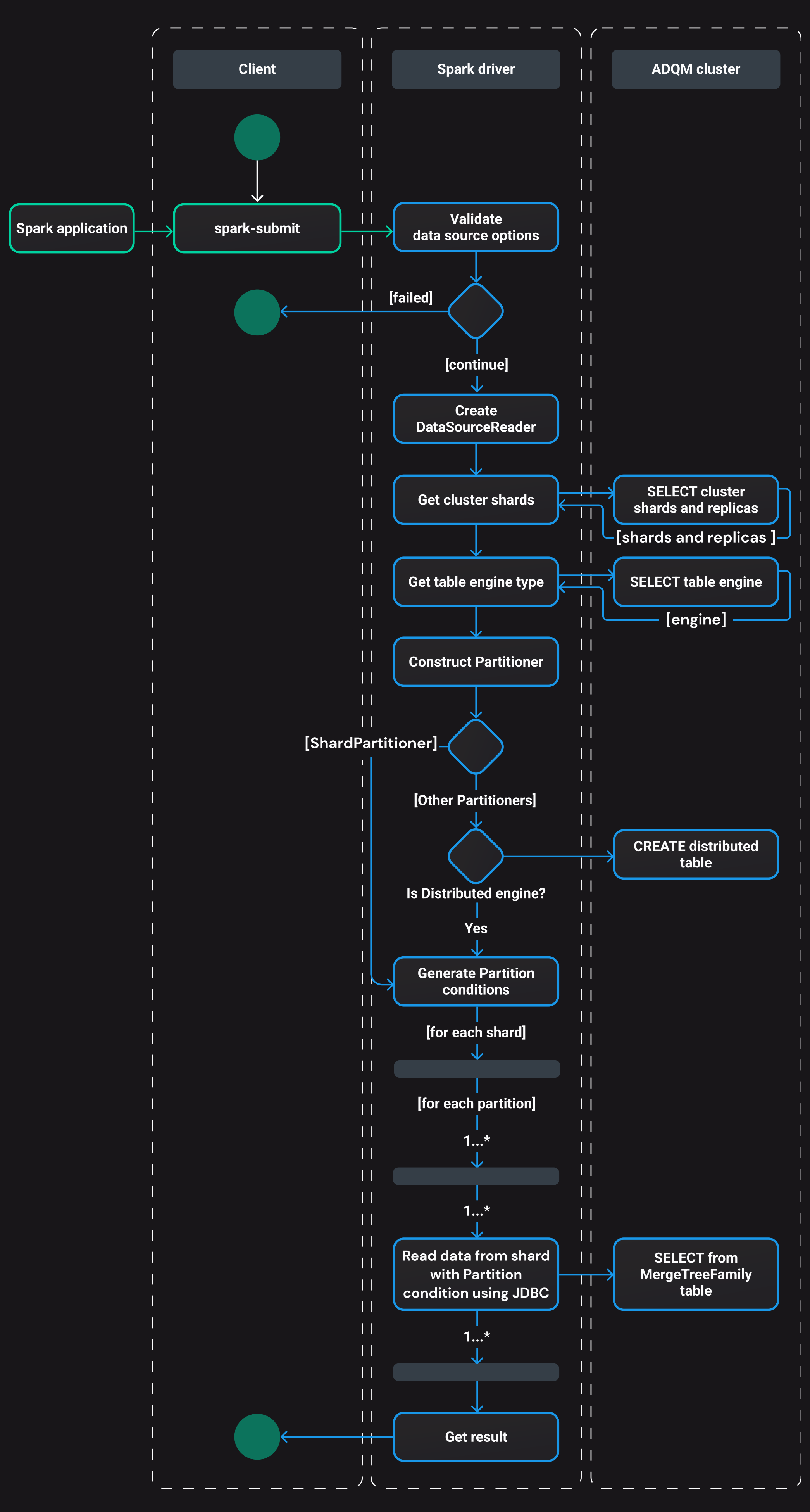
Data in ADQM is stored on shards. A Spark 3 application that uses the connector can receive data from each shard into one or more Spark 3 partitions. In the case of reading from Distributed tables, data is read from the corresponding MergeTree table on each shard in parallel.

There are currently three partitioning modes:

> NOTE
>
> The information is valid for cases when the `spark.adqm.cluster.name` option is specified and not specified.

- **According to shard_id**. Data from ADQM tables is read and distributed across Spark 3 partitions depending on a shard number. The number of Spark 3 partitions corresponds to the number of active ADQM shards. This partitioning mode does not require additional parameters and is used by default.
- **According to the specified column and number of partitions**. Data is divided into Spark 3 partitions according to value ranges of the specified column by the specified number of partitions. The number of Spark 3 partitions corresponds to the number of ranges. It is necessary to set the `spark.adqm.partition.column` and `spark.adqm.partition.count` parameters (see ADQM Spark 3 Connector options). This mode has limitations: only an integer or date/time table field can be used as a column.
- **According to the specified column**. Data is divided into Spark 3 partitions depending on unique values of the specified column. The number of Spark 3 partitions corresponds to the number of unique values in the specified column. You should set the `spark.adqm.partition.column` option (see ADQM Spark 3 Connector options). This mode is recommended for small and limited sets of values.

With this architecture, each Spark 3 executor is assigned a data processing task and a corresponding partition from previously created ones. Data exchange is performed in parallel for each partition with each shard (if the `spark.adqm.cluster.name` option is specified — with each shard of this cluster, otherwise — with shards specified in the `spark.adqm.url` option).


Client
Spark driver
ADQM cluster
Spark application
spark-submit
Validate data source options
[failed]
[continue]
Create DataSourceReader
Get cluster shards
SELECT cluster shards and replicas
[shards and replicas]
Get table engine type
SELECT table engine
[engine]
Construct Partitioner
[ShardPartitioner]
[Other Partitioners]
CREATE distributed table
Is Distributed engine?
Yes
Generate Partition conditions
[for each shard]
[for each partition]
1...*
1...*
Read data from shard with Partition condition using JDBC
SELECT from MergeTreeFamily table
1...*
Get result

Data reading algorithm

Write data

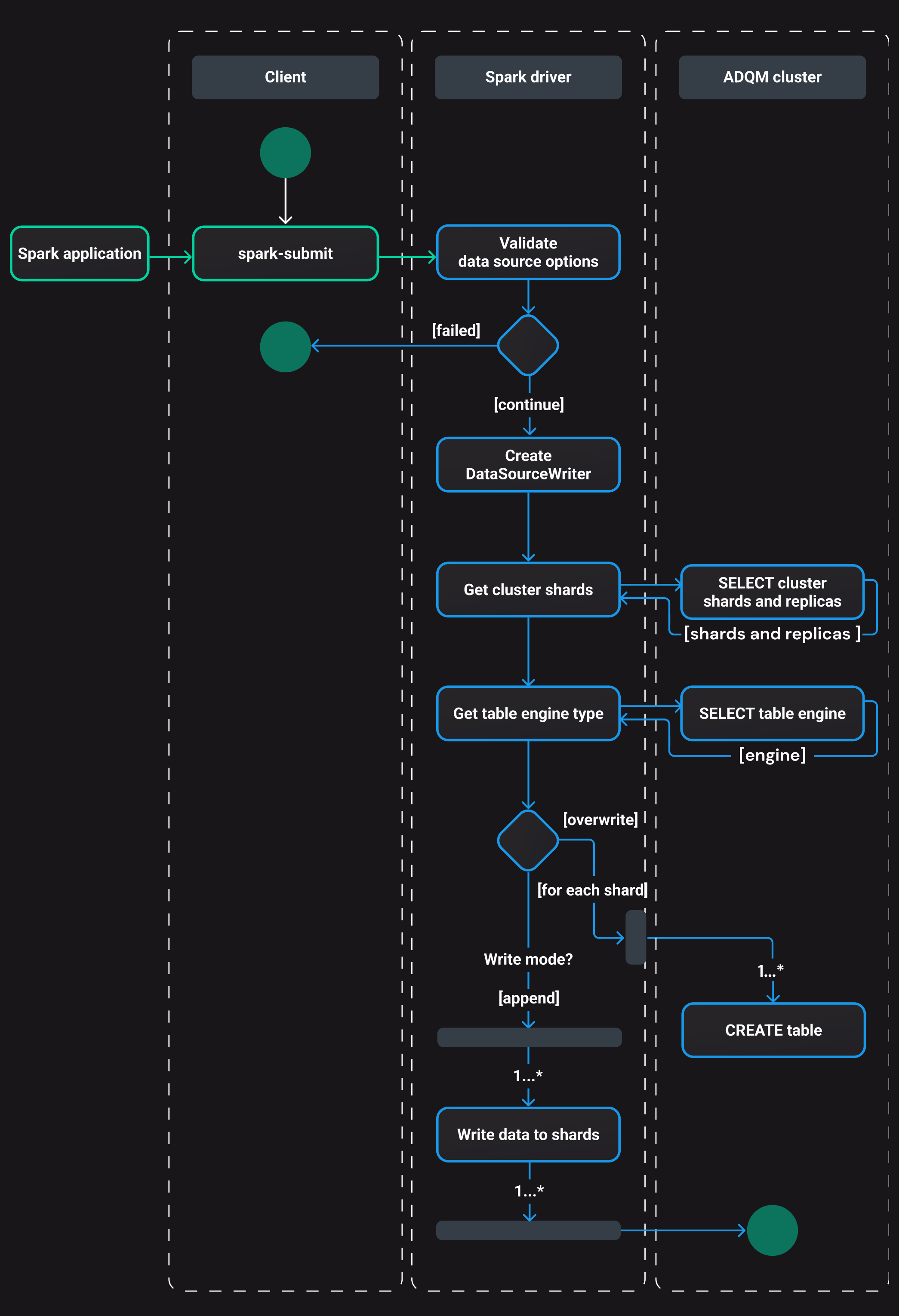
To load a table from Spark 3 into ADQM, ADQM Spark 3 Connector initializes a driver via a connection string specified by the `spark.adqm.url` option. If the `spark.adqm.cluster.name` option is specified, the driver, when connecting to `spark.adqm.url` via JDBC, also reads metadata that contains information about all shards and replicas of the specified ADQM cluster. Depending on the writing mode, each shard is prepared for data loading (tables are created or cleared).

The following writing modes are currently supported:

- `append`. Additional data is written to the target table. If the `spark.adqm.cluster.name` option is specified, the connector attempts to write data to each shard of the cluster, otherwise — only to shards specified in `spark.adqm.url`. If any of the shards does not have a table to be appended to, the loading process fails. In this mode, a target table must be pre-created on all shards.
- `errorIfExists`. If the target table exists, an error occurs. This mode assumes that the target table does not exist, it is created during the loading process. For this mode, you should use the `spark.adqm.create.table.engine` option to specify an engine of the target table to be created. If the `spark.adqm.cluster.name` option is specified, then the target table with the specified engine is created on each shard of the cluster, otherwise — the target table is created on shards specified in the `spark.adqm.url` option.
- `overwrite`. Table data is overwritten. Depending on `spark.adqm.table.truncate`, the target table is either completely deleted, or all data is cleaned up before the writing process. In the case of a complete deletion of a table, the writing process is similar to the `errorIfExists` mode, in the case of a cleanup, it is similar to the `append` mode.

To specify additional settings for target tables to be created, you can use more `spark.adqm.create.table` options (see ADQM Spark 3 Connector Options).

A data loading task is created for each Spark 3 partition. Data exchange is performed in parallel with each shard (if the `spark.adqm.cluster.name` option is specified — with each shard of the specified cluster, otherwise — only with shards specified in the `spark.adqm.url` option).


Client
Spark driver
ADQM cluster
Spark application
spark-submit
Validate data source options
[failed]
[continue]
Create DataSourceWriter
Get cluster shards
SELECT cluster shards and replicas
[shards and replicas]
Get table engine type
SELECT table engine
[engine]
[overwrite]
[for each shard]
Write mode?
[append]
1...*
CREATE table
1...*
Write data to shards
1...*

Data writing algorithm

Supported table engines

Table engine is an ADQM entity that determines how data is stored, how to write it, where to read it from, which data access is provided, and also specifies query, index and replication parameters (for more information, see Table engines in the ADQM documentation).

ADQM Spark 3 Connector supports the following ADQM table engines:

- **MergeTree engine family**. These engines are designed to quickly record a huge amount of data in parts according to the specified rules. The main MergeTree engine is supported, as well as its special versions:
  - AggregatingMergeTree;
  - CollapsingMergeTree;
  - VersionedCollapsingMergeTree;
  - ReplacingMergeTree;
  - SummingMergeTree;
  - ReplicatedMergeTree;
  - ReplicatedAggregatingMergeTree;
  - ReplicatedCollapsingMergeTree;
  - ReplicatedVersionedCollapsingMergeTree;
  - ReplicatedReplacingMergeTree;
  - ReplicatedSummingMergeTree.
- **Distributed engine**. It does not store data, but allows you to process queries distributed across multiple servers, automatically performing parallel reading (read more about creating and using Distributed tables in the Sharding article of the ADQM documentation).

Supported data types

Tables below show how ADQM and Spark 3 data types are matched during data transfers.

Transfer from ADQM to Spark 3

ADQM	Spark 3
Date	DateType
DateTime	TimestampType
DateTime64	TimestampType
Decimal	DecimalType
Decimal128	DecimalType
Decimal256	DecimalType
Decimal32	DecimalType
Decimal64	DecimalType
Float32	FloatType
Float64	DoubleType
IPv4	LongType
Int16	IntegerType
Int32	IntegerType
Int64	LongType
Int8	ShortType
String	StringType
UInt16	IntegerType
UInt32	LongType
UInt64	LongType
UInt8	ShortType
UUID	StringType
float8	DoubleType

Transfer from Spark 3 to ADQM

Spark 3	ADQM
BinaryType	String
BooleanType	UInt8
ByteType	Int8
DateType	Date
DecimalType	Decimal
DoubleType	Float64
FloatType	Float32
IntegerType	Int32
LongType	Int64
ShortType	Int16
StringType	String
TimestampType	DateTime64

Prerequisites

Elena Dvoryadkina

Contents

To Table of Contents

Checklist

Supported platforms and versions

Memory

Checklist

Make sure that the following requirements are met:

- There is access to the ADQM cluster.
- There is access to the Spark 3 cluster.
- There is network connection between all ADQM shards and the Spark 3 driver.
- There is network connection between all ADQM shards and each Spark 3 executor node.

Supported platforms and versions

- ADQM starting with version 20.8.11.17.
- Spark 3.3.x, Spark 3.4.x.
- Scala 2.13.
- ClickHouse Native JDBC 2.5.4.

Memory

In general, Spark 3 runs fine using any memory amount between 8 GB and hundreds of gigabytes per machine. We recommend to allocate 75% of the memory for Spark 3 at most — leave the rest for the operating system and buffer cache.

The memory amount that you need depends on your application. To determine the memory amount that your application uses for a certain dataset size, load a part of your dataset into the Spark 3 RDD, then use the ***Storage*** tab of the Spark 3 monitoring UI (*http://<driver-node>:4040*) to see the memory size for that part. Memory usage is affected by storage level and serialization format. See the tuning guide for tips on how to reduce the memory usage.

NOTE

Java VM does not always behave well if there is more than 200 GB of RAM. If you purchase machines with more RAM, you can run multiple worker Java VMs per node. In Spark 3 standalone mode, you can set the number of worker machines per node via the `SPARK_WORKER_INSTANCES` variable in the *conf/spark-env.sh* script. You can also set the number of cores per worker machine via the `SPARK_WORKER_CORES` variable.

Usage example

Elena Dvoryadkina

Contents

To Table of Contents

Load data from ADQM to Spark 3

Run any SQL query on an ADQM node

Write data from Spark 3 to ADQM

Before you start using ADQM Spark 3 Connector, you need to find out the following information:

- ADQM host name;
- port for interacting with ADQM;
- user credentials for connection;
- name of the database to connect to;
- name of the table to be accessed.

Examples in this article use the following data for testing:

- ADQM host — `10.92.17.146`;
- port — `9000`;
- ADQM user — `default` (no password);
- database — `default`;
- table to be read from ADQM — `users`:

```
┌─user_id─┬─name───┬───role──┐
│       1 │ john   │    admin│
│       2 │ mary   │   author│
│       3 │ andrew │   author│
│       4 │ harry  │ reviewer│
│       5 │ ann    │view only│
└─────────┴────────┴─────────┘
```

ADQM Spark 3 Connector communicates with ADQM using a JDBC connection. The ClickHouse Native JDBC driver comes with the connector.

Commands from examples below should be executed in the interactive shell for Scala — spark3-shell.

Load data from ADQM to Spark 3

1. Specify the connector settings for reading data from the ADQM `default.users` table:

```
val options_read = Map(
    "spark.adqm.url" -> "jdbc:clickhouse://10.92.17.146:9000",
    "spark.adqm.user" -> "default",
    "spark.adqm.dbschema" -> "default",
    "spark.adqm.dbtable" -> "users")
```

2. Create a DataFrame:

```
val adqm_users = spark.read.format("adqm").options(options_read).load()
```

Result of command execution:

```
adqm_users: org.apache.spark.sql.DataFrame = [user_id: int, name: string ... 1 more field]
```

3. Check the DataFrame’s content:

```
adqm_users.show()
```

```
+-------+------+---------+
|user_id|  name|     role|
+-------+------+---------+
|      1|  john|    admin|
|      2|  mary|   author|
|      3|andrew|   author|
|      4| harry| reviewer|
|      5|   ann|view only|
+-------+------+---------+
```

Run any SQL query on an ADQM node

1. Set additional settings for a Spark session and enable auxiliary functions:

```
spark.stop
val spark1 = org.apache.spark.sql.SparkSession.
    builder().
    master("local[*]").
    appName("spark_example").
    config("spark.adqm.url","jdbc:clickhouse://10.92.17.146:9000").
    config("spark.adqm.user","default").
    getOrCreate()
import io.arenadata.spark.adqm.implicits._
```

2. Create an arbitrary table in ADQM and fill it with test data:

```
spark1.executeAdqmQueryOnShard("create table default.test_spark(id integer) ENGINE  = MergeTree() ORDER BY id;")
spark1.executeAdqmQueryOnShard("insert into default.test_spark values(1);")
spark1.executeAdqmQueryOnShard("insert into default.test_spark values(2);")
spark1.executeAdqmQueryOnShard("insert into default.test_spark values(3);")
```

3. Create a DataFrame based on the new ADQM table:

```
val test = spark1.executeAdqmSelectQueryOnShard("select * from default.test_spark;")
```

```
test: org.apache.spark.sql.DataFrame = [id: int]
```

4. Check the DataFrame’s content:

```
test.show()
```

```
+---+
| id|
+---+
|  1|
|  2|
|  3|
+---+
```

5. Check in ADQM (for example, in the clickhouse-client console client) that the table has been created and populated with data:

```
SELECT * FROM test_spark;
```

```
┌─id─┐
│  1 │
│  2 │
│  3 │
└────┘
```

Write data from Spark 3 to ADQM

1. Specify the connector settings:

```
val options_write = Map(
    "spark.adqm.url" -> "jdbc:clickhouse://10.92.17.146:9000",
    "spark.adqm.user" -> "default",
    "spark.adqm.dbschema" -> "default",
    "spark.adqm.dbtable" -> "test_spark_write",
    "spark.adqm.create.table.engine" -> "MergeTree")
```

2. Run the following command to create a `test_spark_write` table in ADQM and write data from the DataFrame created in the previous example into it:

```
test.write.format("adqm").options(options_write).mode(org.apache.spark.sql.SaveMode.Overwrite).save()
```

3. Check in ADQM that data has been added to the new table:

```
SELECT * FROM test_spark_write;
```

```
┌─id─┐
│  1 │
│  2 │
│  3 │
└────┘
```

ADQM Spark 3 Connector options

Elena Dvoryadkina

To Table of Contents

Parameters for connecting to an ADQM database and read/write parameters are set via key/value pairs.

The following options are available for ADQM Spark 3 Connector.

ADQM Spark 3 Connector options

Key	Description	Operation type	Required	Default value
spark.adqm.url	Database connection string via a driver that uses the TCP protocol	Read/write	Yes	—
spark.adqm.dbschema	Name of the database schema to which the table belongs	Read/write	Yes	default
spark.adqm.dbtable	Name of the database table	Read/write	Yes	—
spark.adqm.user	ADQM user/role	Read/write	Yes	—
spark.adqm.password	ADQM user password	Read/write	No	—
spark.adqm.cluster.name	ADQM cluster name	Read/write	No	—
spark.adqm.jdbc.pool.maxsize	Maximum number of connections in the connection pool	Read/write	No	4
spark.adqm.jdbc.pool.timeout	Time in milliseconds after which an inactive connection is considered idle	Read/write	No	10000
spark.adqm.jdbc.pool.minidle	Minimum number of idle connections supported in the connection pool	Read/write	No	0
spark.adqm.partition.column	Name of the table column used for partitioning in Spark 3. This column should be of the integer or date/time type	Read	No	—
spark.adqm.partition.count	Number of partitions in Spark 3. Can only be specified in conjunction with `spark.adqm.partition.column`	Read	No	—
spark.adqm.insert.batchsize	Batch size when inserting into a target table	Write	No	10000
spark.adqm.insert_distributed_sync	Insert mode in ADQM: `0` — insert in the asynchronous mode; `1` — insert in the synchronous mode	Write	No	0
spark.adqm.table.truncate	Used for the `overwrite` writing mode. Performs the truncate table operation, if the value is `true`, otherwise — drops the table	Write	No	false

For the `overwrite` mode with the `spark.adqm.table.truncate` option set to `true` and for the `errorIfExists` mode, there are additional options described in the table below. In other modes, these options are ignored.

ADQM Spark 3 Connector additional options

Key	Description	Required
spark.adqm.create.table.engine	Target table engine	Yes
spark.adqm.create.table.storage_policy	Target table storage policy	No
spark.adqm.create.table.distributed.sharding_key	Sharding key for a Distributed target table	No
spark.adqm.create.table.merge_tree.sorting_key	Sorting key for a target table of the MergeTree family. If `spark.adqm.create.table.merge_tree.primary_key` is specified, the value is `primaryKey`, otherwise — `tuple()`	No
spark.adqm.create.table.merge_tree.sampling_key	Sampling key for a target table of the MergeTree family	No
spark.adqm.create.table.merge_tree.primary_key	Primary key for a target table of the MergeTree family	No
spark.adqm.create.table.merge_tree.partition_key	Partitioning key for a target table of the MergeTree family	No
spark.adqm.create.table.summing_merge_tree.columns	Columns of a (Replicated)SummingMergeTree target table	No
spark.adqm.create.table.replacing_merge_tree.version	Column with version of data records for a (Replicated)ReplacingMergeTree target table	No
spark.adqm.create.table.collapsing_merge_tree.sign	Sign of a (Replicated)CollapsingMergeTree/(Replicated)VersionedCollapsingMergeTree table	No
spark.adqm.create.table.collapsing_merge_tree.version	Version of a (Replicated)CollapsingMergeTree/(Replicated)VersionedCollapsingMergeTree target table	No

PySpark

Konstantin Alpashkin

Contents

To Table of Contents

Cluster requirements

Use PySpark and spark3-submit

Use PySpark shell

Steps for a secured cluster

PySpark is a Python wrapper for Spark that allows invocation of native Spark methods from the Python code. The core of PySpark is the Py4J library that allows a Python interpreter to manipulate Spark objects in a running JVM.

This section describes the use of PySpark via `spark3-submit` and PySpark shell.

Cluster requirements

The examples demonstrated in this section should run in the following environment:

- ADH cluster nodes are CentOS 7-powered machines. For other OS types, the commands/paths may slightly differ.
- ADH cluster 3.1.2.1.b1 or later is installed. The cluster has the Spark3 service installed.
  During the cluster installation, ADCM also installs a Python3 interpreter to */opt/pyspark3-python/* on the hosts with the Spark3 Client component. This Python is used in the scenarios below, however, you can use a custom Python 3.10+ instance to run PySpark.
- A separate HDFS directory for loading test data is created and the user who runs PySpark is the owner of this directory.

```
$ sudo -u hdfs hdfs dfs -mkdir /user/admin
$ sudo -u hdfs hdfs dfs -chown admin:admin /user/admin
```

Use PySpark and spark3-submit

The following scenario shows how to submit a Spark job in Python using `spark3-submit` in the cluster mode.

1. Set the `PYSPARK_PYTHON` environment variable on all cluster hosts to allow Spark to run the specific Python executable.
   For this, in ADCM, go to ***Clusters*** → ***<your_cluster_name>*** → ***Services*** → ***Spark3*** → ***Primary configuration*** and add the line `export PYSPARK_PYTHON=/opt/pyspark3-python/bin/python3` to the ***spark-env.sh*** field (the field appears after selecting ***Show advanced***). Then, save the configuration and restart the Spark3 service.
2. Create a test script named *test.py*.

```
from pyspark import SparkConf
from pyspark import SparkContext
from pyspark.sql import SparkSession

conf = SparkConf()
conf.setMaster('yarn')
conf.setAppName('spark-yarn')
sparkSession = SparkSession.builder.appName("spark-yarn").config(conf=conf).getOrCreate()

data = [('First', 1), ('Second', 2), ('Third', 3), ('Fourth', 4), ('Fifth', 5)]
df = sparkSession.createDataFrame(data) ①
df.write.csv("hdfs:///user/admin/tmp/test") ②
quit()
```

① Creates a sample DataFrame.

② Writes the DataFrame contents to HDFS as a CSV file.

3. Run `spark3-submit` and provide *test.py* as an argument. In this scenario, `spark3-submit` must run under the `admin` user.

```
$ /bin/spark3-submit \
    --deploy-mode cluster \ ①
    --master yarn \ ②
    test.py
```

① Runs Spark in the cluster mode.

② The master URL to run Spark on a YARN cluster. The cluster location is resolved based on the `HADOOP_CONF_DIR` or `YARN_CONF_DIR` environment variables.

4. Verify HDFS writes:

```
$ sudo -u hdfs hdfs dfs -ls /user/admin/tmp/test/
```

Observe the DataFrame parts persisted in HDFS:

```
Found 3 items
-rw-r--r--   3 admin admin          0 2023-07-10 12:51 /user/admin/tmp/test/_SUCCESS
-rw-r--r--   3 admin admin         17 2023-07-10 12:51 /user/admin/tmp/test/part-00000-1a966f98-6c1a-467b-9564-dbbd65dd32a2-c000.
csv
-rw-r--r--   3 admin admin         25 2023-07-10 12:51 /user/admin/tmp/test/part-00001-1a966f98-6c1a-467b-9564-dbbd65dd32a2-c000.
csv
```

Use PySpark shell

PySpark shell allows you to run Spark jobs in the interactive mode. The following example shows how to install PySpark shell and run a Spark job in the local mode.

1. Create a new virtual environment using Python from */opt/pyspark3-python/*:

```
$ cd ~
$ mkdir pyspark-demo
$ /opt/pyspark3-python/bin/python3 -m venv pyspark-demo/venv
```

2. Activate the virtual environment:

```
$ source pyspark-demo/venv/bin/activate
```

1. Run PySpark:

```
$ pyspark3
```

You should see a similar output:

```
Python 3.10.4 (main, Sep  7 2023, 08:17:33) [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
Picked up _JAVA_OPTIONS: -Djava.io.tmpdir=/var/tmp
Picked up _JAVA_OPTIONS: -Djava.io.tmpdir=/var/tmp
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/spark3/jars/log4j-slf4j-impl-2.17.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/09/28 08:37:05 WARN HiveConf: HiveConf of name hive.metastore.event.db.notification.api.auth does not exist
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.3.2
      /_/

Using Python version 3.10.4 (main, Sep  7 2023 08:17:33)
Spark context Web UI available at http://ka-adh-1.ru-central1.internal:4040
Spark context available as 'sc' (master = yarn, app id = application_1695647776253_0020).
SparkSession available as 'spark'.
```

2. Submit the following Python code directly to the PySpark shell. This Spark job creates a sample DataFrame and writes it to HDFS through a NameNode endpoint.

```
from pyspark import SparkConf
from pyspark import SparkContext
from pyspark.sql import SparkSession

conf = SparkConf()
conf.setMaster('yarn')
conf.setAppName('spark-yarn')
sparkSession = SparkSession.builder.appName("spark-yarn").config(conf=conf).getOrCreate()

data = [('First', 1), ('Second', 2), ('Third', 3), ('Fourth', 4), ('Fifth', 5)]
df = sparkSession.createDataFrame(data)
df.write.csv("hdfs://<active-nn-host>:8020/user/admin/tmp/test-nn") ①
quit()
```

① `<active-nn-host>` must point to an active NameNode. To identify the active NameNode, use the command below:

```
$ sudo -u hdfs hdfs haadmin -getAllServiceState
```

A sample output:

```
ka-adh-1.ru-central1.internal:8020                 standby
ka-adh-3.ru-central1.internal:8020                 active
```

3. Verify HDFS writes:

```
$ sudo -u hdfs hdfs dfs -ls /user/admin/tmp/test-nn/
```

Observe the DataFrame persisted in HDFS:

```
-rw-r--r--   3 admin hadoop          9 2023-03-10 16:53 /user/admin/tmp/test-nn/part-00001-dedba941-113e-4fd6-871d-e87dd3291e57-
c000.csv
-rw-r--r--   3 admin hadoop          8 2023-03-10 16:53 /user/admin/tmp/test-nn/part-00002-dedba941-113e-4fd6-871d-e87dd3291e57-
c000.csv
-rw-r--r--   3 admin hadoop         17 2023-03-10 16:53 /user/admin/tmp/test-nn/part-00003-dedba941-113e-4fd6-871d-e87dd3291e57-
c000.csv
```

Steps for a secured cluster

If you have enabled security for your cluster, that is:

- installed Apache Ranger within ADPS;
- kerberized both ADPS and ADH;
- joined all Hadoop nodes into a domain and SSSD is used;
- activated the HDFS/YARN plugins;
- configured Ranger policies for HDFS/YARN plugins.

For such a cluster, the steps on using PySpark are **identical**, the main requirements are:

1. Your user/domain group has access to specific HDFS paths, defined in Ranger ACL.
2. You have to get a Kerberos ticket using `kinit` as shown below:

```
$ kinit myuser@EXAMPLE_REALM.COM
```

Spark Connect usage

Konstantin Alpashkin

Contents

To Table of Contents

- Spark3 Connect component
- Use Spark Connect with Spark applications
  - Use spark3-submit
  - Use PySpark shell
- Authentication

This article shows examples of working with a Spark3 cluster using Spark Connect.

Spark3 Connect component

To interact with your Spark3 cluster via Spark Connect, the corresponding component (Spark3 Connect) of the Spark3 service has to be installed in your ADH cluster. You can add this component using Spark3 service actions in ADCM. After the component is installed, it can be used out-of-the-box without any configurations.

Use Spark Connect with Spark applications

A client Spark application that employs Spark Connect is very similar to a regular Spark app. The key feature is using a **remote** SparkSession object, which is responsible for connecting to a remote Spark Connect server. With the remote session type, the Spark application code runs on the YARN cluster where Spark Connect server is installed. Once a remote `SparkSession` is created, it can be used as if it were a regular Spark session object. All the communication with the remote Spark Server is managed automatically by the Spark Connect client library.

The main ways of creating a remote Spark session are shown below.

Use spark3-submit

When using */bin/spark3-submit* to launch Spark client applications, a remote session can be created in the application code as shown below.

```
from pyspark import SparkConf
from pyspark import SparkContext
from pyspark.sql import SparkSession

conf = SparkConf()
conf.setMaster('yarn')
conf.setAppName('spark-yarn')
sparkSession = SparkSession
    .builder
    .appName("spark-yarn")
    .config(conf=conf)
    .remote("sc://<sc_host>:<sc_port>") ①
    .getOrCreate()

data = [('First', 1), ('Second', 2), ('Third', 3)]
df = sparkSession.createDataFrame(data)
df.write.csv("hdfs:///user/admin/tmp/test") ②
quit()
```

① Creates a remote Spark session. `<sc_host>:<sc_port>` is the gRPC endpoint of the Spark3 Connect component. You can find the up-to-date endpoint on the ***Clusters*** → ***<your_cluster_name>*** → ***Services*** → ***Spark3*** → ***Info*** page in ADCM.

② Writes the DataFrame to HDFS. All the operations are performed on the Spark3 cluster with the Spark3 Connect component.

Once submitted, the following output indicates that the application has successfully connected to the Spark Connect server:

```
Client connected to the Spark Connect server at ka-adh-1.ru-central1.internal:15002
```

NOTE

Only one `SparkSession` object can exist within a Spark application at a time. You can kill a Spark session using SparkSession.stop() and create a remote session as shown in the example above.

Use PySpark shell

When using PySpark shell (*/usr/bin/pyspark3*), the `SparkSession` object is created automatically during the shell startup.

```
...
SparkSession available as 'spark'.
```

To force PySpark shell to create the remote session type, use the `--remote` flag when running */usr/bin/pyspark3*. For example:

```
/bin/pyspark3 --remote "sc://<sc_host>:<sc_port>"
```

Where `<sc_host>:<sc_port>` is a gRPC endpoint of the Spark3 Connect component. You can find the up-to-date endpoint on the ***Clusters*** → ***<your_cluster_name>*** → ***Services*** → ***Spark3*** → ***Info*** page in ADCM.

Authentication

Spark Connect does not provide built-in authentication and the gRPC channel between a Spark client app and the Spark Connect server remains unsecure. A way to secure this communication channel is through using gRPC proxies.

Work with Livy

Konstantin Alpashkin

Contents

To Table of Contents

- Prerequisites
  - Add user permissions
  - Enable Livy impersonation
- Session creation example
- Get session status
- Delete session
- Common issues

Apache Livy is a Spark3 service component that allows interaction with a Spark cluster via HTTP. Livy exposes a REST API through which one can invoke common Spark functions, like:

- submit Spark jobs;
- retrieve synchronous/asynchronous results;
- manage Spark Context.

This article describes the use of basic Livy operations like create/view/delete a session and also shows how to use Livy impersonation. All the examples in this article refer to the Batch session type, however the REST API for working with Interactive sessions is very similar. Batch sessions are intended for asynchronous, non-interactive execution of Spark jobs and are created automatically when a Spark app starts in the batch mode.

For more details, see Apache Livy REST API.

TIP

It is recommended to use the Livy server distributed with the Spark3 service. This Livy instance runs on the `8999` port and contains latest bug fixes.

Prerequisites

To run the examples presented in this section, you need to prepare your ADH cluster.

Add user permissions

First, add permissions to Hadoop directories and copy Spark sample JARs to HDFS.

```
$ sudo -u hdfs hdfs dfs -mkdir /user/livyDemoUser
$ sudo -u hdfs hdfs dfs -chown livyDemoUser:hadoop /user/livyDemoUser
$ sudo -u hdfs hdfs dfs -chmod -R a+rwx /user/livyDemoUser
$ sudo -u hdfs hdfs dfs -chmod -R a+rwx /var/log/spark3/apps
$ sudo -u hdfs hdfs dfs -mkdir /user/livyDemoUser/spark_examples
$ sudo -u hdfs hdfs dfs -put /usr/lib/spark3/examples/jars/* /user/livyDemoUser/spark_examples
```

Enable Livy impersonation

By default, when an HTTP request reaches Livy and spawns a new YARN/Spark job, the new job process starts as `livy` user. Livy impersonation allows you to substitute the user name and run the process as a different user. To enable impersonation, follow the steps below:

1. In ADCM, go to ***<YourClusterName> → Services → HDFS → Custom core-site.xml*** and specify the `hadoop.proxyuser.livy.groups = *` and `hadoop.proxyuser.livy.hosts = *` properties. Allowing all groups/hosts (`*`) is a bad practice from the security perspective so it is used solely for demonstration purpose.
2. Ensure that */etc/livy/conf/livy.conf* (located on the same host as Livy server) contains the `livy.impersonation.enabled = True` parameter.
3. Restart HDFS, YARN, and Spark.
4. If Ranger plugin is enabled, add corresponding policies for `livyDemoUser` used in the examples.

Session creation example

The example below shows how to create a new Batch session that runs the SparkPi job:

1. First, create a JSON file that will be POSTed to Livy as HTTP body.

```
{
    "file": "/user/livyDemoUser/spark_examples/spark-examples_2.11-2.3.2.3.1.0.15-1.jar", ①
    "className": "org.apache.spark.examples.SparkPi", ②
    "name": "test-livy-1", ③
    "args": [10], ④
    "proxyUser": "livyDemoUser" ⑤
}
```

① The path to a sample Spark JAR to run.

② The main class of the SparkPi application.

③ The name for the new session. Must be unique cluster-wide.

④ The list of command-line arguments passed to the Spark application.

⑤ The user name for impersonation. This user will be used to run the new Spark job process.

2. Submit the following POST request to create a new batch session.

```
$ curl -X POST 'http://<livyServerHost>:8999/batches'
-H "X-Requested-By: livyDemoUser" ①
-d @request.json -H 'Content-Type: application/json'
```

① A mandatory header for CSRF protection

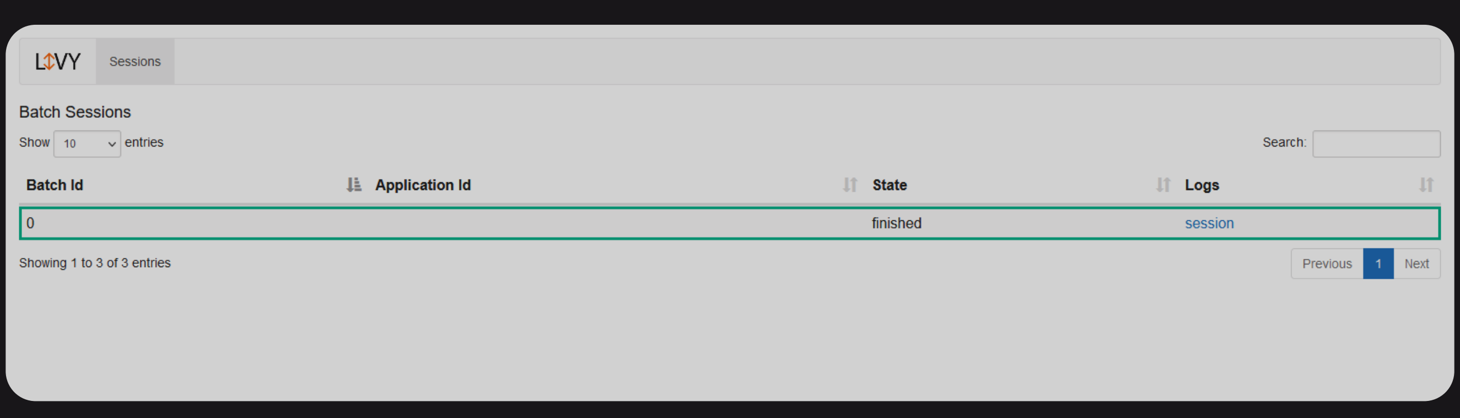
NOTE

If your ADH cluster is kerberized, you should enable SPNEGO negotiation for curl. For this, add the `--negotiate -v -u:` flag to your curl requests.

The response from Livy looks as follows:

```
{"id":1,"name":"test-livy-2","state":"starting","appId":null,"appInfo":{"driverLogUrl":null,"sparkUiUrl":null},"log":["stdout:
","\nstderr: ","\nYARN Diagnostics: "]}
```

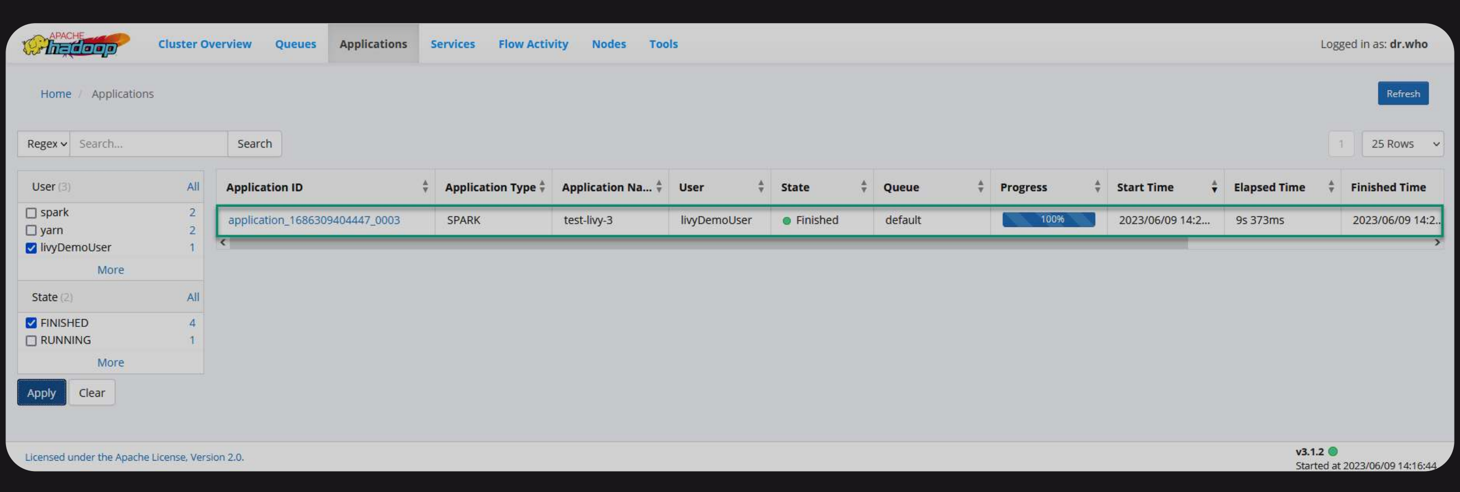
When Livy gets the request, it creates a new session named `test-livy-1` (the `name` field in request.json). The new session is displayed in Livy UI.



Batch Id	Application Id	State	Logs
0		finished	session

New session in Livy UI

The new session also appears in YARN(UI2).



Application ID	Application Type	Application Na...	User	State	Queue	Progress	Start Time	Elapsed Time	Finished Time
application_1686309404447_0003	SPARK	test-livy-3	livyDemoUser	Finished	default	100%	2023/06/09 14:2...	9s 373ms	2023/06/09 14:2...

New session in YARN

Get session status

To get the status of a batch session, submit the following request.

```
$ curl -X GET 'http://<livyServerHost>/batches/x'
```

where `x` is the session ID, for example `123`. The response from Livy looks as follows:

```
{"id":2,"name":"test-livy-4","state":"starting","appId":null,"appInfo":{"driverLogUrl":null,"sparkUiUrl":null},"log": [...]}
```

To get all sessions, submit a request to the following endpoint.

```
$ curl -X GET 'http://<livyServerHost>/batches'
```

This returns a list of all sessions with metadata, for example:

```
{"from":0,"total":4,"sessions":[{"id":0,"name":"test-livy-1","state":"finished","appId":null,"appInfo":
{"driverLogUrl":null,"sparkUiUrl":null},"log":[...]},...]}
```

Delete session

To delete a session, make the following request.

```
$ curl -X DELETE 'http://<livyServerHost>:8999/batches/x'
-H 'X-Requested-By: livyDemoUser'
```

Where `x` is the ID of the session to delete.

Common issues

The most common issues that you may encounter while using Livy are described below:

- Impersonation error.
  You may observe the following exception in Livy logs:

```
org.apache.hadoop.security.authorize.AuthorizationException: User: livy is not allowed to impersonate <username> ...
```

  This indicates misconfigured impersonation. See Enable Livy impersonation for more details.

- `HTTP ERROR: 400` in the curl response.
  This error indicates a missing required HTTP header for CSRF protection. Try adding `-H "X-Requested-By: <userName>"` to your curl requests.
- Errors related to Ranger policies.
  If Ranger plugin is enabled, the user can’t start a new session in the default YARN queue. For this, you need to set corresponding Ranger policies for YARN.

Scheduler jobs

Konstantin Alpashkin

Contents

To Table of Contents

- Schedule across a cluster
  - Static resource allocation
  - Dynamic resource allocation
  - Use an external shuffle service
- Schedule within application
  - FIFO and Fair sharing modes
  - Fair scheduler pools

When you run Spark in the cluster mode, there are two scopes where scheduling and resource allocation is applicable:

- Scheduling across a cluster. This assumes scheduling of separate Spark applications that run on different cluster nodes.
- Scheduling within an application. This assumes scheduling of parallel Spark jobs like `collect`, `save`, etc., within one Spark application.

Schedule across a cluster

To run Spark in the cluster mode, a cluster manager like YARN can be used to schedule Spark jobs and allocate resources for them on each cluster host.

Static resource allocation

One way to control resource allocation with YARN is by specifying a fixed resource amount when running `spark3-submit` (hence called static resource allocation). Below is an example of running Spark on YARN with special flags that specify the amount of allocated resources.

```
$ ./bin/spark3-submit --class <app-class-name> \
    --master yarn \ ①
    --deploy-mode cluster \ ②
    --driver-memory 4g \ ③
    --num-executors ④
    --executor-memory 2g \ ⑤
    --executor-cores 1 \ ⑥
    ...
```

① Defines the cluster manager to use.

② Instructs Spark to run the driver program on worker nodes.

③ The maximum memory to allocate for the driver program.

④ The number of executors to allocate on the cluster.

⑤ The maximum memory to allocate per one executor.

⑥ The number of CPU cores to allocate per one executor.

Dynamic resource allocation

Another approach to control the allocation of cluster resources is the Spark's built-in dynamic resource allocation feature. With this feature enabled, Spark can dynamically allocate and deallocate executors based on the application workload. To enable the feature for Spark/Spark3 ADH services, use the Switch dynamic resource allocation action in ADCM UI.

Use an external shuffle service

When dynamic resource allocation is enabled, you might want to switch to an external shuffle service that provides better reliability and resource consumption. Spark shuffle service (SS) is responsible for moving blocks of data between Spark executors or between executors and the driver program during the shuffle phase. By default, Spark uses the SS implementation for YARN that runs on every NodeManager of the YARN cluster. The default SS implementation is available at */usr/lib/spark3/yarn/spark-{version}-yarn-shuffle.jar*. However, this SS implementation has a drawback — it removes all the shuffle data persisted on disk if an executor dies.

For maximum reliability and speed, you might want to use an external SS implementation, and below are the steps on wiring your SS to the Spark cluster:

1. Add the JAR with your SS implementation to the classpath, e.g */usr/lib/spark3/**. When creating your own SS implementation, you can refer to */usr/lib/spark3/yarn/spark-{version}-yarn-shuffle.jar* as an example.
2. In ADCM, go to ***Clusters*** → ***<YOUR_CLUSTER>*** → ***Services*** → ***YARN*** → ***Primary configuration***, and specify your SS class name in the `yarn.nodemanager.aux-services.spark_shuffle.class` field.
3. Increase each NodeManager's heap size by setting `YARN_HEAPSIZE` (1000 by default) in *etc/hadoop/yarn-env.sh* to avoid GC issues during the shuffle phase.
4. Restart YARN.

Alternatively, you can provide SS configuration in the *spark-shuffle-site.xml* file and put this file to the SS classpath. For more details on SS configuration properties, see Spark documentation.

Schedule within application

Inside a Spark application (hereinafter an "application" refers to a single SparkContext instance) scheduling and resource allocation is also applicable as multiple parallel Spark jobs can run simultaneously if they are submitted from separate threads. In this section a "job" means a Spark action like `reduce`, `save`, `collect`, etc., as well as any related tasks required to evaluate that action.

FIFO and Fair sharing modes

By default, Spark jobs run in the FIFO order. Once submitted, each job gets split into stages (for example, map/reduce phases), and the first job to start gets the priority on all available resources while its stages have tasks to run. Then, the second job gets the priority, and so on. If a job at the head of the queue does not need all the available resources, the pending jobs can start right away, but if the jobs at the head of the queue are compute-intensive, the queued jobs may be delayed significantly.

Apart from FIFO, Spark can also run jobs in the fair sharing mode. In this mode, Spark assigns tasks between jobs in a round-robin fashion, so that all jobs get roughly equal shares of cluster resources. This means that short jobs submitted while a long job is still running can start receiving resources right away without waiting for the long job to complete, thus still showing good response time. Fair sharing is disabled by default and can be enabled by setting `spark.scheduler.mode=FAIR` on a SparkContext instance as shown below:

```
val conf = new SparkConf()
          .setMaster(...)
          .setAppName(...)
conf.set("spark.scheduler.mode", "FAIR")
val sc = new SparkContext(conf)
```

Fair scheduler pools

Spark allows you to group jobs into pools and set various scheduling parameters (e.g. weight) for each pool. This can be useful to create high-priority pools for more important jobs.

By default, all submitted jobs go to the default pool. You can force jobs to go to a specific pool by adding the `spark.scheduler.pool` local property to your SparkContext. This must be done in the same thread where jobs are submitted, for example:

```
sc.setLocalProperty("spark.scheduler.pool", "mPool")
```

By default, each pool gets an equal share of the cluster, and inside each pool, jobs run in the FIFO order. For example, if you create one pool per user, then each user will get an equal share of cluster resources, and the queries from each user will run one-by-one.

You can tune the scheduler pools behavior using the configuration properties below:

- `schedulingMode`. Defines whether jobs within a pool should run either in the FIFO order or share the pool's resources fairly. The value should be either `FIFO` (default) or `FAIR`.
- `weight`. Defines the pool's priority relative to other pools. For example, if you set a pool's weight to `2`, this pool gets 2x more resources than other active pools. Using high weight values (for example, `1000`), you can implement the priority for several pools. The default weight is `1`.
- `minShare`. Apart from `weight`, each pool can be given minimum shares as a number of CPU cores. The fair scheduler always attempts to meet all active pools' minimum shares before redistributing extra resources according to the weights. Therefore, the `minShare` property can be another way to ensure that a pool can always get up to a certain number of resources (e.g. 10 cores) quickly without giving it a high priority for the rest of the cluster. By default, each pool's `minShare` is 0.

The above properties should be defined in the *fairscheduler.xml* file similar to */etc/spark3/conf.dist/fairscheduler.xml.template* as shown below.

```
<allocations>
  <pool name="production">
    <schedulingMode>FAIR</schedulingMode>
    <weight>1</weight>
    <minShare>2</minShare>
  </pool>
  <pool name="test">
    <schedulingMode>FIFO</schedulingMode>
    <weight>2</weight>
    <minShare>3</minShare>
  </pool>
</allocations>
```

Then, you should either put your *fairscheduler.xml* on the classpath, or set the SparkConf `spark.scheduler.allocation.file` property that points to your *fairscheduler.xml*, for example:

```
conf.set("spark.scheduler.allocation.file", "file:///path/to/local-file")
conf.set("spark.scheduler.allocation.file", "hdfs:///path/to/hdfs-file")
```

For more details on fair scheduler pool properties, see Spark documentation.

Logging in Spark

Konstantin Alpashkin

Contents

To Table of Contents

- View logs
  - Text logs
  - History Server web UI
- Logging configuration

By default, Spark logging is enabled and all logs are stored to the */var/log/spark3/apps* directory in HDFS. Under this base directory, Spark creates subdirectories with logs for each submitted Spark application.

Spark components like History Server and Livy also log their activity on the local file system under following directories:

- */var/log/spark3/history* — stores History Server logs.
- */var/log/livy-spark3* — stores Livy logs.

View logs

There are two ways to view Spark logs:

- As plain text files.
- Using History Server web UI. In this case, Spark logs must be stored in HDFS so History Server could access all the log data generated by cluster nodes.

Text logs

Spark stores logs as JSON files that consist of individual records called **events**. To view logs as text, do the following:

1. List the contents of the base log directory (`spark.eventLog.dir`).

```
$ sudo -u hdfs hdfs dfs -ls /var/log/spark3/apps
```

The output lists log files related to corresponding Spark applications:

```
Found 3 items
-rwxrwx---   3 admin hadoop      57123 2023-08-24 13:32 /var/log/spark3/apps/application_1690315534352_0005
-rw-rw----   3 admin hadoop     126205 2023-08-24 14:52 /var/log/spark3/apps/application_1690315534352_0006
-rw-rw----   3 admin hadoop     235626 2023-08-25 09:14 /var/log/spark3/apps/application_1690315534352_0007
```

2. View a specific log with the following command.

```
$ sudo -u hdfs hdfs dfs -tail /var/log/spark3/apps/application_1690315534352_0007
```

The log contents looks similar to the following:

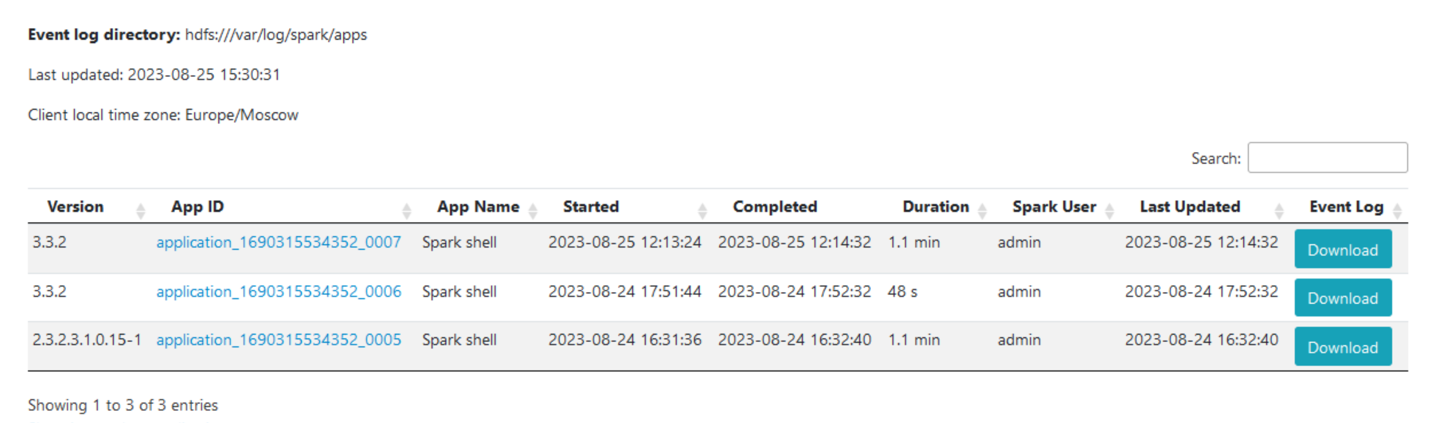
```
... ID":"2","Executor Info":{"Host":"ka-adh-2.ru-central1.internal","Total Cores":1,"Log Urls":{"stdout":"http://ka-adh-2.ru-
central1.internal:8042/node/containerlogs/container_e01_1690315534352_0006_01_000003/admin/stdout?
start=-4096","stderr":"http://ka-adh-2.ru-central1.
internal:8042/node/containerlogs/container_e01_1690315534352_0006_01_000003/admin/stderr?start=-4096"},"Attributes":
{"NM_HTTP_ADDRESS":"ka-adh-2.ru-central1.
internal:8042","USER":"admin","LOG_FILES":"stderr,stdout","NM_HTTP_PORT":"8042","CLUSTER_ID":"","NM_PORT":"8041","HTTP_SCHEME":"h
ttp://","NM_HOST":"ka-adh-2.ru-central1.internal","CONTAINER_ID":"container_e01_1690315534352_0006_01_000003"},"Resources":
{},"Resource Profile Id":0}}
{"Event":"SparkListenerBlockManagerAdded","Block Manager ID":{"Executor ID":"2","Host":"ka-adh-2.ru-central1.
internal","Port":46065},"Maximum Memory":384093388,"Timestamp":1692888721218,"Maximum Onheap Memory":384093388,"Maximum Offheap
Memory":0}
{"Event":"SparkListenerApplicationEnd","Timestamp":1692888752753}
```

History Server web UI

The Spark3 History Server component provides a web UI to visualize event logs of completed Spark applications. The web UI is available at *http://<history-host>:<history-port>* where:

- <history-host> — the address of the host where the Spark3 History Server component is installed.
- <history-port> — defaults to `18092`. Can be changed with the `spark.history.ui.port` property in ADCM.

The History Server web UI has the following view.



Event log directory: hdfs:///var/log/spark/apps

Last updated: 2023-08-25 15:30:31

Client local time zone: Europe/Moscow

Version	App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.3.2	application_1690315534352_0007	Spark shell	2023-08-25 12:13:24	2023-08-25 12:14:32	1.1 min	admin	2023-08-25 12:14:32	Download
3.3.2	application_1690315534352_0006	Spark shell	2023-08-24 17:51:44	2023-08-24 17:52:32	48 s	admin	2023-08-24 17:52:32	Download
2.3.2.3.1.0.15-1	application_1690315534352_0005	Spark shell	2023-08-24 16:31:36	2023-08-24 16:32:40	1.1 min	admin	2023-08-24 16:32:40	Download

Showing 1 to 3 of 3 entries
Show incomplete applications

Spark History Server interface

Spark3 History Server picks up event logs from a directory defined by the `spark.history.fs.logDirectory` property in ADCM (defaults to *hdfs:///var/log/spark/apps*). Typically, the properties `spark.history.fs.logDirectory` and `spark.eventLog.dir` should point to the same location to allow History Server to parse all events generated by Spark.

Logging configuration

Under the hood Spark uses the `Log4j` library for logging. Technically you can configure Spark logging by editing the *log4j.properties* file (you can create one based on */etc/spark3/conf.dist/log4j.properties.template*). However, editing *log4j.properties* is beyond the scope of this article. Instead, below is a list of ADCM configuration properties used to configure the Spark logging process.

Property	Description
spark.eventLog.enabled	Enables/disables Spark logging
spark.eventLog.dir	The base directory to store Spark logs. This parameter should point to the same directory as `spark.history.fs.logDirectory` to allow Spark3 History Server to load event logs generated by Spark
spark.history.fs.cleaner.enabled	Defines whether History Server should clean up event logs in the storage
spark.history.store.path	The path on the local file system to cache application history data
spark.history.ui.port	The port number for Spark History Server web UI
spark.history.fs.logDirectory	The directory with Spark event logs to load by History Server

Spark performance tuning

Konstantin Alpashkin

Contents

To Table of Contents

- Cache data in memory
- Use JOIN strategy hints
- Use partitioning hints
- Adaptive query execution
  - Coalesce post-shuffle partitions
  - Convert sort-merge JOIN to broadcast JOIN
  - Convert sort-merge JOIN to shuffled hash JOIN
  - Optimize skew JOIN

This section describes Spark performance optimization techniques and provides relevant Spark configuration properties used to fine-tune the optimization features.

> **NOTE**
>
> Throughout this section, "Spark" refers to the Spark3 ADH service.

Spark configuration properties mentioned in this section can be set using `sparkSession.setConf("property=value")` or via Spark SQL `SET key=value` commands. For details on configuration parameters used for performance tuning, see Spark documentation.

Cache data in memory

Spark SQL can cache tables in memory using a columnar format by calling `spark.catalog.cacheTable("tableName")` or `dataFrame.cache()`. After invoking these methods, Spark will scan only required columns and will automatically manage compression to minimize memory usage and GC load. To remove a table from memory, use `spark.catalog.uncacheTable("tableName")` or `dataFrame.unpersist()`.

Property	Description	Default value
spark.sql.inMemoryColumnarStorage.compressed	If set to `true`, Spark automatically selects a compression codec for each column based on data statistics	true
spark.sql.inMemoryColumnarStorage.batchSize	Controls the batch size for columnar caching. A larger batch size can improve memory utilization/compression at the cost of higher OOMs risk	10000
spark.sql.files.maxPartitionBytes	The maximum number of bytes to pack into a single partition when reading files. This property is effective only when using file-based data sources such as Parquet, ORC, and JSON	134217728 (128 MB)
spark.sql.files.openCostInBytes	Affects how many partitions the input data will be read into. The estimated "cost" to open a file, measured by the number of bytes that can be scanned at a time. This setting is useful when putting multiple files into a partition. It is better to over-estimate the cost, in this case the partitions with small files will be faster than partitions with bigger files (which is scheduled first). This property is effective only when using file-based data sources such as Parquet, ORC, and JSON	4194304 (4 MB)
spark.sql.files.minPartitionNum	The suggested (but not guaranteed) minimum number of split file partitions. If not set, the default value is taken from `spark.default.parallelism`. This property is effective only when using file-based data sources such as Parquet, ORC, and JSON	Value of `spark.default.parallelism`
spark.sql.broadcastTimeout	The timeout in seconds to complete a broadcast operation during broadcast JOINs	300
spark.sql.autoBroadcastJoinThreshold	The maximum size (in bytes) for a table to be broadcast to all worker nodes during a JOIN. Setting this value to `-1` disables the broadcasting	10485760 (10 MB)
spark.sql.shuffle.partitions	Sets the number of partitions to use when shuffling data for joins/aggregations	200
spark.sql.sources.parallelPartitionDiscovery.threshold	Sets a threshold to enable parallel listing for job input paths. If the number of input paths is larger than this threshold, Spark will list the files by using a Spark distributed job. Otherwise, it will fall back to sequential listing. This property is effective only when using file-based data sources such as Parquet, ORC, and JSON	32
spark.sql.sources.parallelPartitionDiscovery.parallelism	Sets the maximum listing parallelism for job input paths. If the number of input paths is larger than this value, it will be throttled down to this value. This property is effective only when using file-based data sources such as Parquet, ORC, and JSON	10000

Use JOIN strategy hints

Spark SQL JOINs are transformations that involve massive data shuffling over the network hence they may have performance issues when not designed with care. Using JOIN strategy hints you can instruct Spark to follow a specific JOIN strategy for a given table. You can greatly improve Spark JOIN performance by hinting a strategy that best fits the particular JOIN case. An example of joining 2 tables using the `BROADCAST` hint is below.

```
val joined_table = spark.table("src").join(spark.table("records").hint("broadcast"), "key")
```

If several hints are added for a table, Spark applies hints with the following priority:

1. BROADCAST
2. MERGE
3. SHUFFLE_HASH
4. SHUFFLE_REPLACE_NL

> **NOTE**
>
> There is no guarantee that Spark will actually use the hinted strategy since the strategy may not support all the JOIN types.

Use partitioning hints

In Spark SQL, partitioning hints allow you to suggest a specific partitioning strategy. The following partitioning hints are available:

- **COALESCE**. Used to control the number of output files. Accepts a partition number as a parameter.
- **REPARTITION**. Used to repartition to the specified number of partitions. Accepts a partition number, columns, or both/neither of them as parameters.
- **REPARTITION_BY_RANGE**. Accepts column names and optional partition number.
- **REBALANCE**. Used to re-balance the query results so that the partitions become of the reasonable size, without a significant skew.

The hints usage examples are below.

```
SELECT /*+ COALESCE(5) */ * FROM t
SELECT /*+ REPARTITION(5) */ * FROM t
SELECT /*+ REPARTITION(c) */ * FROM t
SELECT /*+ REPARTITION(5, c) */ * FROM t
SELECT /*+ REPARTITION */ * FROM t
SELECT /*+ REPARTITION_BY_RANGE(c) */ * FROM t
SELECT /*+ REPARTITION_BY_RANGE(5, c) */ * FROM t
SELECT /*+ REBALANCE */ * FROM t
SELECT /*+ REBALANCE(5) */ * FROM t
SELECT /*+ REBALANCE(c) */ * FROM t
SELECT /*+ REBALANCE(5, c) */ * FROM t
```

Adaptive query execution

Adaptive Query Execution (AQE) is an optimization technique that relies on the run-time statistics to choose the most efficient query execution plan. AQE is enabled by default and can be disabled using the `spark.sql.adaptive.enabled` property. The major AQE features are described below.

Coalesce post-shuffle partitions

This AQE feature coalesces the post-shuffle partitions based on the output statistics, so you do not need to manually set a shuffle partition number that would fit your dataset. Instead, Spark will pick a suitable shuffle partition number at run time (assuming you have set a large enough initial number of shuffle partitions via the `spark.sql.adaptive.coalescePartitions.initialPartitionNum` property).

This feature is activated by setting both `spark.sql.adaptive.enabled=true` and `spark.sql.adaptive.coalescePartitions.enabled=true`.

Property	Description	Default value
spark.sql.adaptive.coalescePartitions.enabled	If set to `true` and `spark.sql.adaptive.enabled=true`, Spark will coalesce contiguous shuffle partitions according to the target size (defined by `spark.sql.adaptive.advisoryPartitionSizeInBytes`) to avoid too many small tasks	true
spark.sql.adaptive.coalescePartitions.parallelismFirst	If set to `true`, Spark ignores the target size specified by `spark.sql.adaptive.advisoryPartitionSizeInBytes` (defaults to 64 MB) when coalescing contiguous shuffle partitions, and only respects the minimum partition size specified by `spark.sql.adaptive.coalescePartitions.minPartitionSize` (defaults to 1 MB) to maximize the parallelism. This allows avoiding performance regression with enabled AQE. It is recommended to set this property to `false` and respect the target size specified by `spark.sql.adaptive.advisoryPartitionSizeInBytes`	true
spark.sql.adaptive.coalescePartitions.minPartitionSize	The minimum size of shuffle partitions after coalescing. This value can be at most 20% of `spark.sql.adaptive.advisoryPartitionSizeInBytes`. This parameter is useful when the target size is ignored during partition coalescing, which is the default behavior	1 MB
spark.sql.adaptive.coalescePartitions.initialPartitionNum	The initial number of shuffle partitions before coalescing. If not set, the value is equal to `spark.sql.shuffle.partitions`. This property only has an effect if `spark.sql.adaptive.enabled=true` and `spark.sql.adaptive.coalescePartitions.enabled=true`	—
spark.sql.adaptive.advisoryPartitionSizeInBytes	The advisory size in bytes for shuffle partitions during adaptive optimization (when `spark.sql.adaptive.enabled=true`). This property has an effect when Spark coalesces small shuffle partitions or splits skewed shuffle partitions	64 MB

Convert sort-merge JOIN to broadcast JOIN

AQE automatically converts sort-merge JOIN to the broadcast hash JOIN when the run-time statistics of any JOINed side is smaller than the adaptive broadcast hash JOIN threshold (defined by `spark.sql.adaptive.autoBroadcastJoinThreshold`). This is not as efficient as using the broadcast hash JOIN initially, but it is still more efficient than sort-merge joining, as you can save sorting of both JOIN sides, and read shuffle files locally to reduce network traffic (assuming `spark.sql.adaptive.localShuffleReader.enabled=true`).

Property	Description	Default value
spark.sql.adaptive.autoBroadcastJoinThreshold	Sets the maximum size in bytes for a table to be broadcast to all worker nodes during a join. Setting this value to `-1` disables the broadcasting. The default value is taken from `spark.sql.autoBroadcastJoinThreshold`. Notice that this parameter is used only in AQE	—

Convert sort-merge JOIN to shuffled hash JOIN

AQE automatically converts sort-merge JOINs to shuffled hash JOINs when all post-shuffle partitions are smaller than the threshold defined by `spark.sql.adaptive.maxShuffledHashJoinLocalMapThreshold`.

Property	Description	Default value
spark.sql.adaptive.maxShuffledHashJoinLocalMapThreshold	Sets the maximum size in bytes per partition that can be allowed to build a local hash map. If this value is not smaller than `spark.sql.adaptive.advisoryPartitionSizeInBytes` and all the partitions do not exceed this value, Spark prefers to use the shuffled hash JOIN instead of sort merge JOIN regardless of the value of `spark.sql.join.preferSortMergeJoin`	0

Optimize skew JOIN

Data skew often leads to severe degradation of JOIN operations. This AQE feature dynamically handles a skew in sort-merge JOINs by splitting (and replicating if needed) skewed tasks into more or less evenly sized tasks. The feature is activated when both `spark.sql.adaptive.enabled=true` and `spark.sql.adaptive.skewJoin.enabled=true` are set.

Property	Description	Default value
spark.sql.adaptive.skewJoin.enabled	If set to `true` and `spark.sql.adaptive.enabled=true`, Spark dynamically handles the skew in a sort-merge JOIN by splitting (and replicating if needed) skewed partitions	true
spark.sql.adaptive.skewJoin.skewedPartitionFactor	A partition is considered skewed if its size (in bytes) is larger than this factor multiplied by the median partition size, and the partition is also larger than `spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes`	5
spark.sql.adaptive.skewJoin.skewedPartitionThresholdInBytes	A partition is considered skewed if its size (in bytes) is larger than this threshold and the partition is also larger than `spark.sql.adaptive.skewJoin.skewedPartitionFactor` multiplied by the median partition size. This property should be larger than `spark.sql.adaptive.advisoryPartitionSizeInBytes`	256MB

Work with Iceberg tables in Spark

Konstantin Alpashkin

Contents

To Table of Contents

- Iceberg catalogs
  - Catalog backend types
  - SparkCatalog vs SparkSessionCatalog
  - Catalog configuration properties
- DDL commands
  - Create an Iceberg table
  - Create a partitioned table
  - CREATE TABLE using CTAS/RTAS
  - DROP TABLE
  - ALTER TABLE
- Query data
  - Time travel reads
  - Metadata tables
- Write data
  - Branch writes
- Roll back

Apache Iceberg is an open, high-performance format for large analytic tables. The ADH Spark3 service adopts this format allowing you to work with Iceberg tables through Spark.

> **NOTE**
>
> Spark3 supports Iceberg tables starting ADH 3.2.4.3.

The support for Iceberg tables is enabled by default in the Spark3 ADH service. In this article, the Spark SQL module is used to demonstrate the examples of working with Iceberg tables through Spark. The Spark SQL module allows running traditional ANSI SQL commands on Spark DataFrame objects.

You can run Spark SQL queries in the *spark3-sql* shell. For this, run the command on a host where the Spark3 Client ADH component is installed.

```
$ spark3-sql
```

In this article, all SQL examples are designed for *spark3-sql* shell.

Iceberg catalogs

The Iceberg architecture introduces the concept of catalogs, which allow managing and loading Iceberg tables by name. An Iceberg catalog is a named repository of tables and their metadata, which are organized into namespaces. Catalog names and namespaces are used in Spark SQL queries to qualify tables that belong to different catalogs, similarly to `SELECT … FROM <database_name>.<table_name>` notation. The following syntax is used to fully qualify a specific Iceberg table.

```
SELECT ... FROM <catalog_name>.<namespace>.<table_name> ...
```

Catalog backend types

Iceberg offers several catalog backends to track tables, such as REST, Hive, JDBC, etc. An example of creating a Hive-based Iceberg catalog that loads tables from a Hive Metastore is shown below:

```
$ spark3-sql \
    --conf spark.sql.catalog.test_catalog_hive=org.apache.iceberg.spark.SparkCatalog \
    --conf spark.sql.catalog.test_catalog_hive.type=hive
```

To load tables from the catalog created above, you should prefix the table names with `test_catalog_hive`.

> **NOTE**
>
> Catalog configuration properties specified using `--conf` notation in the example above can also be set in ADCM (***Clusters → <clusterName> → Services → Spark3 → Primary Configuration → Custom spark-defaults.conf***).

In Spark3, the default catalog is named `spark-catalog`. This catalog supports single-level namespaces that correspond to Hive database names. Throughout this article, the examples use the catalog named `spark-catalog` and the `default` namespace.

SparkCatalog vs SparkSessionCatalog

Iceberg provides two catalog implementations for Spark:

- `org.apache.iceberg.spark.SparkCatalog`. Allows working only with Iceberg tables using Hive Metastore or an HDFS warehouse directory.
- `org.apache.iceberg.spark.SparkSessionCatalog`. Allows working with both Iceberg and non-Iceberg tables. When Spark attempts to access a non-Iceberg table, `SparkSessionCatalog` delegates interaction to the built-in Spark catalog implementation. In the Spark3 service, this catalog implementation is used by default.

Catalog configuration properties

Iceberg catalogs can be configured using `spark.sql.catalog.<catalog-name>.*` properties. Common configuration properties for Hive and Hadoop are presented in the table below.

Property	Description
spark.sql.catalog.<catalog-name>.type	Defines the underlying Iceberg catalog implementation. The following values are supported: - `hive` (HiveCatalog); - `hadoop` (HadoopCatalog); - `rest` (RESTCatalog); - `glue` (GlueCatalog); - `jdbc` (JdbcCatalog); - `nessie` (NessieCatalog); - empty value if a custom catalog implementation is used.
spark.sql.catalog.<catalog-name>.catalog-impl	A custom Iceberg catalog implementation. If `spark.sql.catalog.<catalog-name>.type` is null, `spark.sql.catalog.<catalog-name>.catalog-impl` must not be null
spark.sql.catalog.<catalog-name>.io-impl	A custom FileIO implementation
spark.sql.catalog.<catalog-name>.metrics-reporter-impl	A custom MetricsReporter implementation
spark.sql.catalog.<catalog-name>.default-namespace	The default current namespace for the catalog. The default value is `default`
spark.sql.catalog.<catalog-name>.uri	Hive Metastore URL (*thrift://host:port*) for Hive-based catalogs or REST URL for REST-based catalogs
spark.sql.catalog.<catalog-name>.warehouse	The base path to the warehouse directory. For example, *hdfs://nn:8020/warehouse/path*
spark.sql.catalog.<catalog-name>.cache-enabled	Defines whether to enable catalog caching. Defaults to `true`
spark.sql.catalog.<catalog-name>.cache.expiration-interval-ms	Sets a timeout after which cached catalog entries are expired. This parameter is effective only if `spark.sql.catalog.<catalog-name>.cache-enabled=true`. Setting `-1` disables cache expiration and setting `0` disables caching entirely, regardless of the `cache-enabled` value. Defaults to `30000` (30 seconds)
spark.sql.catalog.<catalog-name>.table-default.propertyKey	Sets a default value for an Iceberg table property with `propertyKey`. This value will be set for all tables created by the current catalog unless overridden
spark.sql.catalog.<catalog-name>.table-override.propertyKey	Sets a value for an Iceberg table property with `propertyKey` that cannot be overridden

For more information on creating and configuring Iceberg catalogs, see Iceberg documentation.

DDL commands

Create an Iceberg table

To create an Iceberg table, append the `USING iceberg` clause to the standard `CREATE TABLE` statement.

Example:

```
CREATE TABLE spark_catalog.default.transactions(
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
USING iceberg;
```

Run the `DESCRIBE FORMATTED` command to view information about the newly created table:

```
DESCRIBE FORMATTED spark_catalog.default.transactions;
```

Sample output:

```
txn_id                  int
acc_id                  int
txn_value               double
txn_date                date

# Metadata Columns
_spec_id                int
_partition              struct<>
_file                   string
_pos                    bigint
_deleted                boolean

# Detailed Table Information
Name                    spark_catalog.default.transactions
Type                    MANAGED
Location                hdfs://adh/apps/hive/warehouse/transactions
Provider                iceberg
Owner                   hdfs
Table Properties        [current-snapshot-id=none,format=iceberg/parquet,format-version=2,write.parquet.compression-codec=zstd]
```

The `Provider` value in the output indicates that the table has been created as an Iceberg table.

Create a partitioned table

To create a partitioned table, use the syntax as shown below:

```
CREATE TABLE spark_catalog.default.transactions_partitioned (
    txn_id bigint,
    acc_id int,
    txn_value double,
    txn_date date)
USING iceberg
PARTITIONED BY (acc_id);
```

You can also use transform expressions in the `PARTITIONED BY` clause to create hidden partitions instead of specifying partitions explicitly. An example is below.

```
CREATE TABLE spark_catalog.default.transactions_partitioned_transform (
    id bigint,
    acc_id int,
    txn_value double,
    txn_date timestamp)
USING iceberg
PARTITIONED BY (day(txn_date)); ①
```

① `day(txn_date)` is a transformation expression to create partitions by day.

To verify the partitioning, insert some data into the test table as shown below:

```
INSERT INTO spark_catalog.default.transactions_partitioned_transform VALUES
(1, 1002, 10.00, cast('2023-01-01' as timestamp)),
(2, 1001, 20.00, cast('2023-01-02' as timestamp));
```

Then, check the warehouse directory contents in HDFS:

```
$ hdfs dfs -ls -R /apps/hive/warehouse/transactions_partitioned_transform
```

Sample output:

```
drwxrwxr-x   - hdfs hadoop          0 2024-06-19 16:06 /apps/hive/warehouse/transactions_partitioned_transform/data
drwxrwxr-x   - hdfs hadoop          0 2024-06-19 16:06
/apps/hive/warehouse/transactions_partitioned_transform/data/txn_date_day=2023-01-01
-rw-r--r--   3 hdfs hadoop       1206 2024-06-19 16:06
/apps/hive/warehouse/transactions_partitioned_transform/data/txn_date_day=2023-01-01/00000-12-a029426f-ca10-4efc-b87c-
389694c218bc-0-00001.parquet
drwxrwxr-x   - hdfs hadoop          0 2024-06-19 16:06
/apps/hive/warehouse/transactions_partitioned_transform/data/txn_date_day=2023-01-02
-rw-r--r--   3 hdfs hadoop       1206 2024-06-19 16:06
/apps/hive/warehouse/transactions_partitioned_transform/data/txn_date_day=2023-01-02/00000-12-a029426f-ca10-4efc-b87c-
389694c218bc-0-00002.parquet
drwxrwxr-x   - hdfs hadoop          0 2024-06-19 16:06 /apps/hive/warehouse/transactions_partitioned_transform/metadata
-rw-r--r--   3 hdfs hadoop       1381 2024-06-19 16:06 /apps/hive/warehouse/transactions_partitioned_transform/metadata/00000-
28c5f350-f14a-4fcb-b509-10a327fa19cb.metadata.json
-rw-r--r--   3 hdfs hadoop       2519 2024-06-19 16:06 /apps/hive/warehouse/transactions_partitioned_transform/metadata/00001-
7c83b7d2-0a43-463e-b180-6589a0f696af.metadata.json
-rw-r--r--   3 hdfs hadoop       7157 2024-06-19 16:06 /apps/hive/warehouse/transactions_partitioned_transform/metadata/28884384-
0539-4ca5-8dcb-95d00de4419e-m0.avro
-rw-r--r--   3 hdfs hadoop       4264 2024-06-19 16:06 /apps/hive/warehouse/transactions_partitioned_transform/metadata/snap-
2743337418634934372-1-28884384-0539-4ca5-8dcb-95d00de4419e.avro
```

According to the sample output, the test data has been distributed among several HDFS directories (partitions) named *txn_date_day={date}*, each representing a distinct day.

CREATE TABLE using CTAS/RTAS

You can use the `CREATE TABLE AS <tbl_name>` (CTAS) and `REPLACE TABLE AS <tbl_name>` (RTAS) syntax to create/replace an Iceberg table based on an existing table. The examples are below.

```
CREATE TABLE spark_catalog.default.transactions_ctas USING iceberg
AS SELECT txn_id, txn_value
FROM spark_catalog.default.transactions;
```

```
REPLACE TABLE spark_catalog.default.transactions_rtas USING iceberg
AS SELECT txn_id, txn_value
FROM spark_catalog.default.transactions
WHERE txn_value < 50;
```

The table creation/replacement operation using CTAS/RTAS syntax is atomic only when using SparkCatalog. With SparkSessionCatalog implementation (used by default in Spark3), the atomicity of the CTAS/RTAS operations is not guaranteed.

DROP TABLE

To delete an Iceberg table, use the standard syntax:

```
DROP TABLE <table_name> [PURGE]
```

The optional `PURGE` flag defines whether the table's data/metadata files should also be deleted from the storage (HDFS, Ozone, and alike). If `PURGE` is not set, only the table metadata is deleted from the catalog.

By default, the `DROP TABLE` command does not remove empty table directories from the storage (even with the `PURGE` flag set). For example:

```
hdfs dfs -ls -R /apps/hive/warehouse/my_test_table
drwxrwxr-x   - hdfs hadoop          0 2025-05-29 15:22 /apps/hive/warehouse/my_test_table/data
drwxrwxr-x   - hdfs hadoop          0 2025-05-29 15:22 /apps/hive/warehouse/my_test_table/metadata
```

If you need the `DROP TABLE` command to delete stale directories from the storage, use the following properties when creating a table:

```
CREATE TABLE <table_name>(id int) USING ICEBERG
TBLPROPERTIES (
    'external.table.purge'='TRUE' ①
    'drop.base-directory.enabled'='TRUE', ②
 );
```

① Forces `DROP TABLE` to delete data/metadata files (equivalent to `PURGE`).

② Forces `DROP TABLE` to delete empty table directories from the storage (HDFS, Ozone, and so on).

ALTER TABLE

The Spark3 service supports rich `ALTER TABLE` capabilities for Iceberg tables. Below you can find common `ALTER TABLE` operations with examples. More details about Iceberg `ALTER TABLE` commands can be found in Iceberg documentation.

ALTER TABLE … RENAME TO

The following command renames an Iceberg table. The command affects only the table's metadata, keeping the table data intact.

```
ALTER TABLE spark_catalog.default.transactions
RENAME TO spark_catalog.default.transactions_new;
```

ALTER TABLE … SET TBLPROPERTIES

The following example sets the write format to ORC using the Iceberg table properties.

```
ALTER TABLE spark_catalog.default.transactions
SET TBLPROPERTIES ('write.format.default'='orc');
```

> **TIP**
>
> To clear a table property, use the `UNSET` command.

ALTER TABLE … ADD COLUMN(S)

You can add a column to an existing Iceberg table by using the `ADD COLUMN` clause. Example:

```
ALTER TABLE spark_catalog.default.transactions
ADD COLUMN (
    description string comment "A transaction's description"
);
```

The `DESCRIBE` command output:

```
DESCRIBE spark_catalog.default.transactions;
txn_id                  int
acc_id                  int
txn_value               double
txn_date                date
description             string                  A transaction's description
```

To add multiple columns with one command, use the `ADD COLUMNS` clause and separate the columns by a comma. Example:

```
ALTER TABLE spark_catalog.default.transactions
ADD COLUMNS (
    is_committed boolean,
    is_acknowledged boolean
);
```

The `DESCRIBE` command output:

```
DESCRIBE spark_catalog.default.transactions;
txn_id                  int
acc_id                  int
txn_value               double
txn_date                date
description             string                  A transaction's description
is_committed            boolean
is_acknowledged         boolean
```

To add a column with a struct data type, such as `struct<x, y>`, use the following syntax:

```
ALTER TABLE spark_catalog.default.transactions
ADD COLUMN processor_metadata struct<x: double, y: double>;
```

For more details on working with composite data types, like arrays, maps, etc., see Iceberg documentation.

ALTER TABLE … ALTER COLUMN

You can change the definition of a column; for example, change the column data type, make the column nullable, add a comment, etc. Iceberg allows changing the column data type if the update is safe, for example:

- `int` → `bigint`
- `float` → `double`
- `decimal(i,j)` → `decimal(i1,j)` assuming `i1` > `i`

```
ALTER TABLE spark_catalog.default.transactions
ALTER COLUMN txn_id TYPE bigint;
```

The `DESCRIBE` command output:

```
DESCRIBE spark_catalog.default.transactions;
txn_id                  bigint
acc_id                  int
txn_value               double
txn_date                date
description             string                  A transaction's description
is_committed            boolean
is_acknowledged         boolean
processor_metadata      struct<x:double,y:double>
```

To make a field nullable, use the syntax as shown below.

```
ALTER TABLE spark_catalog.default.transactions
ALTER COLUMN acc_id DROP NOT NULL;
```

However, you cannot change a nullable column to a non-nullable since Iceberg does not know whether there are existing entries with `NULL` values.

ALTER TABLE … ADD PARTITION FIELD

Iceberg supports partitioning scheme evolution, allowing you to modify partition fields in existing tables. The modification of a partitioning scheme is a metadata operation that does not affect the existing table data. New data will be written as per the new partitioning spec, however, the existing data will adhere to the old partitioning layout. Old data files will have `NULL` values for the new partition fields in the metadata tables. Example:

```
ALTER TABLE spark_catalog.default.transactions_partitioned
ADD PARTITION FIELD txn_date;
```

Run the `DESCRIBE` command to ensure the partitioning spec has been updated. The output:

```
DESCRIBE spark_catalog.default.transactions_partitioned;
txn_id                  bigint
acc_id                  int
txn_value               decimal(10,2)
txn_date                date
# Partition Information
# col_name              data_type               comment
acc_id                  int
txn_date                date
```

> **CAUTION**
>
> Changing the partitioning scheme of an Iceberg table should always be done with care. Some changes may lead to unexpected behavior and can result in corrupted table metadata. For more details, see ADD, DROP, and REPLACE partition field commands.

ALTER TABLE … WRITE ORDERED BY

You can specify the sorting order for an Iceberg table that is used to automatically sort data written to that table. For example, the Spark's MERGE INTO operation uses table ordering.

```
ALTER TABLE spark_catalog.default.transactions
WRITE ORDERED BY txn_id;
```

> **NOTE**
>
> The table write order does not guarantee data order when querying data. It only affects how data is written to the table.

To unset the sorting order for a table, use the `UNORDERED` keyword as shown in the example below.

```
ALTER TABLE spark_catalog.default.transactions
WRITE UNORDERED;
```

ALTER TABLE … CREATE BRANCH

You can create branches for an Iceberg table by using the `CREATE BRANCH` statement. Consider the examples below.

```
ALTER TABLE spark_catalog.default.transactions CREATE BRANCH `test-branch`; ①
ALTER TABLE spark_catalog.default.transactions CREATE BRANCH IF NOT EXISTS `test-branch` RETAIN 7 DAYS; ②
ALTER TABLE spark_catalog.default.transactions CREATE OR REPLACE BRANCH `test-branch`; ③
ALTER TABLE spark_catalog.default.transactions CREATE BRANCH `test-branch` AS OF VERSION 123 ④
```

① Creates a branch with the default retention policy.

② Creates a branch if it does not exist and sets the retention time to 7 days.

③ Creates or replaces the branch if it already exists.

④ Creates a branch at the `123` snapshot using the default retention. The `123` snapshot must be available when running this command.

To replace and drop branches, use the REPLACE BRANCH and DROP BRANCH commands.

ALTER TABLE … CREATE TAG

You can create Iceberg tags to "label" the state of a table and return to this table state in the future. Consider the examples below.

```
ALTER TABLE spark_catalog.default.transactions CREATE TAG `test-tag`; ①
ALTER TABLE spark_catalog.default.transactions CREATE TAG IF NOT EXISTS `test-tag` RETAIN 365 DAYS; ②
ALTER TABLE spark_catalog.default.transactions CREATE OR REPLACE TAG `test-tag`; ③
ALTER TABLE spark_catalog.default.transactions CREATE TAG `test-tag` AS OF VERSION 123 ④
```

① Creates a tag with the default retention policy.

② Creates a tag if it does not exist at the current snapshot and sets the retention time to 365 days.

③ Creates or replaces the tag if it already exists.

④ Creates a tag at the `123` snapshot using the default retention. The `123` snapshot must be pre-created when running this command.

Querying the table data using the tag created in the above example looks as follows:

```
SELECT * FROM spark_catalog.default.transactions
VERSION AS OF 'test-tag';
```

To replace and drop tags, use the REPLACE TAG and DROP TAG commands.

Query data

To query data from an Iceberg table using Spark SQL, you have to provide a catalog name, namespace, and table name.

Example:

```
SELECT * FROM spark_catalog.default.transactions;
```

Where:

- `spark_catalog` is the default catalog name.
- `default` is the default namespace.
- `transactions` is the requested table name.

To load a table as a DataFrame, use the `table()` method. For example (PySpark):

```
df = spark.table("spark_catalog.default.transactions")
```

Time travel reads

Iceberg tables support the **time travel** feature that allows you to query data from a specific table snapshot that was created at some point in the past and is identified by an ID or timestamp.

> **TIP**
>
> The information about all the snapshots available for a given table is stored in the `snapshots` metadata table.

The following scenario shows how to run time-travel queries through Spark.

1. Create a test table using *spark3-sql*:

```
CREATE TABLE spark_catalog.default.transactions_tt(
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
USING iceberg;
```

2. Insert some data into the test table:

```
INSERT INTO spark_catalog.default.transactions_tt VALUES
(1, 1002, 100.00, cast('2023-01-01' as date)),
(2, 1001, 50.00, cast('2023-01-02' as date));
```

3. Inspect the `snapshot` metadata table:

```
SELECT * FROM spark_catalog.default.transactions_tt.snapshots;
```

This produces an output similar to the following:

```
2024-06-27 14:54:05.664 6216997363030961663     NULL    append  hdfs://adh/apps/hive/warehouse/transactions_tt/metadata/snap-
6216997363030961663-1-f55143c4-0351-4f6a-8978-f0c2fa5666b7.avro    {"added-data-files":"2","added-files-size":"2254","added-
records":"2","changed-partition-count":"1","spark.app.id":"application_1719409181169_0020","total-data-files":"2","total-delete-
files":"0","total-equality-deletes":"0","total-files-size":"2254","total-position-deletes":"0","total-records":"2"}
```

The output indicates that currently there is one snapshot (`6216997363030961663`) available for the table. This snapshot has been created right after the `INSERT` operation.

4. Add more test data to the table:

```
INSERT INTO spark_catalog.default.transactions_tt VALUES
(3, 1003, 150.00, cast('2024-01-03' as date));
```

5. Query the `snapshot` metadata table again:

```
SELECT * FROM spark_catalog.default.transactions_tt.snapshots;
```

The result:

```
2024-06-27 14:54:05.664 6216997363030961663     NULL    append  hdfs://adh/apps/hive/warehouse/transactions_tt/metadata/snap-
6216997363030961663-1-f55143c4-0351-4f6a-8978-f0c2fa5666b7.avro    {"added-data-files":"2","added-files-size":"2254","added-
records":"2","changed-partition-count":"1","spark.app.id":"application_1719409181169_0020","total-data-files":"2","total-delete-
files":"0","total-equality-deletes":"0","total-files-size":"2254","total-position-deletes":"0","total-records":"2"}
2024-06-27 14:56:20.781 3775233323500475562     6216997363030961663     append
hdfs://adh/apps/hive/warehouse/transactions_tt/metadata/snap-3775233323500475562-1-6500b975-c67e-4171-8c5d-70623933abc3.avro
{"added-data-files":"1","added-files-size":"1126","added-records":"1","changed-partition-count":"1","spark.app.
id":"application_1719409181169_0020","total-data-files":"3","total-delete-files":"0","total-equality-deletes":"0","total-files-
size":"3380","total-position-deletes":"0","total-records":"3"}
```

Now the output indicates one more snapshot — the one that was created by the second `INSERT` operation.

6. Run the time-travel query using the ID or the timestamp of the older snapshot. You can run time travel queries in SQL using the `TIMESTAMP AS OF <timestamp>` or `VERSION AS OF <snapshot-id>` clauses. The `VERSION AS OF` clause can accept a snapshot ID, branch name, or tag name.
Example:

```
SELECT * FROM spark_catalog.default.transactions_tt VERSION AS OF 6216997363030961663; ①
SELECT * FROM spark_catalog.default.transactions_tt TIMESTAMP AS OF '2024-06-27 14:55:20.780'; ②
```

① Selects data using the ID of the older snapshot.

② Selects data using a timestamp. In this example, the timestamp value is chosen intentionally to pick data from the older snapshot.

The output:

```
1       1002    100.0   2023-01-01
2       1001    50.0    2023-01-02
```

The result set contains only the data that was present in the table right after the first `INSERT` operation.

7. Run an ordinary `SELECT` query:

```
SELECT * FROM spark_catalog.default.transactions_tt;
```

The output:

```
1       1002    100.0   2023-01-01
2       1001    50.0    2023-01-02
3       1003    150.0   2024-01-03
```

Unlike the previous time-travel queries, this result set contains all the data related to both `INSERT` operations.

Metadata tables

Iceberg maintains metadata tables, like `history`, `files`, `snapshots`, and so on. These tables store valuable information about snapshots, files, operations, etc., that can be useful for time-travel, branching, and tag-related queries. You can inspect metadata tables by adding the metadata table name after the original table name.

Example:

```
SELECT * FROM spark_catalog.default.transactions.snapshots;
```

Sample output:

```
2024-06-21 14:30:36.001 7505187482383765765     NULL    append  hdfs://adh/apps/hive/warehouse/transactions/metadata/snap-
7505187482383765765-1-be783958-0d56-4b2c-b802-54cdae7a349d.avro       {"changed-partition-count":"0","total-data-
files":"0","total-delete-files":"0","total-equality-deletes":"0","total-files-size":"0","total-position-deletes":"0","total-
records":"0"}
2024-06-21 16:09:08.645 5529064852849657419     NULL    append  hdfs://adh/apps/hive/warehouse/transactions/metadata/snap-
5529064852849657419-1-af755165-d1e2-476e-85a9-40ea179c2e9f.avro       {"added-data-files":"2","added-files-size":"2522","added-
records":"9","changed-partition-count":"1","spark.app.id":"application_1719409181169_0020","total-data-files":"2","total-delete-
files":"0","total-equality-deletes":"0","total-files-size":"2522","total-position-deletes":"0","total-records":"9"}
```

Write data

In Spark3, you can write data to Iceberg tables using traditional SQL syntax like INSERT INTO, INSERT OVERWRITE, UPDATE, and so on. Also, Iceberg supports row-level operations like `MERGE INTO` and `DELETE FROM`, which are demonstrated below. When writing to an Iceberg table, Spark data types are converted to Iceberg types according to the Spark types conversion table. For more details on Iceberg write operations, see Iceberg documentation.

MERGE INTO

`MERGE INTO` is a row-level update operation that modifies a target table using a set of updates from a source table. The row update condition is defined by the `ON` clause that is used similarly to a `JOIN` condition.

The `MERGE INTO` command syntax is shown below:

```
MERGE INTO <target_table>
   USING <source_table>
   ON <merge_condition>
   WHEN MATCHED <matched_condition> THEN <matched_action> |
   WHEN NOT MATCHED THEN <not_matched_action>
```

The updates to rows in the target table are defined using the `WHEN [NOT] MATCHED <matched_condition> THEN <matched_action>`. Target rows can be updated and deleted, and the source rows that do not match can be inserted. Consider the following example that demonstrates `MERGE INTO` usage.

Assume there are two Iceberg tables with the following contents:

spark_catalog.default.transactions

```
txn_id  acc_id  txn_value txn_date
1       1001    25.0      2023-02-03
2       1001    50.0      2023-03-03
3       1002    10.0      2023-04-01
```

spark_catalog.default.transactions_copy

```
txn_id  acc_id  txn_value txn_date
1       1003    20.0      2023-02-03
3       1001    50.0      2023-03-03
5       1002    100.0     2023-04-02
```

The following `MERGE INTO` command deletes all rows in the `spark_catalog.default.transactions_copy` table that have a match in the `spark_catalog.default.transactions` table by the `txn_id` field.

```
MERGE INTO spark_catalog.default.transactions_copy USING spark_catalog.default.transactions
  ON spark_catalog.default.transactions_copy.txn_id = spark_catalog.default.transactions.txn_id
  WHEN MATCHED THEN DELETE;
```

After running this command, the tables contents becomes as follows.

spark_catalog.default.transactions

```
txn_id  acc_id  txn_value txn_date
1       1001    25.0      2023-02-03
2       1001    50.0      2023-03-03
3       1002    10.0      2023-04-01
```

spark_catalog.default.transactions_copy

```
txn_id  acc_id  txn_value txn_date
5       1002    100.0     2023-04-02
```

The following `MERGE INTO` command updates all rows in the `spark_catalog.default.transactions` table whose `acc_id` field matches `acc_id` in the `spark_catalog.default.transactions_copy` table. For rows with matched `acc_id` values, the query sets `txn_value=0`.

```
MERGE INTO spark_catalog.default.transactions USING spark_catalog.default.transactions_copy
  ON spark_catalog.default.transactions.acc_id = spark_catalog.default.transactions_copy.acc_id
  WHEN MATCHED THEN UPDATE SET txn_value=0;
```

After running this command, the tables contents becomes as follows.

spark_catalog.default.transactions

```
txn_id  acc_id  txn_value txn_date
1       1001    25.0      2023-02-03
2       1001    50.0      2023-03-03
3       1002    10.0      2023-04-01
```

spark_catalog.default.transactions_copy

```
txn_id  acc_id  txn_value txn_date
5       1002    0.0       2023-04-02
```

DELETE FROM

`DELETE FROM` is a row-level operation to remove data from Iceberg tables. Delete queries accept a filter to match rows to delete, as shown below. If a delete filter matches the entire table partition, Iceberg will only modify the table's metadata. If the filter matches individual rows, Iceberg will rewrite only the affected data files.

Examples:

```
DELETE FROM spark_catalog.default.transactions
WHERE txn_date > '2023-03-03';
```

```
DELETE FROM spark_catalog.default.transactions
WHERE txn_value < (SELECT min(txn_value) FROM spark_catalog.default.transactions_copy);
```

Branch writes

To write data to a specific Iceberg table's branch, you have to specify a branch identifier in your query.

> **NOTE**
>
> The target branch must be created before performing a write.

For example, to write data to the branch named `testbranch`, use the following syntax:

```
INSERT INTO spark_catalog.default.transactions.branch_testbranch VALUES
(1, 1003, 20.00, cast('2023-02-03' as date));
```

If a branch name includes non-alphanumeric characters (e.g. `test-branch`), the branch name must be back-quoted as shown in the example below.

```
INSERT INTO spark_catalog.default.transactions.`branch_test-branch` VALUES
(1, 1003, 20.00, cast('2023-02-03' as date));
```

To read data from the `testbranch` created above, use the syntax:

```
SELECT * FROM spark_catalog.default.transactions
VERSION AS OF 'testbranch';
```

Apart from writing data to a branch using `INSERT`, you can also use `UPDATE`, `DELETE`, and `MERGE INTO` operations.

Examples:

```
UPDATE spark_catalog.default.transactions.branch_testbranch
SET acc_id = 1005
WHERE acc_id = 1003
```

```
DELETE FROM spark_catalog.default.transactions.branch_testbranch
WHERE txn_id = 1;
```

```
MERGE INTO spark_catalog.default.transactions.branch_testbranch USING spark_catalog.default.transactions_copy
  ON spark_catalog.default.transactions.branch_testbranch.txn_id = spark_catalog.default.transactions_copy.txn_id
  WHEN MATCHED THEN DELETE;
```

Roll back

Iceberg supports the rollback function that allows rolling a table back to an older snapshot. This feature can be useful, for example, for undoing incidental or mistaken table updates. Table data can be rolled back to the desired state as long as the target snapshot is not expired. Consider the following use case that demonstrates the rollback feature usage.

1. Create a test Iceberg table with the following contents:

```
txn_id  acc_id  txn_value txn_date
1       1002    10.0      2024-01-01
2       1001    20.0      2024-01-02
```

For this, execute the following queries in *spark3-sql*:

```
CREATE TABLE spark_catalog.default.transactions_rollback_demo (
    txn_id int,
    acc_id int,
    txn_value double,
    txn_date date)
USING iceberg;
```

```
INSERT INTO spark_catalog.default.transactions_rollback_demo VALUES
(1, 1002, 10.00, cast('2024-01-01' as timestamp)),
(2, 1001, 20.00, cast('2024-01-02' as timestamp));
```

2. Delete a single row from the test table. This operation will be rolled back on further steps.

```
DELETE FROM spark_catalog.default.transactions_rollback_demo
WHERE txn_id = 2;
```

3. Get information about the snapshots currently available for the test table. For this, query the metadata database as shown below:

```
SELECT * FROM spark_catalog.default.transactions_rollback_demo.snapshots;
```

Sample output:

```
2024-07-17 07:59:56.31  6142059163774284478     NULL    append
hdfs://adh/apps/hive/warehouse/transactions_rollback_demo/metadata/snap-6142059163774284478-1-c2c3b442-648b-4e94-8b14-
594ec65f66c5.avro{"added-data-files":"2","added-files-size":"2254","added-records":"2","changed-partition-count":"1","spark.app.
id":"application_1720628013974_0014","total-data-files":"2","total-delete-files":"0","total-equality-deletes":"0","total-files-
size":"2254","total-position-deletes":"0","total-records":"2"}
2024-07-17 08:02:55.859 8329113954968361656     6142059163774284478     delete
hdfs://adh/apps/hive/warehouse/transactions_rollback_demo/metadata/snap-8329113954968361656-1-375044c0-90fa-44f5-a6c6-
72ebc12cc473.avro {"changed-partition-count":"1","deleted-data-files":"1","deleted-records":"1","removed-files-
size":"1127","spark.app.id":"application_1720628013974_0015","total-data-files":"1","total-delete-files":"0","total-equality-
deletes":"0","total-files-size":"1127","total-position-deletes":"0","total-records":"1"}
```

The output indicates that there are two snapshots available for the test table. The first snapshot was created right after inserting data, and the second — after the delete operation.

4. Roll back the table contents to a state before the delete operation:

```
CALL spark_catalog.system.rollback_to_snapshot('spark_catalog.default.transactions_rollback_demo', <snapshot-id>);
```

Where `<snapshot-id>` is the ID of the snapshot created before the delete request.

> **TIP**
>
> Alternatively, you can provide a timestamp to roll back to the required snapshot.

5. Select data from the test table to verify the rollback results.

```
SELECT * FROM spark_catalog.default.transactions_rollback_demo;
1       1002    10.0    2023-01-01
2       1001    20.0    2023-01-02
```

The output indicates that the deleted row has been restored.

Spark3 service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Spark3 service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Spark3 service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Spark3 components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM
Manage Ranger plugin	Enables/disables the Spark3 Ranger plugin
Switch dynamic allocation	Enables/disables dynamic resource allocation for Spark3

Spark3 components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

SSM UI overview

Konstantin Alpashkin, Elena Kostyuchenko

Contents

To Table of Contents

Cluster Info
Rules
Actions
Audit

SSM provides a web interface (AD.SSM) where you can manage SSM rules, run SSM actions, monitor storage state, view action execution logs, and so on.

By default, the web UI is available at *http://<host>:7045/* where `<host>` is the ADH host with the SSM Server component installed. You can change the port number (`7045` by default) using the `smart.server.http.address` configuration property.

SSM UI includes the following tabs:

- Cluster Info
- Rules
- Actions
- Audit

At the bottom of the left panel, there are three options:

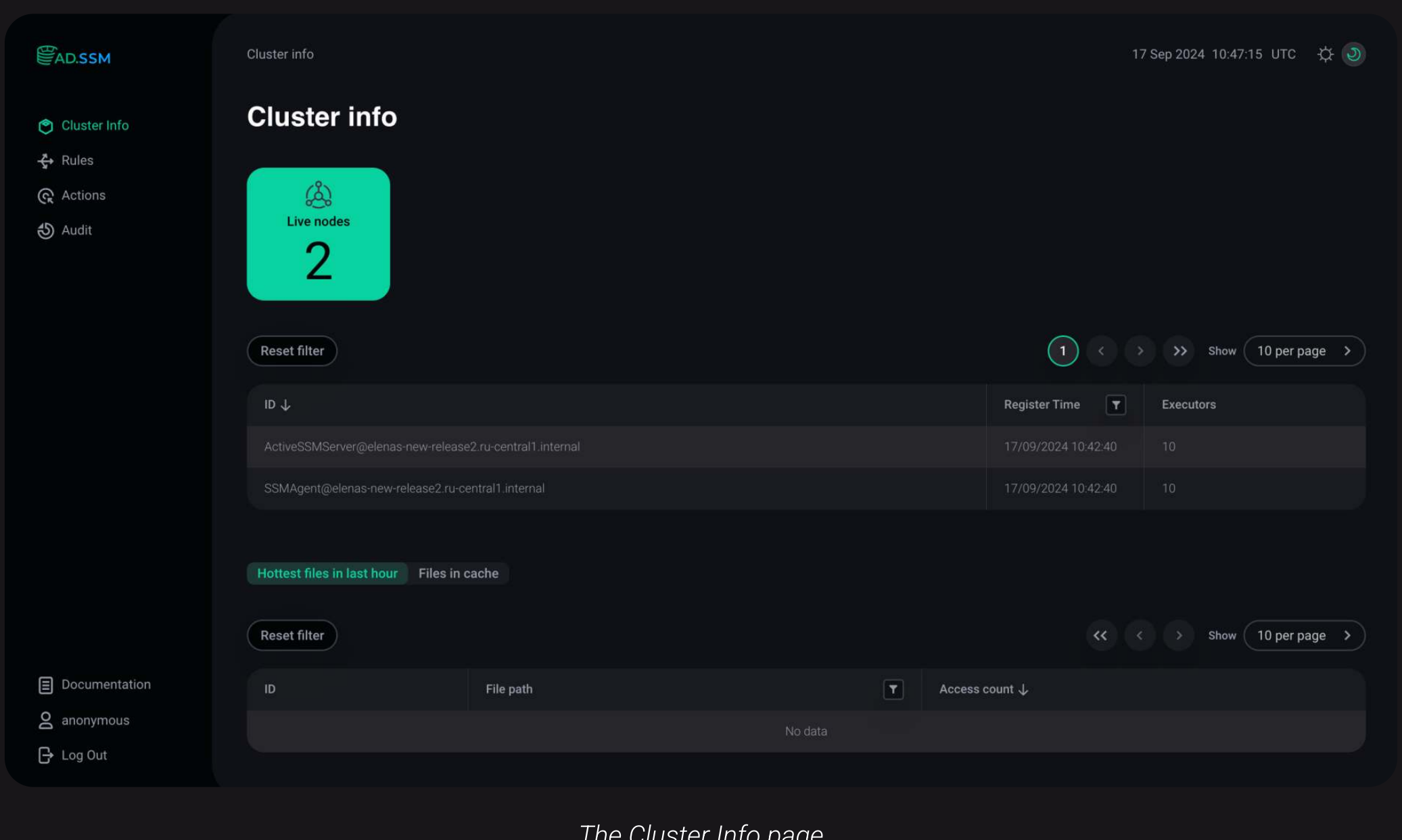
- ☰ ***Documentation*** — a link to the SSM documentation.
- 👤 — a user profile indicator that shows the name of the current logged in user.
- ⇥ ***Log Out*** — an option that closes the current user's session and redirects to the login screen.

Cluster Info

This page displays three tables:

- The nodes table displays information about the SSM Server/SSM Agent activity.
- The file statistics tables display information about all files and the files in cache:
  - ***Hottest files in last hour*** — displays top hot (frequently accessed) files in the last hour with the `accessCount` counter.
  - ***Files in cache*** — displays information about cached files.

To view either one of the file statistics tables, select ***Hottest files in last hour*** or ***Files in cache*** option over the table.



ID	Register Time	Executors
ActiveSSMServer@elenas-new-release2.ru-central1.internal	17/09/2024 10:42:40	10
SSMAgent@elenas-new-release2.ru-central1.internal	17/09/2024 10:42:40	10

ID	File path	Access count
No data		

The Cluster Info page

Nodes table

Field	Description
ID	The ID of the node, which has the `<component>@<host>` format. Where: - `<component>` — the name of the component installed on the host; - `<host>` — the FQDN of the host.
Register Time	The component start time in the `DD/MM/YYYY HH:mm:ss` format
Executors	The number of executors per node

You can sort table rows by clicking on a column's header. The ***Register Time*** column supports a filter, which can be accessed via the ▼ icon that you can use to select specific date range.

To reset all filters, click ***Reset***.

Hottest files in last hour table

Field	Description
ID	The numerical ID of the file
File path	The path to the file
Access count	The number of times the file was requested

You can sort table rows by clicking on a column's header. The ***File path*** column supports a filter, which can be accessed via the ▼ icon that you can use to search for a specific file path. To reset all filters, click ***Reset***.

Files in cache table

Field	Description
ID	The numerical ID of the file
File path	The path to the file
Cached Time	The time when the file was cached
Last Accessed Time	Last time the file was accessed
Access count	The number of times the file was requested

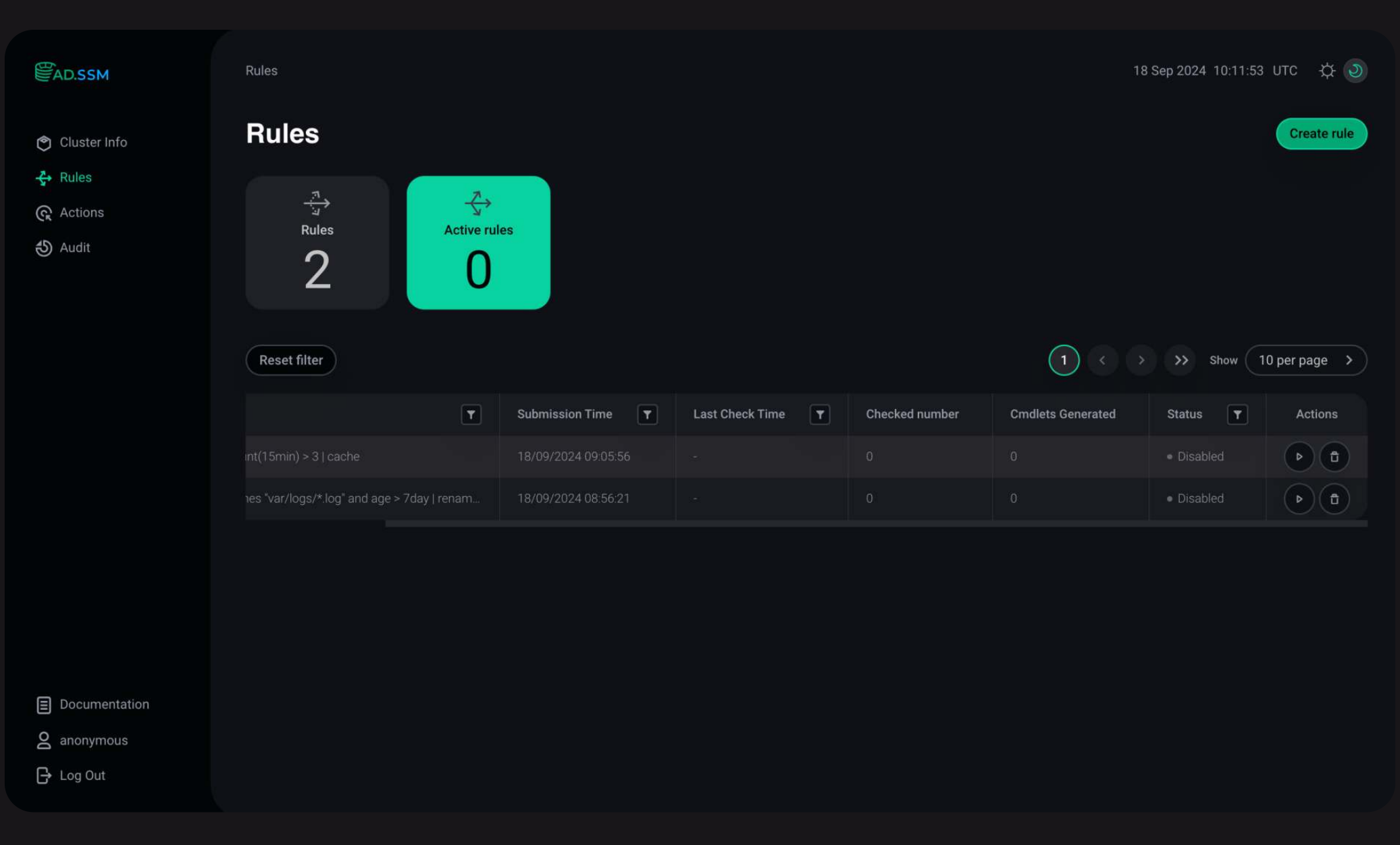
You can sort table rows by clicking on a column's header. Some columns support filters, which can be accessed via the ▼ icon:

- ***File path*** — provides the ability to search for a specific file path;
- ***Cached Time*** — allows filtering by the time a file was cached;
- ***Last Accessed Time*** — enables selecting the date of the last time a file was cached.

To reset all filters, click ***Reset***.

Rules

On this page, you can create a rule, view all the defined rules, change their status, and view the rules execution details. The widgets at the top of the page display the total number of existing rules and the number of active rules. The rules table contains the statistics on rules.



	Submission Time	Last Check Time	Checked number	Cmdlets Generated	Status	Actions
ent(15min) > 3 | cache	18/09/2024 09:05:56	-	0	0	● Disabled	
es "var/logs/*.log" and age > 7day | renam...	18/09/2024 08:56:21	-	0	0	● Disabled	

The Rules page

Rules table

Field	Description
ID	The numerical ID of the rule
Rule Text	The rule's expression
Submission Time	The time the rule was created in the `DD/MM/YYYY HH:mm:ss` format
Last Check Time	Last time the rule checked the condition in the `DD/MM/YYYY HH:mm:ss` format
Checked number	The number of times the rule checked the condition
Cmdlets Generated	The number of generated cmdlets
Status	The rule's activity indicator. Can be either `Disabled` or `Active`
Actions	The actions available for the rule: - ▷ — activate the rule; - ⏸ — pause the rule; - 🗑 — delete the rule.

You can sort table rows by clicking on a column's header. Some columns support filters, which can be accessed via the ▼ icon:

- ***Rule Text*** — provides the ability to search for rules with specific words or symbols;
- ***Submission Time*** — allows filtering by the time a rule was created;
- ***Last Check Time*** — enables selecting the date of the last time a rule checked conditions;
- ***Status*** — allows filtering by the rule status.

To reset all filters, click ***Reset***.

To create a rule:

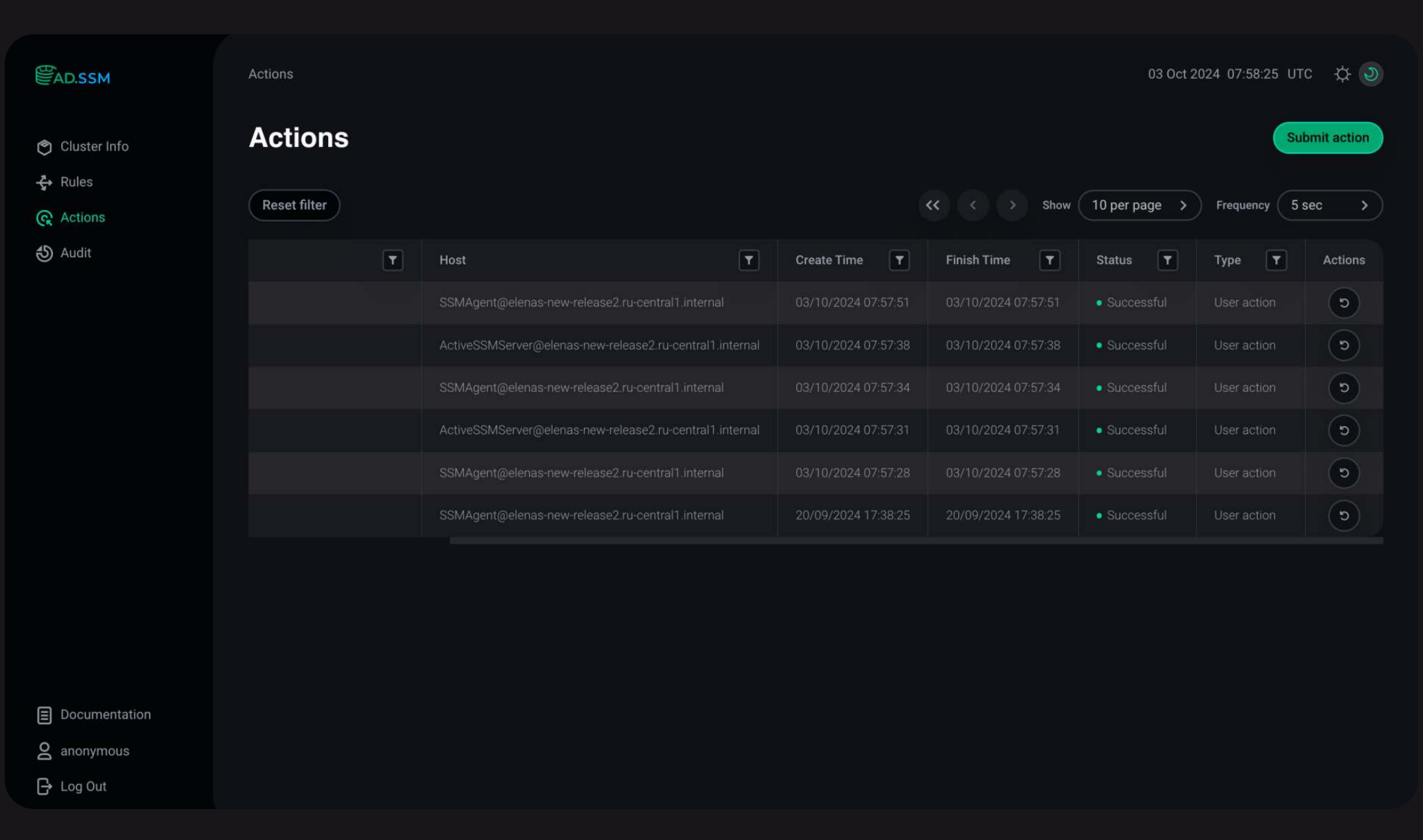
1. In the AD.SSM interface, go to the ***Rules*** section.
2. Select the ***Create rule*** option.
3. In the window that appears, describe the desired rule and click ***Create***.

Newly created rules appear on this page with the `DISABLED` status, meaning that even if a rule's condition is satisfied, no data movement actions take place. For a rule to become effective, you should activate it by clicking ▷.

Reversely, to stop the rule's activity, click ⏸. If you want to delete a rule, click 🗑.

Actions

This page lists all the submitted actions triggered by a rule or started manually.



	Host	Create Time	Finish Time	Status	Type	Actions
	SSMAgent@elenas-new-release2.ru-central1.internal	03/10/2024 07:57:51	03/10/2024 07:57:51	● Successful	User action	
	ActiveSSMServer@elenas-new-release2.ru-central1.internal	03/10/2024 07:57:38	03/10/2024 07:57:38	● Successful	User action	
	SSMAgent@elenas-new-release2.ru-central1.internal	03/10/2024 07:57:34	03/10/2024 07:57:34	● Successful	User action	
	ActiveSSMServer@elenas-new-release2.ru-central1.internal	03/10/2024 07:57:31	03/10/2024 07:57:31	● Successful	User action	
	SSMAgent@elenas-new-release2.ru-central1.internal	03/10/2024 07:57:28	03/10/2024 07:57:28	● Successful	User action	
	SSMAgent@elenas-new-release2.ru-central1.internal	20/09/2024 17:38:25	20/09/2024 17:38:25	● Successful	User action	

The Actions page

Actions table

Field	Description
ID	The numerical ID of the action
Action	The action's expression
Host	The node that ran the action
Create Time	The time the action was created in the `DD/MM/YYYY HH:mm:ss` format
Finish Time	The time the action was finished in the `DD/MM/YYYY HH:mm:ss` format
Status	The action's activity status. Can be one of the following: - `Running` - `Successful` - `Failed`
Type	The type of action depending on its trigger. It can be either triggered by a rule (`Rule action`) or a user (`User action`)
Actions	The operations available for the action. Currently, the only available operation is ↺ ***Repeat action***

You can sort table rows by clicking on a column's header. Some columns support filters, which can be accessed via the ▼ icon:

- ***Action*** — provides the ability to search for actions with specific words or symbols;
- ***Host*** — enables displaying the actions that were executed by a specific node;
- ***Create Time*** — allows filtering by the time an action was created;
- ***Finish Time*** — allows filtering by the time an action was finished;
- ***Status*** — displays actions with a specific status;
- ***Type*** — displays actions of a specific type.

To reset all filters, click ***Reset***.

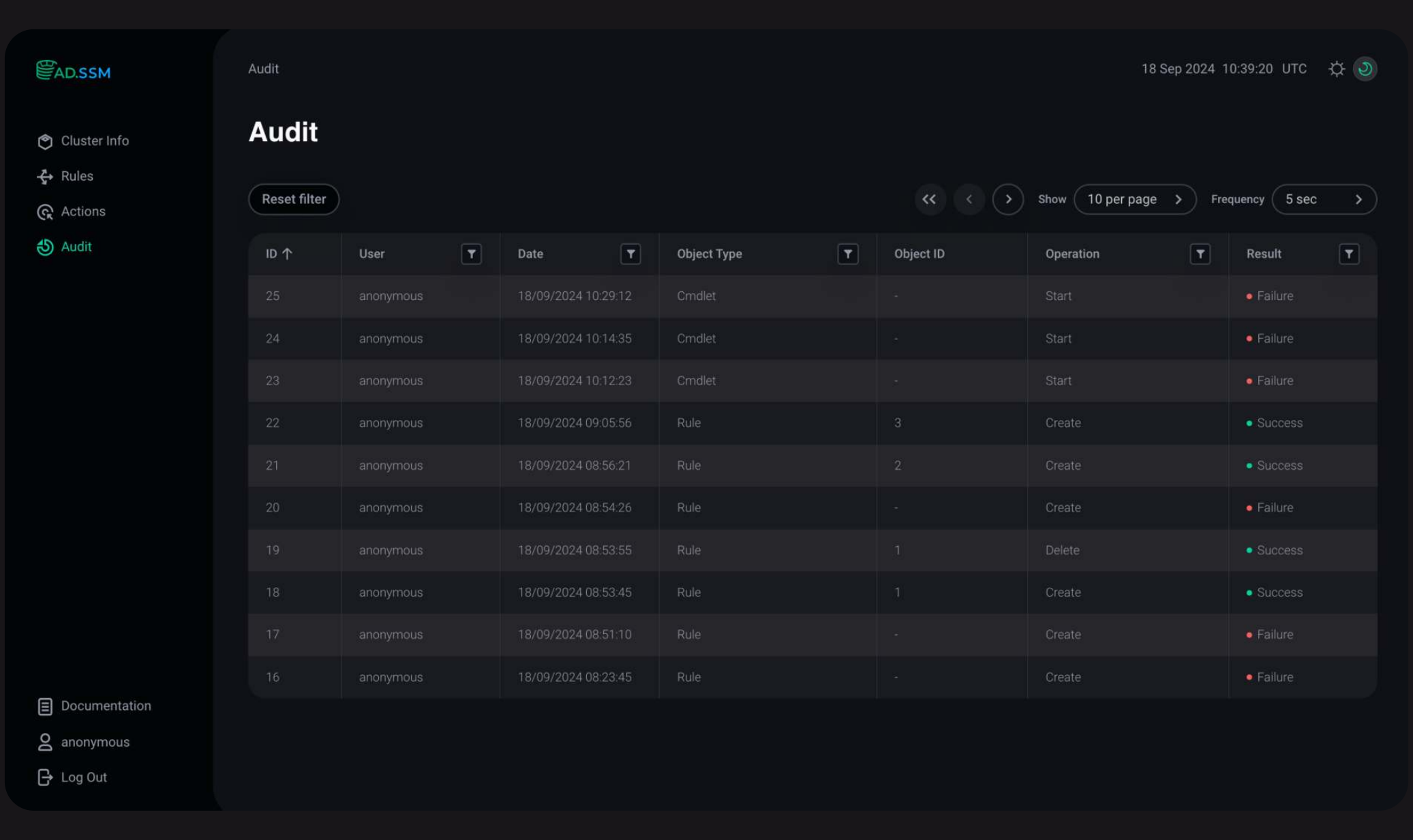
To run an action:

1. In the AD.SSM interface, go to the ***Actions*** section.
2. Select the ***Submit action*** option.
3. In the window that appears, describe the desired action and click ***Run***.

The action runs immidiately. If you want to repeat the action, click ↺.

Audit

This page displays a table with information about user actions in AD.SSM.



ID	User	Date	Object Type	Object ID	Operation	Result
25	anonymous	18/09/2024 10:29:12	Cmdlet	-	Start	● Failure
24	anonymous	18/09/2024 10:14:35	Cmdlet	-	Start	● Failure
23	anonymous	18/09/2024 10:12:23	Cmdlet	-	Start	● Failure
22	anonymous	18/09/2024 09:05:56	Rule	3	Create	● Success
21	anonymous	18/09/2024 08:56:21	Rule	2	Create	● Success
20	anonymous	18/09/2024 08:54:26	Rule	-	Create	● Failure
19	anonymous	18/09/2024 08:53:55	Rule	1	Delete	● Success
18	anonymous	18/09/2024 08:53:45	Rule	1	Create	● Success
17	anonymous	18/09/2024 08:51:10	Rule	-	Create	● Failure
16	anonymous	18/09/2024 08:23:45	Rule	-	Create	● Failure

The Audit page

Audit table

Field	Description
ID	The numerical ID of the event
User	The name of the user who performed an action
Date	The date and time of the event in the `DD/MM/YYYY HH:mm:ss` format
Object Type	The type of object that performed the action: a rule or a cmdlet
Object ID	The ID of the object
Operation	The operation performed by the user
Result	The result of the operation: `success` or `failure`

You can sort table rows by clicking on a column's header. Some columns support filters, which can be accessed via the ▼ icon:

- ***User*** — provides the ability to search for user names with specific words or symbols;
- ***Date*** — allows filtering by the time of the event;
- ***Object Type*** — displays events with a specific type of object;
- ***Operation*** — allows filtering by a specific type of operation: `Create`, `Delete`, `Start`, `Stop`;
- ***Result*** — displays events of a specific result.

To reset all filters, click ***Reset***.

Define rules in SSM

Sergei Tikhomirov

Contents

To Table of Contents

Overview

Add rules

Overview

In SSM, a rule is a construction that defines what action should be taken with given data once a certain condition was met. Such rules give you flexibility to adjust your cluster for specific needs. For example, with SSM rules, you can automate moving log files that are older than 7 days to a special location and rename them, or if a file is requested rather frequently, you can move it to cache.

A rule consists of four parts:

- **Objects**. Files to which the command will be applied.
- **Trigger**. An optional parameter that defines when the condition will be checked, e.g. at a certain time or every N hours.
- **Conditions**. An expression composed out of object property values and operators that is evaluated to `true` or `false`.
- **Commands**. Actions that are launched once the condition is met.

A template for a rule is presented below:

```
<objects>: <trigger> | <conditions> | <commands>
```

NOTE

The symbols `:` and `|` are reserved by SSM as separators and cannot be used within the rule parts, otherwise the rule parse will fail.

The forementioned examples can be implemented as shown below.

```
file: path matches "var/logs/*.log" and age > 7day | rename -dest "var/old_logs/"
file: accessCount(15min) > 3 | cache
```

A rule can be in four states:

- **Active**. When a rule is active, SSM will evaluate the conditions depending on the trigger and execute commands when the conditions are met.
- **Disabled**. Default state of a rule after it was added. An active rule that is paused gets into this state, it can be reactivated afterwards. If there were unfinished actions while pausing, you can choose whether to finish these actions or to cancel them during the reactivation. The default behavior is to cancel them.
- **Finished**. If a rule had a time trigger that was met and won't be met again, it gets into this state.
- **Deleted**. A state that a rule gets when the user deletes it, but there are still actions to complete. Once the actions are completed, the rule will be permanently deleted.

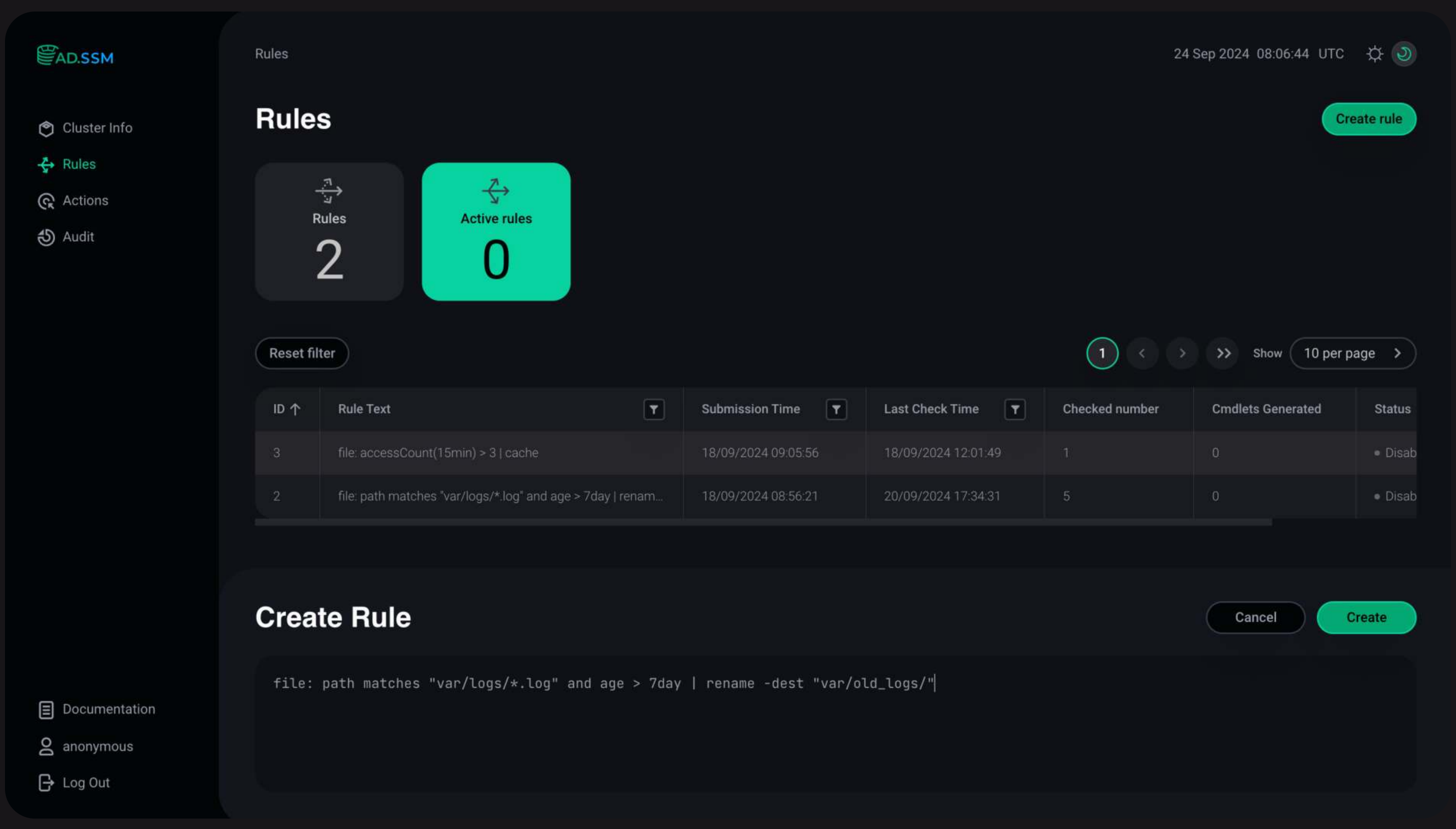
More information about the SSM rules is available in the SSM rules reference.

Add rules

To add a rule, you need to go the AD.SSM web interface. To do that, follow the steps below:

1. On the ***Clusters*** page, find your ADH cluster with SSM installed and click its name.
2. On the ***Services*** tab, click ***SSM*** and go to the ***Info*** tab in the service menu.
3. Click the link leading to the web interface.

In the web UI, you can add rules on the ***Rules*** page.



ID ↑	Rule Text	Submission Time	Last Check Time	Checked number	Cmdlets Generated	Status
3	file: accessCount(15min) > 3 | cache	18/09/2024 09:05:56	18/09/2024 12:01:49	1	0	● Disab
2	file: path matches "var/logs/*.log" and age > 7day | renam...	18/09/2024 08:56:21	20/09/2024 17:34:31	5	0	● Disab

```
file: path matches "var/logs/*.log" and age > 7day | rename -dest "var/old_logs/"
```

The Rules page

To create a rule:

1. In the AD.SSM interface, go to the ***Rules*** section.
2. Select the ***Create rule*** option.
3. In the window that appears, describe the desired rule and click ***Create***.

The created rule will have the `Disabled` status. To activate it, click ▷. The status will change to `Active`.

Define actions in SSM

Sergei Tikhomirov

Contents

To Table of Contents

Overview

Launch an action

Action usage example

Overview

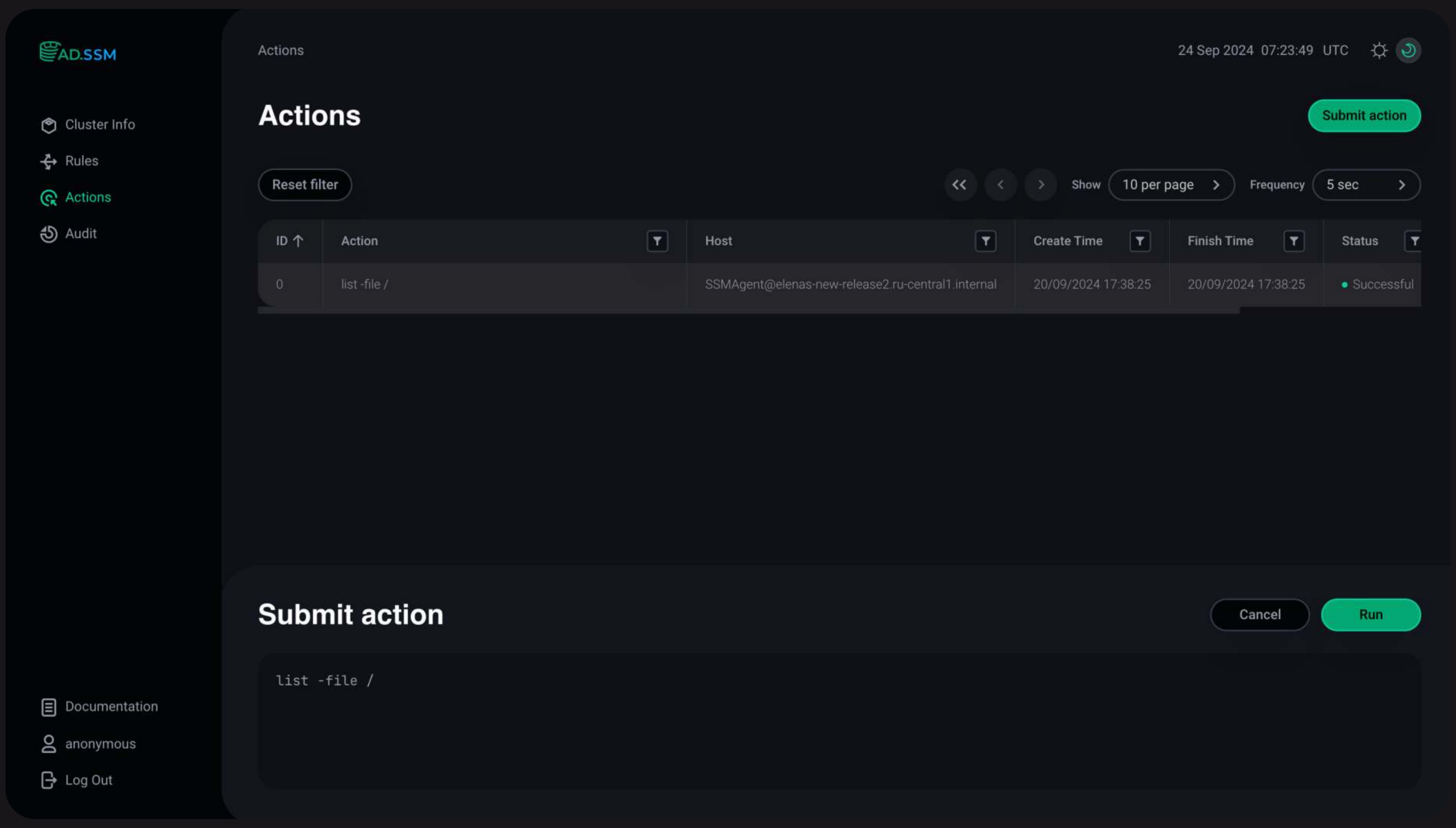
Actions in SSM are commands that are applied to data if a certain condition in a rule is met. You can launch an action at any time you want using the ***Actions*** page in the AD.SSM UI.

For a full list of actions available in SSM, refer to the SSM Actions reference.

Launch an action

To launch an action, follow the steps below:

1. In the AD.SSM interface, open the ***Actions*** page.
2. Select the ***Submit action*** option.
3. In the ***Submit action*** pane, type the command with parameters and click ***Run***.



AD.SSM
Cluster Info
Rules
Actions
Audit
Documentation
anonymous
Log Out
Actions
24 Sep 2024 07:23:49 UTC
Actions
Submit action
Reset filter
Show
10 per page
Frequency
5 sec
Submit action
Cancel
Run

ID	Action	Host	Create Time	Finish Time	Status
0	list -file /	SSMAgent@elenas-new-release2.ru-central1.internal	20/09/2024 17:38:25	20/09/2024 17:38:25	● Successful

```
list -file /
```

The Actions page with the Submit action pane

NOTE

For navigation, you can use the command below, which lists all content in an HDFS directory:

```
list -file /
```

Action usage example

Below, you can see how to define an action that copies files to an S3 storage.

After configuring S3 credentials, you can copy files to an S3 storage by running an action like this:

```
copy2s3 -file /user/sergei/demoCopyS3/f1.txt -dest s3a://develop-test-bucket/f1.txt
```

You can run the following command to check that a file got copied:

```
$ hdfs dfs -cat s3a://develop-test-bucket/f1.txt
```

If there are no errors, you will see the file contents.

SSM rule usage examples

Sergei Tikhomirov

Contents

To Table of Contents

Overview

Cache files

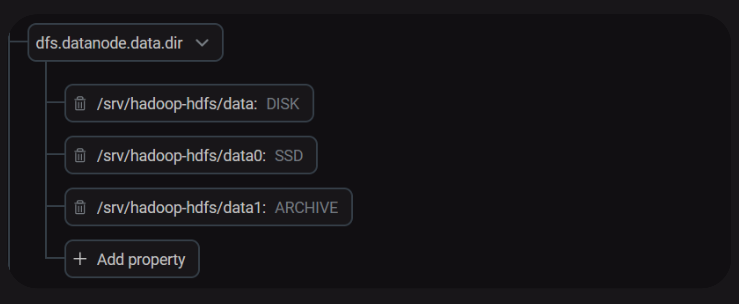
Move data between cold storage and hot storage

Sync data

Overview

This article contains advanced examples of rule usage in SSM. Before running the examples, complete the steps below:

1. Deploy SSM.
2. Configure the storage types such as SSD, DISK, and ARCHIVE in the ***hdfs-site.xml*** → ***dfs.datanode.data.dir*** parameter in ADCM.



dfs.datanode.data.dir

/srv/hadoop-hdfs/data: DISK

/srv/hadoop-hdfs/data0: SSD

/srv/hadoop-hdfs/data1: ARCHIVE

Storage types

Cache files

If you want to move frequently accessed files to cache, you can use the rule like below:

```
file: accessCount(1min) > 0 and path matches "/user/sergei/demoCacheFile/*" | cache
```

With this rule active, SSM caches all files in the *demoCacheFile* directory which were accessed at least once during the last minute.

For test purposes, several files were created in the *user/sergei/demoCacheFile* directory in HDFS. The command below prints out the filenames:

```
list -file /user/sergei/demoCacheFile
```

```
Action starts at Wed Feb 14 14:05:59 UTC 2024 : List /user/sergei/demoCacheFile
-rw-r--r--     3 sergei sergei             10 2024-02-14 14:04 /user/sergei/demoCacheFile/f1.txt
-rw-r--r--     3 sergei sergei             13 2024-02-14 14:04 /user/sergei/demoCacheFile/f2.txt
-rw-r--r--     3 sergei sergei             12 2024-02-14 14:04 /user/sergei/demoCacheFile/f3.txt
```

To check that the example works, try accessing one file by running the `read` action:

```
read -file /user/sergei/demoCacheFile/f1.txt
```

If there are no errors, you should be able to see that file on the ***Cluster Info*** → ***Files in cache*** page.

Move data between cold storage and hot storage

IMPORTANT

In order for this example to work, make sure you have configured the SSD and ARCHIVE storage types in HDFS.

To move files from cold storage to hot storage, use the rule like below:

```
file: accessCount(1min) > 0 and path matches "/user/sergei/demoCold2Hot/*" | allssd
```

To check the storage type, call the `checkstorage` action in the ***Submit action*** pane:

```
checkstorage -file /user/sergei/demoCold2Hot/f1.txt
```

Before the rule application, the command output is:

```
File offset = 0, Block locations = {10.92.41.28:9866[DISK] 10.92.41.202:9866[DISK] 10.92.41.33:9866[DISK] }
```

After triggering the condition by reading the file, the rule got applied and the output became:

```
File offset = 0, Block locations = {10.92.41.33:9866[SSD] 10.92.41.28:9866[SSD] 10.92.41.202:9866[SSD] }
```

Now, you can also see the file on the ***Cluster Info*** → ***Hottest files in last hour*** page.

To move files from the hot storage to cold storage, use the rule below:

```
file : age > 3min and path matches "/user/sergei/demoHot2Cold/*" | archive
```

Sync data

To synchronize files in a directory with another HDFS cluster, use the rule like below:

```
file: path matches "/user/sergei/demoSyncSrc/*" | sync -dest hdfs://stikhomirov-adh1.ru-central1.internal/demoSyncDest
```

If the SSM cluster has a special namespace configured, you can substitute the namespace name for a host name in the command above.

NOTE

In order for the `sync` command to work, at least one server (source or destination) needs to have SSM installed. Also, the source directory doesn't have to be local, just as the destination doesn't have to be on the remote server.

To check the correctness, compare the checksums. You can call the `checksum` command in the ***Submit action*** pane:

```
checksum -file /user/sergei/demoSyncSrc/f1.txt
```

The output for this particular file is:

```
/user/sergei/demoSyncSrc/f1.txt MD5-of-0MD5-of-512CRC32C        000002000000000000000000049c19b14a1077d0a6fc95bee2db8914c
```

To check the checksum on the destination server, call the following command there:

```
$ hadoop fs -checksum /demoSyncDest/f1.txt
```

The checksum matches the one above:

```
/demoSyncDest/f1.txt    MD5-of-0MD5-of-512CRC32C        000002000000000000000000049c19b14a1077d0a6fc95bee2db8914c
```

SSM service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the SSM service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the SSM service are listed in the table below.

SSM service actions

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute SSM components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

The actions available for the SSM service components are listed in the table below.

SSM component actions

Action	Description
Add/Remove Agents	Adds/removes the SSM Agent component(s). Running this action opens the component-host mapping interface where you can distribute SSM Agent components
Add/Remove Servers	Adds/removes the SSM Server component(s). Running this action opens the component-host mapping interface where you can distribute SSM Server components
Check	Runs component-specific tests to check the health of the given SSM component
Restart	Restarts the component. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all component configurations defined in ADCM settings will be applied during the component restart. Otherwise, the component restarts without applying configurations from ADCM
Start	Starts the component. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all component configurations defined in ADCM settings will be applied on the component startup. Otherwise, the component starts without applying configurations from ADCM
Stop	Stops the component

Connect to Trino via CLI

Vladimir Adamenkov, Konstantin Alpashkin

Contents

To Table of Contents

- Trino Client component
  - Usage
  - Statements execution
  - Configuration
  - TLS/HTTPS
- Remote connection

Trino Client component

The Trino service includes a Trino Client component, which provides a command-line interface (CLI) for submitting SQL statements to the Trino Coordinator. By default, when a Trino Client component is installed, it is automatically configured to work with the Trino Coordinator available in your ADH cluster.

Usage

To connect to Trino via CLI, on the ADH host with the Trino Client component installed, run the command:

```
$ trino
```

This opens the Trino Client prompt:

```
trino>
```

To view the list of supported commands, execute the `help` command. Example:

```
help
```

```
Supported commands:
QUIT
EXIT
CLEAR
EXPLAIN [ ( option [, ...] ) ] <query>
    options: FORMAT { TEXT | GRAPHVIZ | JSON }
            TYPE { LOGICAL | DISTRIBUTED | VALIDATE | IO }
DESCRIBE <table>
SHOW COLUMNS FROM <table>
SHOW FUNCTIONS
SHOW CATALOGS [LIKE <pattern>]
SHOW SCHEMAS [FROM <catalog>] [LIKE <pattern>]
SHOW TABLES [FROM <schema>] [LIKE <pattern>]
USE [<catalog>.]<schema>
```

To clear the terminal screen, use the `clear` command. To exit the Trino CLI and go back to the OS CLI, use the `exit` or `quit` commands.

Statements execution

When the Trino Client prompt is active, you can run SQL statements. The CLI will return the results of statements processing and their statistics. Example:

```
select * from "hive-adh"."default"."solar_objects";
```

The output:

```
  name   |    mass    | diameter
---------+------------+----------
 Sun     | 1989100000 |  1392000
 Mercury |        330 |     4879
 Venus   |       4867 |    12104
 Earth   |       5972 |    12742
 Mars    |        642 |     6780
 Jupiter |    1898187 |   139822
 Saturn  |     568317 |   116464
 Uranus  |      86813 |    50724
 Neptune |     102413 |    49244
(9 rows)

Query 20250115_092203_00003_7thb6, FINISHED, 1 node
Splits: 1 total, 1 done (100,00%)
1,51 [9 rows, 172B] [5 rows/s, 114B/s]
```

In this example, `hive-adh` is the name of the catalog, `default` is the schema name, and `solar_objects` is the table name.

TIP

By using Trino Client settings in ADCM, you can specify the default Trino catalog and schema.

Configuration

By default, the Trino Client component is configured to work with the Trino Coordinator available in the ADH cluster and requires no manual setup. If you need to tune the Trino Client behavior, use the Trino Client settings in ADCM.

TLS/HTTPS

The Trino Client component supports secure connection to Trino Coordinator using TLS. To configure a secure connection, use the ***SSL Configuration*** section of the Trino Client component settings in ADCM.

Remote connection

You can also connect to your Trino service from a remote host outside the ADH cluster. For this, you can use the trino-cli tool. It is implemented by a self-executing JAR file, which can be launched in any OS provided there is a Java runtime environment (JRE) installed.

To use the Trino CLI tool, you need to download the JAR executable. The link to the most recent file is available at the Trino documentation website. If you use Linux OS, rename the file to *trino* and make it executable using the `chmod +x` command. If you use Windows OS, run the Windows CLI as administrator before you launch the JAR file in it.

To launch the Trino CLI tool, open a terminal in your OS and use the following command syntax, using the network address of the Trino Coordinator component.

Windows

```
java -jar trino-cli-468-executable.jar http://coordinator.trino.example:18188
```

Linux

```
$ ./trino http://coordinator.trino.example:18188
```

You can also use various options when starting the Trino CLI. A list of those options is provided on the Command line interface page of the Trino documentation.

Connect to Trino via JDBC

Vladimir Adamenkov

Contents

To Table of Contents

Overview

Installation

Usage

- Connection
- Connection parameters

Overview

The Trino service supports user access from Java applications using the JDBC driver. It uses the client protocol over HTTP/HTTPS to connect to the Trino Coordinator component.

Installation

Before the installation of the JDBC driver make sure the following prerequisites are met:

1. Java runtime environment (JRE) version 8 or higher is installed. JRE version 22 or higher is recommended.
2. Users that are supposed to connect to Trino must have access to query tables, which is defined in the `system.jdbc` schema.
3. The Trino Coordinator component is network accessible.

Some client applications already have a built-in Trino JDBC driver. If yours does not, download the driver file. The link to the most recent file is available at the Trino documentation website or at Maven central repository. Add the downloaded file to the directory in the users classpath on the machines that are supposed to connect to Trino, and restart the client application. You may probably have to manually register the Trino JDBC driver in the client application: some of them do it automatically, some do not. Refer to the documentation of your client application for assistance.

Usage

Connection

After the Trino JDBC driver has been registered and configured in your client application, you can connect to Trino. Use the following JDBC URL formats:

```
jdbc:trino://<host>:<port>
jdbc:trino://<host>:<port>/<catalog>
jdbc:trino://<host>:<port>/<catalog>/<schema>
```

where:

- `<host>` is the network address of the Trino Coordinator component node;
- `<port>` is the listening port of the Trino Coordinator component (18188 by default);
- `<catalog>` is the name of the Trino catalog to use;
- `<schema>` is the name of the DB or schema to connect to.

Connection parameters

You can define additional connection parameters as URL parameters or as properties passed to the `DriverManager` class. For example, you may want to connect to Trino using the following parameters:

- network address and port — `myorganization.org:18188`;
- catalog and schema — `hive/tech`;
- user — `admin`;
- password — `MySecretPassword1984`.

In this case, you will need to use the following URL parameters:

```
String mycustomurl = "jdbc:trino://myorganization.org:18188/hive/tech?user=admin&password=MySecretPassword1984";
Connection connection = DriverManager.getConnection(mycustomurl);
```

Or if you want to pass the parameters to `DriverManager`, you will need to define the following properties:

```
String url = "jdbc:trino://myorganization.org:18188/hive/tech";
Properties mycustomproperties = new Properties();
mycustomproperties.setProperty("user", "admin");
mycustomproperties.setProperty("password", "MySecretPassword1984");
Connection connection = DriverManager.getConnection(url, mycustomproperties);
```

You can use a combination of these approaches, but overlapping of the parameters is prohibited: a parameter defined by one method cannot be defined by the other.

List of available parameters is provided on the JDBC driver page of the Trino documentation.

Trino UI overview

Vladimir Adamenkov

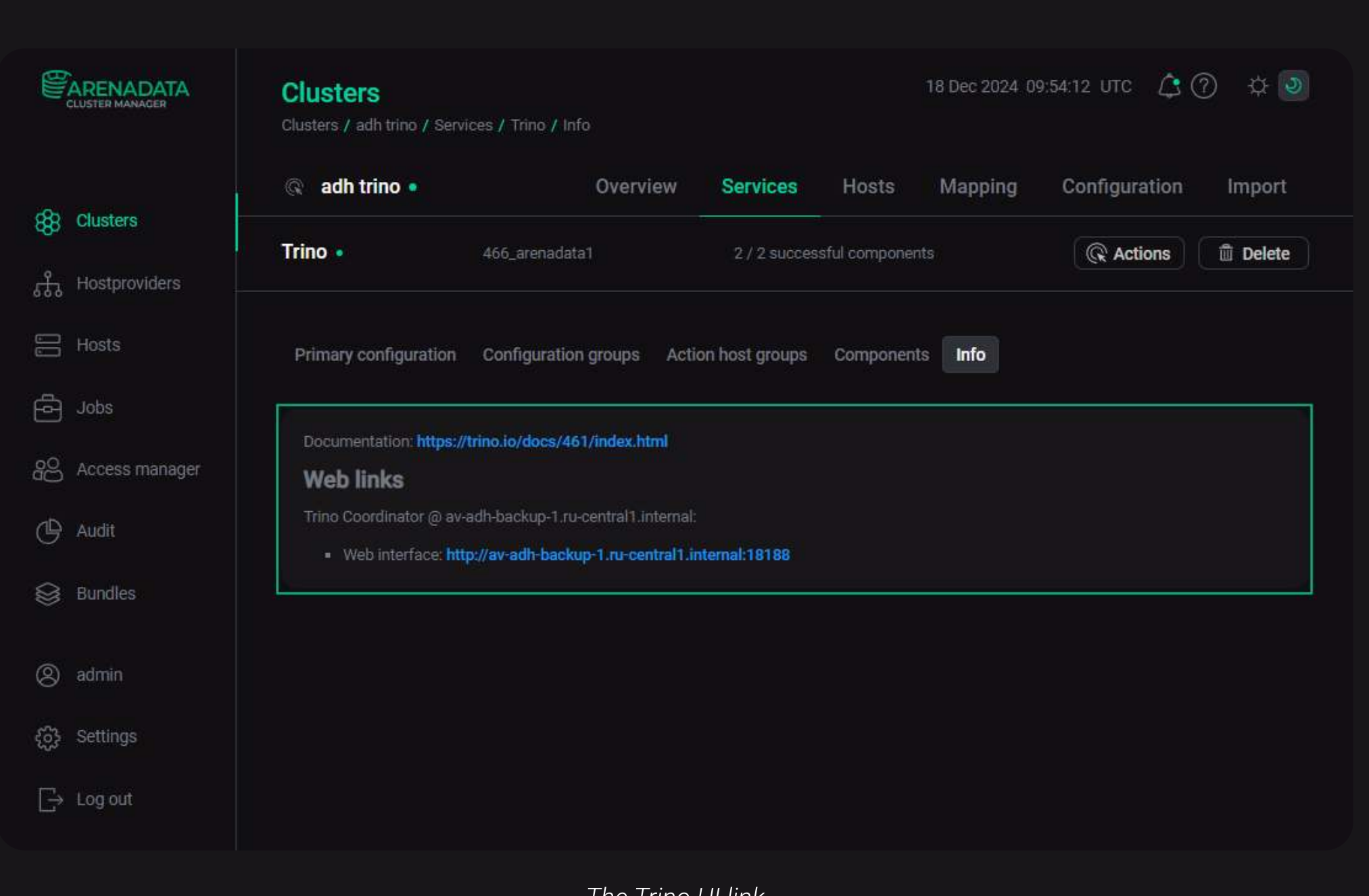
Contents

To Table of Contents

- Cluster overview
- Query details
  - Overview
  - Live plan
  - Stage performance
  - Splits
  - JSON
  - References

The Trino service has a web user interface (UI) that allows you to view and control the execution of the queries ran against the databases located in the Trino cluster.

You can find the link to the Trino UI on the ***Info*** tab of the Trino service in the ADCM UI.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / adh trino / Services / Trino / Info
18 Dec 2024 09:54:12 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
adh trino
Overview
Services
Hosts
Mapping
Configuration
Import
Trino
466_arenadata1
2 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info

Documentation: https://trino.io/docs/461/index.html

Web links

Trino Coordinator @ av-adh-backup-1.ru-central1.internal:

- Web interface: http://av-adh-backup-1.ru-central1.internal:18188

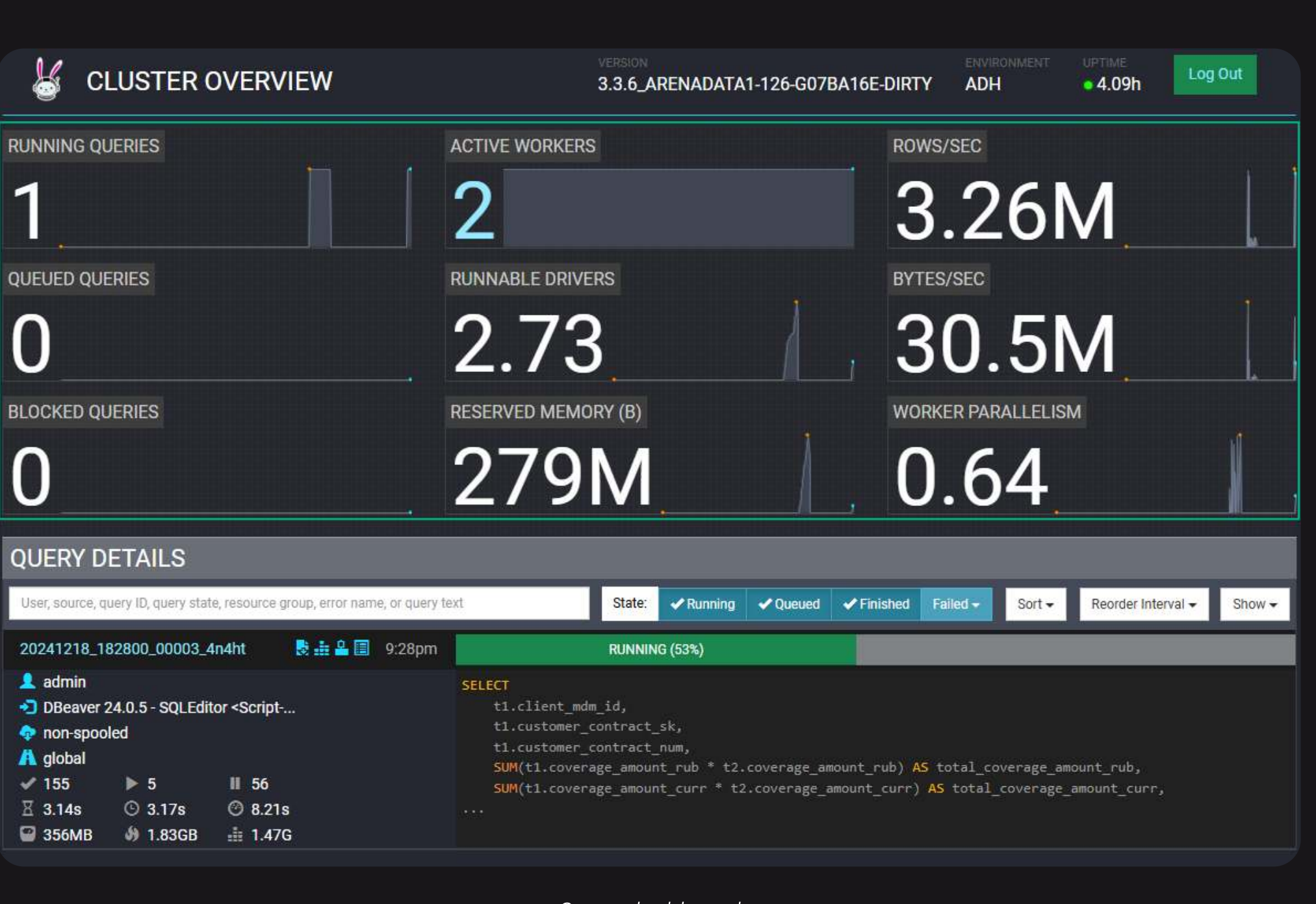
The Trino UI link

Cluster overview

The start page of the Trino UI is the ***Cluster overview*** page. On the top of it, you can see the following details:

- ***Version*** — version name of the installed Trino distribution.
- ***Environment*** — name of the environment in which the current Trino cluster is working.
- ***Uptime*** — time of the continuous functioning of the Trino Coordinator component.
- ***Log out*** — button that allows you to log out of the UI.

Below that there is a dashboard with a number of gauges.



CLUSTER OVERVIEW
VERSION
3.3.6_ARENADATA1-126-G07BA16E-DIRTY
ENVIRONMENT
ADH
UPTIME
4.09h
Log Out
RUNNING QUERIES
1
ACTIVE WORKERS
2
ROWS/SEC
3.26M
QUEUED QUERIES
0
RUNNABLE DRIVERS
2.73
BYTES/SEC
30.5M
BLOCKED QUERIES
0
RESERVED MEMORY (B)
279M
WORKER PARALLELISM
0.64
QUERY DETAILS
User, source, query ID, query state, resource group, error name, or query text
State:
Running
Queued
Finished
Failed
Sort
Reorder Interval
Show
20241218_182800_00003_4n4ht
9:28pm
RUNNING (53%)
admin
DBeaver 24.0.5 - SQLEditor <Script-...
non-spooled
global
155
5
56
3.14s
3.17s
8.21s
356MB
1.83GB
1.47G

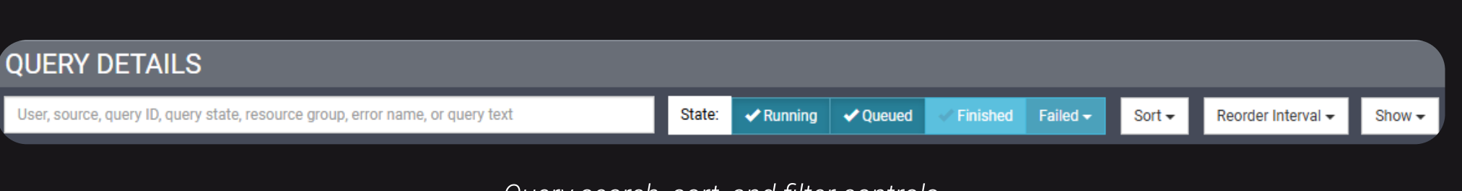
```
SELECT
    t1.client_mdm_id,
    t1.customer_contract_sk,
    t1.customer_contract_num,
    SUM(t1.coverage_amount_rub * t2.coverage_amount_rub) AS total_coverage_amount_rub,
    SUM(t1.coverage_amount_curr * t2.coverage_amount_curr) AS total_coverage_amount_curr,
...
```

Query dashboard

Designation of the gauges:

- ***Running queries*** — number of queries currently underway.
- ***Active workers*** — number of Trino workers engaged in execution of the running queries.
- ***Rows/sec*** — number of table rows being processed per second at the current moment.
- ***Queued queries*** — number of queries waiting in the queue.
- ***Runnable drivers*** — number of runnable drivers at the current moment.
- ***Bytes/sec*** — amount of data being processed per second at the current moment.
- ***Blocked queries*** — number of queries prevented from execution.
- ***Reserved memory*** — amount of Trino Java virtual machine (JVM) memory reserved for the currently running queries.
- ***Worker parallelism*** — measure of effectivenes of using multiple workers at the current moment.

Under the dashboard there is the ***Query details*** pane. You can view the statistics on all kind of queries. Also, you can search, filter, and sort the list of queries by various criteria.



QUERY DETAILS
User, source, query ID, query state, resource group, error name, or query text
State:
Running
Queued
Finished
Failed
Sort
Reorder Interval
Show

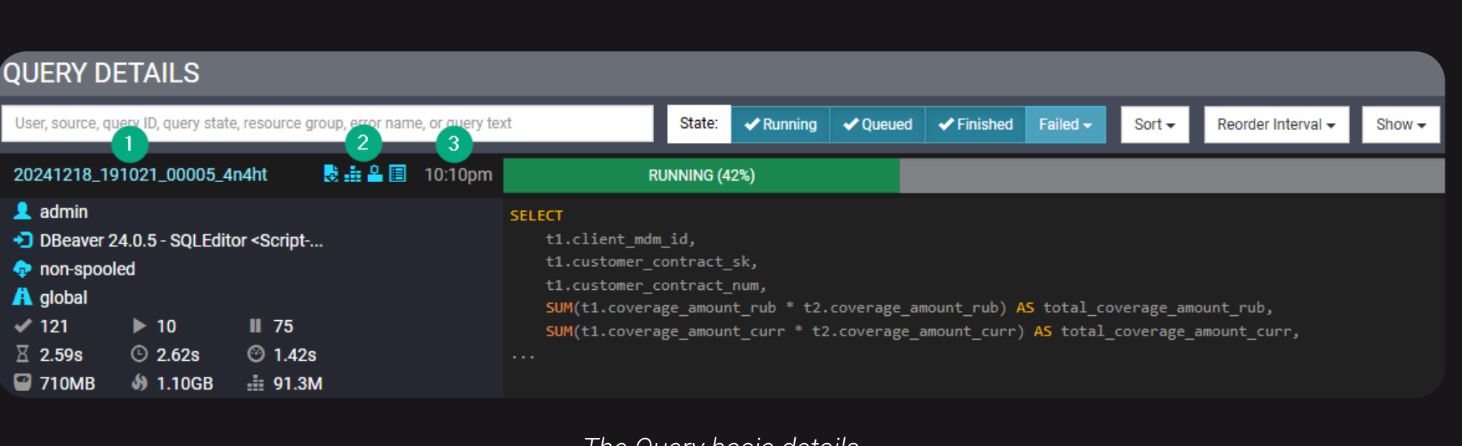
Query search, sort, and filter controls

In the input field, you can search for queries by the following properties:

- username that executed the query;
- source of the query;
- ID of the query;
- state of the query;
- resource group of the query;
- name of the error with which the query has terminated;
- text of the query.

To the right of the search field there are the filter and sort controls. To view the running, queued, or successfully finished queries in the list, click the corresponding toggle buttons. The ***Failed*** drop-down list combines the toggles for the queries that finished unsuccessfully for different reasons: internal or external error, resources error, or user error. Using the ***Show*** drop-down list, you can choose how many queries to display. The ***Sort*** drop-down list allows you to sort the list by various parameters. Finally, using the ***Reorder interval*** control, you can set the refresh frequency for the list or disable it.

Below that begins the list of queries. Each query entry displays basic details of that query.



QUERY DETAILS
User, source, query ID, query state, resource group, error name, or query text
State:
Running
Queued
Finished
Failed
Sort
Reorder Interval
Show
1
2
3
20241218_191021_00005_4n4ht
10:10pm
RUNNING (42%)
admin
DBeaver 24.0.5 - SQLEditor <Script-...
non-spooled
global
121
10
75
2.59s
2.62s
1.42s
710MB
1.10GB
91.3M

```
SELECT
    t1.client_mdm_id,
    t1.customer_contract_sk,
    t1.customer_contract_num,
    SUM(t1.coverage_amount_rub * t2.coverage_amount_rub) AS total_coverage_amount_rub,
    SUM(t1.coverage_amount_curr * t2.coverage_amount_curr) AS total_coverage_amount_curr,
...
```

The Query basic details

The top left heading contains the following details:

① Query ID — the ID of the query. Clicking on it will open the ***Overview*** tab of the Query details page.

② Query JSON, Stage performance, Query plan, and References icons — clicking on them is equivalent to clicking on the tabs with the same name on the ***Query details*** page.

③ Submit time — time when the query was submitted for execution.

Below the heading the following information is compiled:

- username that executed the query;
- source of the query;
- protocol encoding;
- resource group of the query;
- numbers of completed, running, and queued splits;
- wall time spent executing the query excluding the queue time;
- total query wall time;
- CPU time spent by the query execution;
- current total reserved memory;
- peak total reserved memory;
- cumulative user memory.

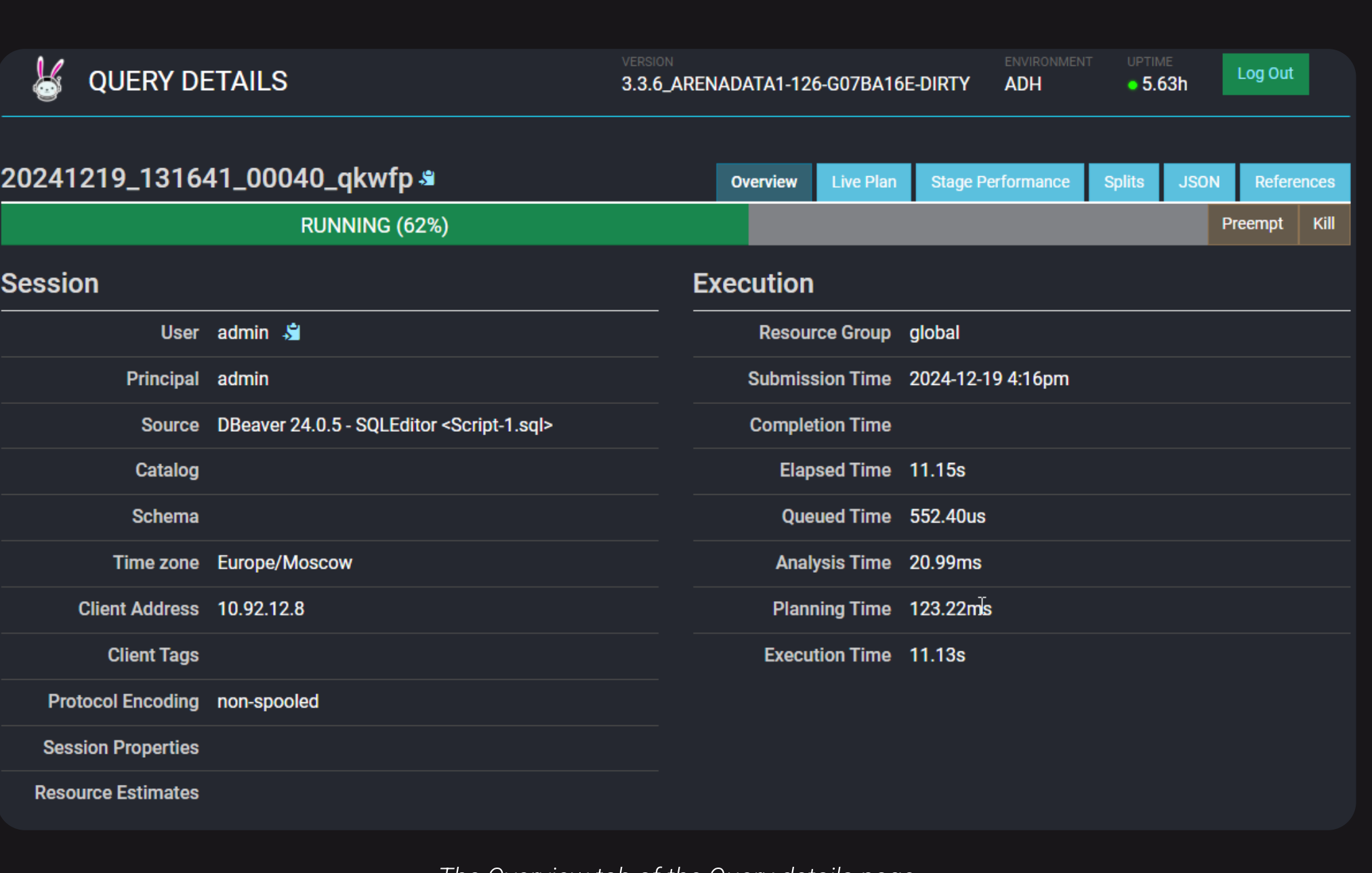
In the right part of the query entry, you can view the query execution progress status or its resolution, depending whether it is still running or already finished. Also, the text of the query (or its beginning) is listed here.

Query details

The ***Query details*** page can be opened by clicking the query ID on the Cluster overview page. It contains several tabs described below.

Overview

The ***Overview*** tab contains the general details of the query.



QUERY DETAILS
VERSION
3.3.6_ARENADATA1-126-G07BA16E-DIRTY
ENVIRONMENT
ADH
UPTIME
5.63h
Log Out
20241219_131641_00040_qkwfp
Overview
Live Plan
Stage Performance
Splits
JSON
References
RUNNING (62%)
Preempt
Kill

Session	
User	admin
Principal	admin
Source	DBeaver 24.0.5 - SQLEditor <Script-1.sql>
Catalog	
Schema	
Time zone	Europe/Moscow
Client Address	10.92.12.8
Client Tags	
Protocol Encoding	non-spooled
Session Properties	
Resource Estimates	

Execution	
Resource Group	global
Submission Time	2024-12-19 4:16pm
Completion Time	
Elapsed Time	11.15s
Queued Time	552.40us
Analysis Time	20.99ms
Planning Time	123.22ms
Execution Time	11.13s

The Overview tab of the Query details page

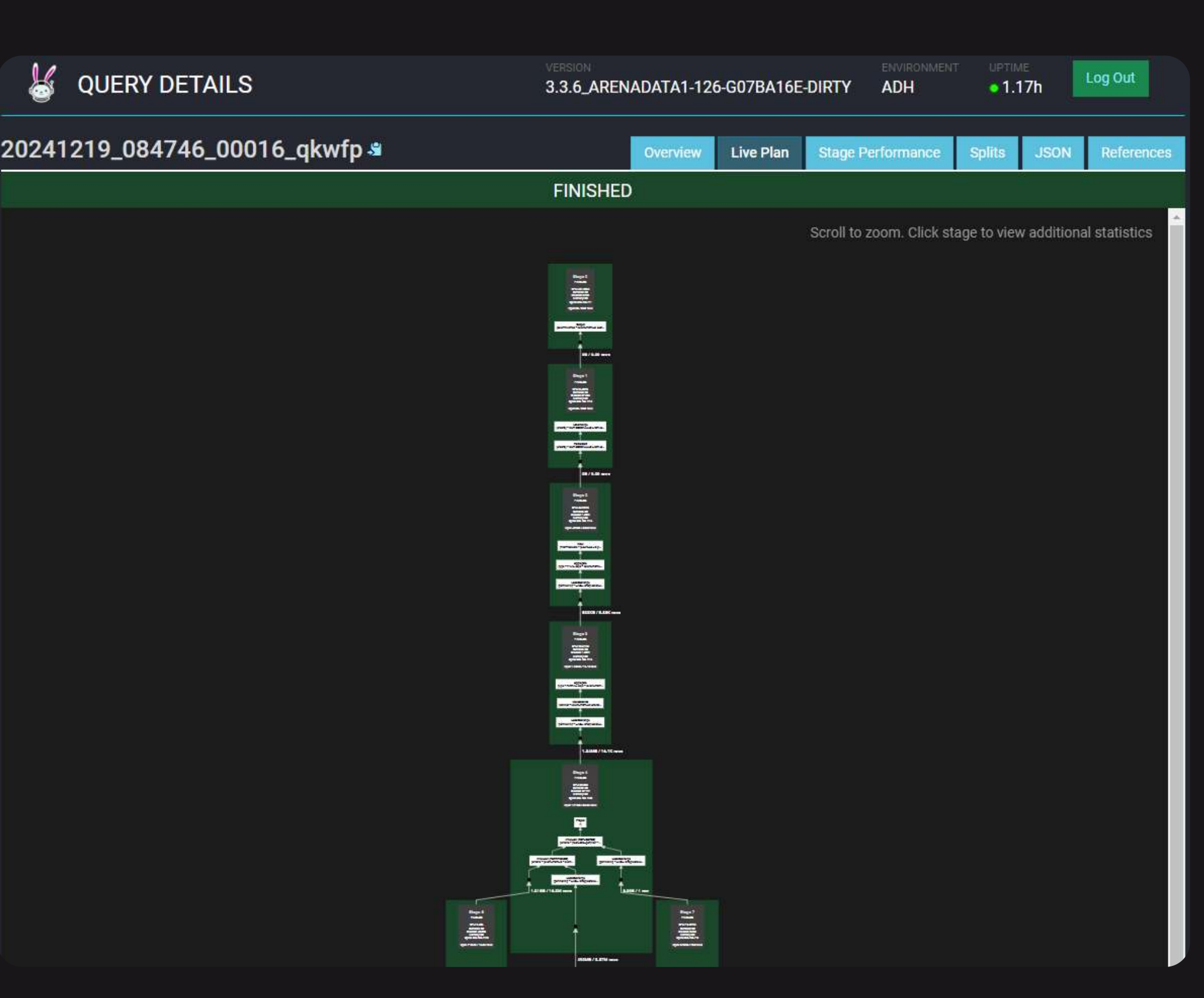
Here you can preempt or kill the query execution if it is still running, and find the detailed information on the following aspects of the query:

- ***Session*** — user, source, location, and others.
- ***Execution*** — various timings of the query.
- ***Resource utilization summary*** — statistics of resource usage while running the query.
- ***Timeline*** — statistics of query execution metrics in terms of the timeline.
- ***Error information*** — error type and code, and the listing of the stack trace.

In the bottom of this tab, you also can see the full query text and the list of detailed query execution stages.

Live plan

On the ***Live plan*** tab, you can see the scheme of the query execution by stages with detailed information on each stage: completion status, resource usage, clauses processed, and the direction of further processing of the query. You can zoom in and out by scrolling up and down respectively, and move the scheme by left click dragging. Clicking on a stage in the scheme will take you to the Stage performance tab with that particular stage chosen.

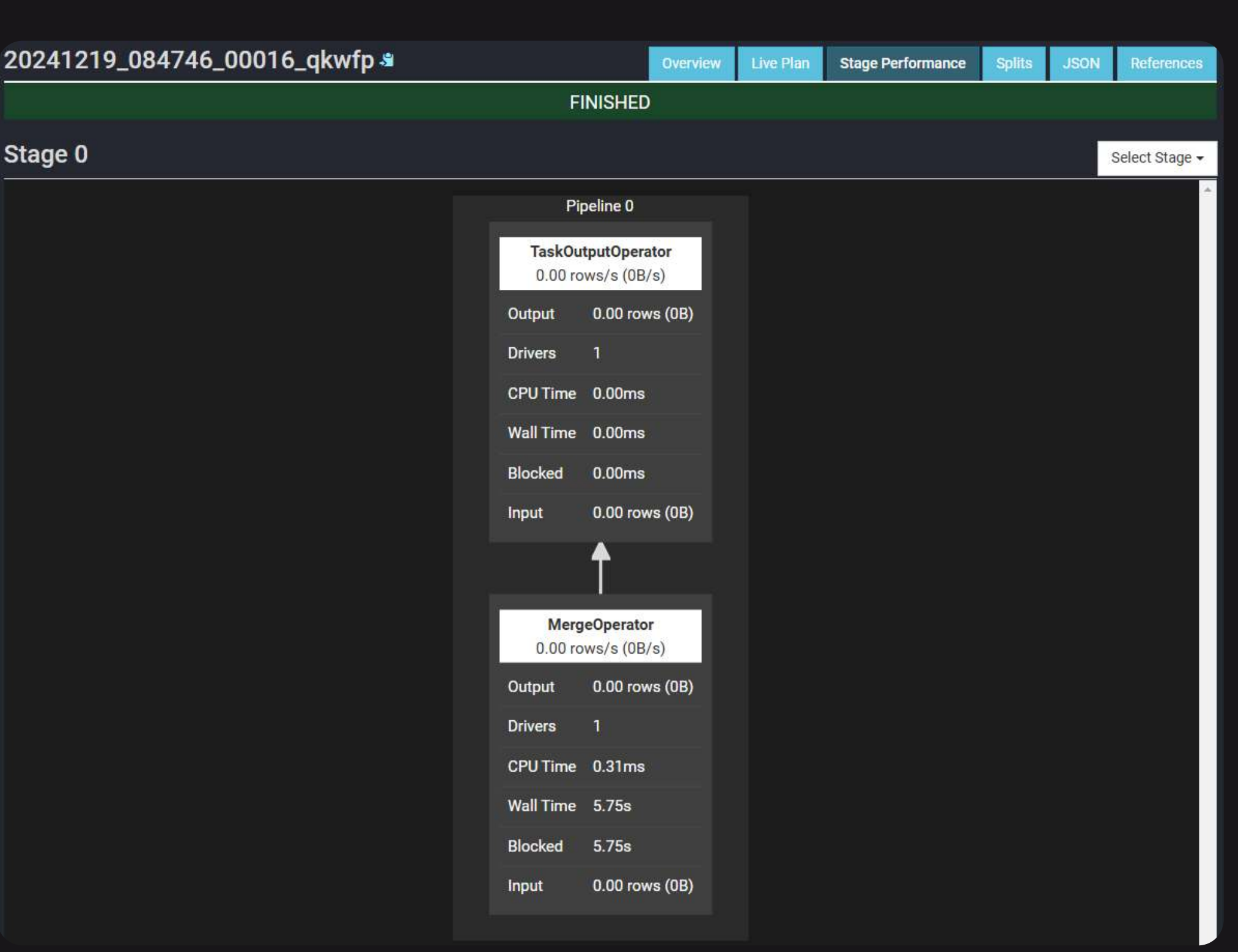


QUERY DETAILS
VERSION
3.3.6_ARENADATA1-126-G07BA16E-DIRTY
ENVIRONMENT
ADH
UPTIME
1.17h
Log Out
20241219_084746_00016_qkwfp
Overview
Live Plan
Stage Performance
Splits
JSON
References
FINISHED
Scroll to zoom. Click stage to view additional statistics

The Live plan tab of the Query details page

Stage performance

On the ***Stage performance*** tab, you can see the futher breakdown of the selected stage of the query execution. You can select the stage to inspect in the top right corner of the screen (the ***Select stage*** list).



20241219_084746_00016_qkwfp
Overview
Live Plan
Stage Performance
Splits
JSON
References
FINISHED
Stage 0
Select Stage
Pipeline 0

TaskOutputOperator
0.00 rows/s (0B/s)

Output	0.00 rows (0B)
Drivers	1
CPU Time	0.00ms
Wall Time	0.00ms
Blocked	0.00ms
Input	0.00 rows (0B)

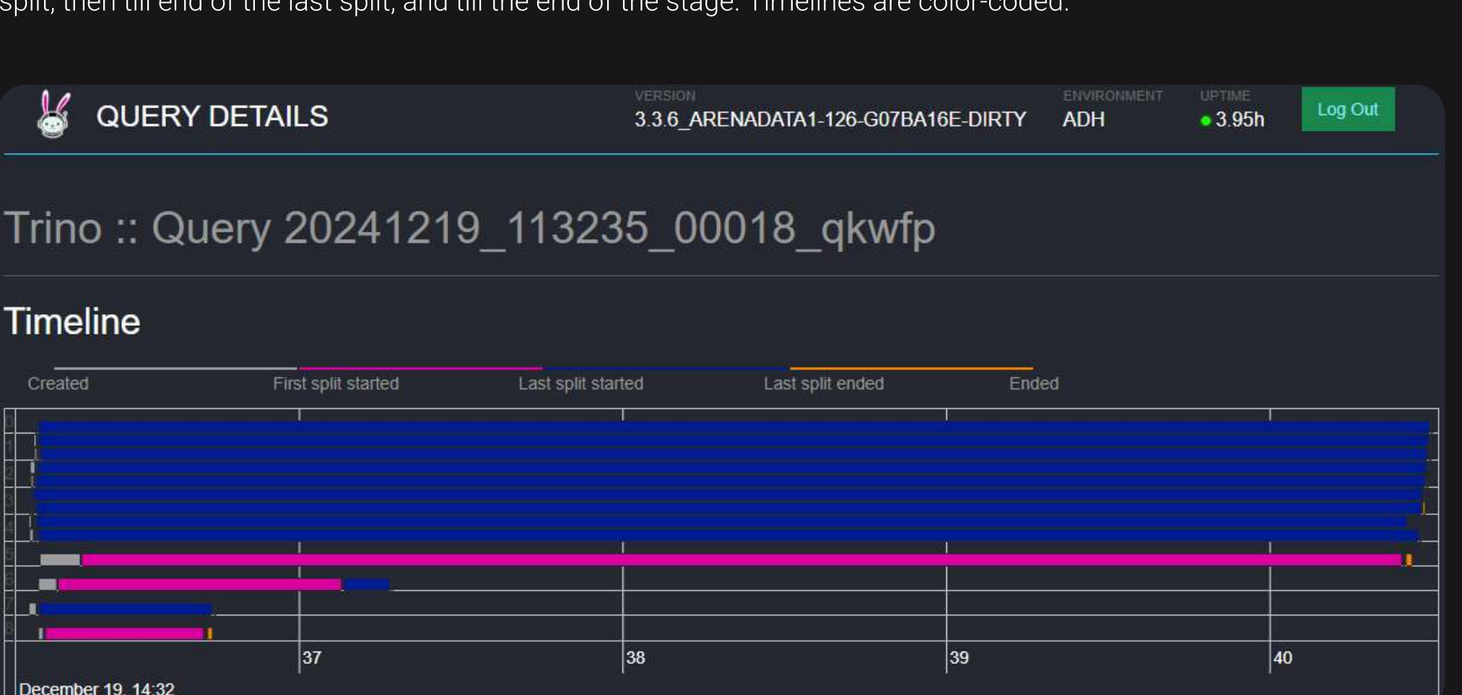
MergeOperator
0.00 rows/s (0B/s)

Output	0.00 rows (0B)
Drivers	1
CPU Time	0.31ms
Wall Time	5.75s
Blocked	5.75s
Input	0.00 rows (0B)

The Stage performance tab of the Query details page

Splits

On the ***Splits*** tab, you can see the splits timelines for each stage of the query execution. Each timeline displays how much time passed since the stage creation till the start of the first split, then till the start of the last split, then till end of the last split, and till the end of the stage. Timelines are color-coded.

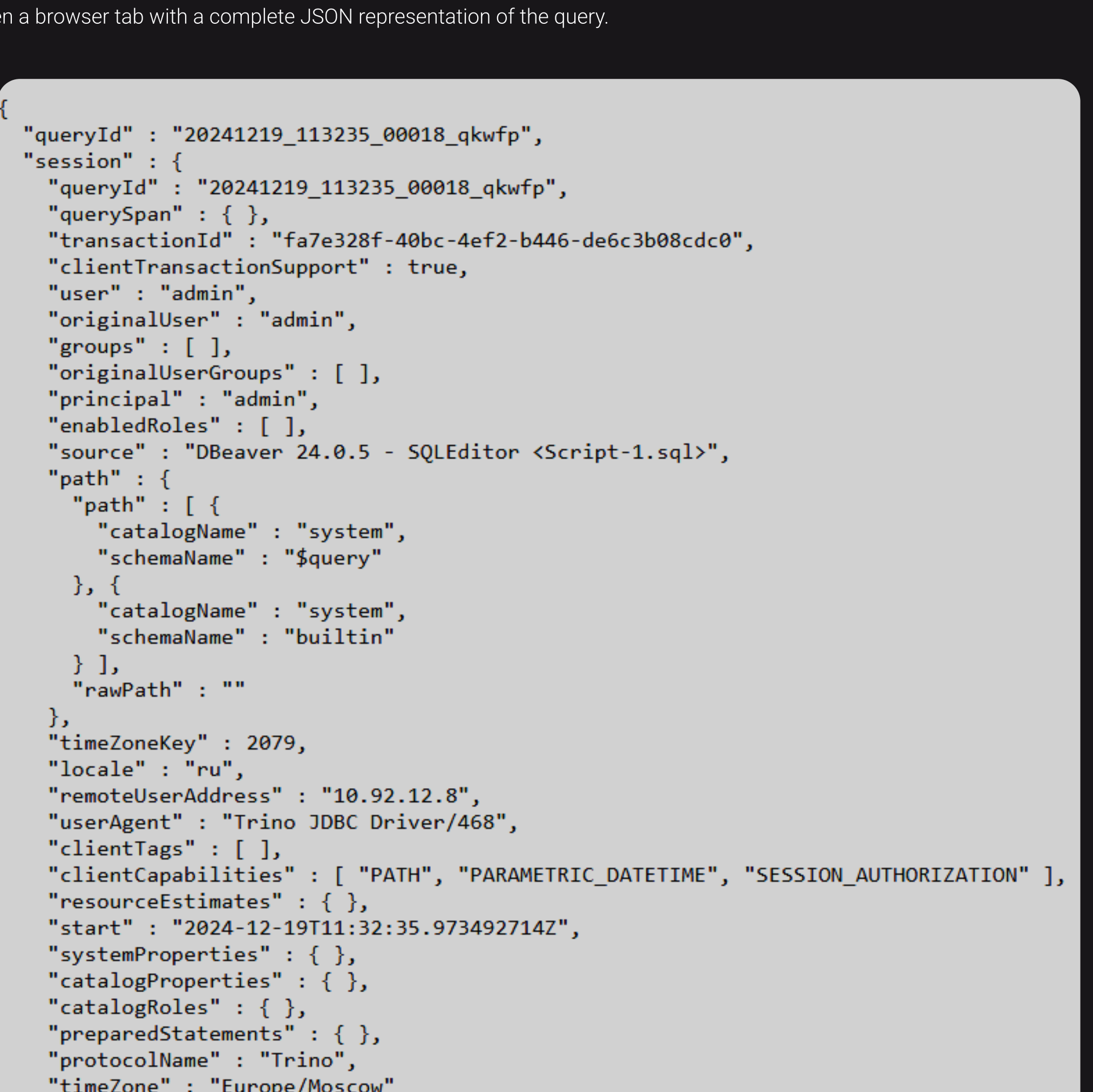


QUERY DETAILS
VERSION
3.3.6_ARENADATA1-126-G07BA16E-DIRTY
ENVIRONMENT
ADH
UPTIME
3.95h
Log Out
Trino :: Query 20241219_113235_00018_qkwfp
Timeline
Created
First split started
Last split started
Last split ended
Ended
37
38
39
40
December 19, 14:32

The Splits tab of the Query details page

JSON

Clicking on the ***JSON*** tab will open a browser tab with a complete JSON representation of the query.

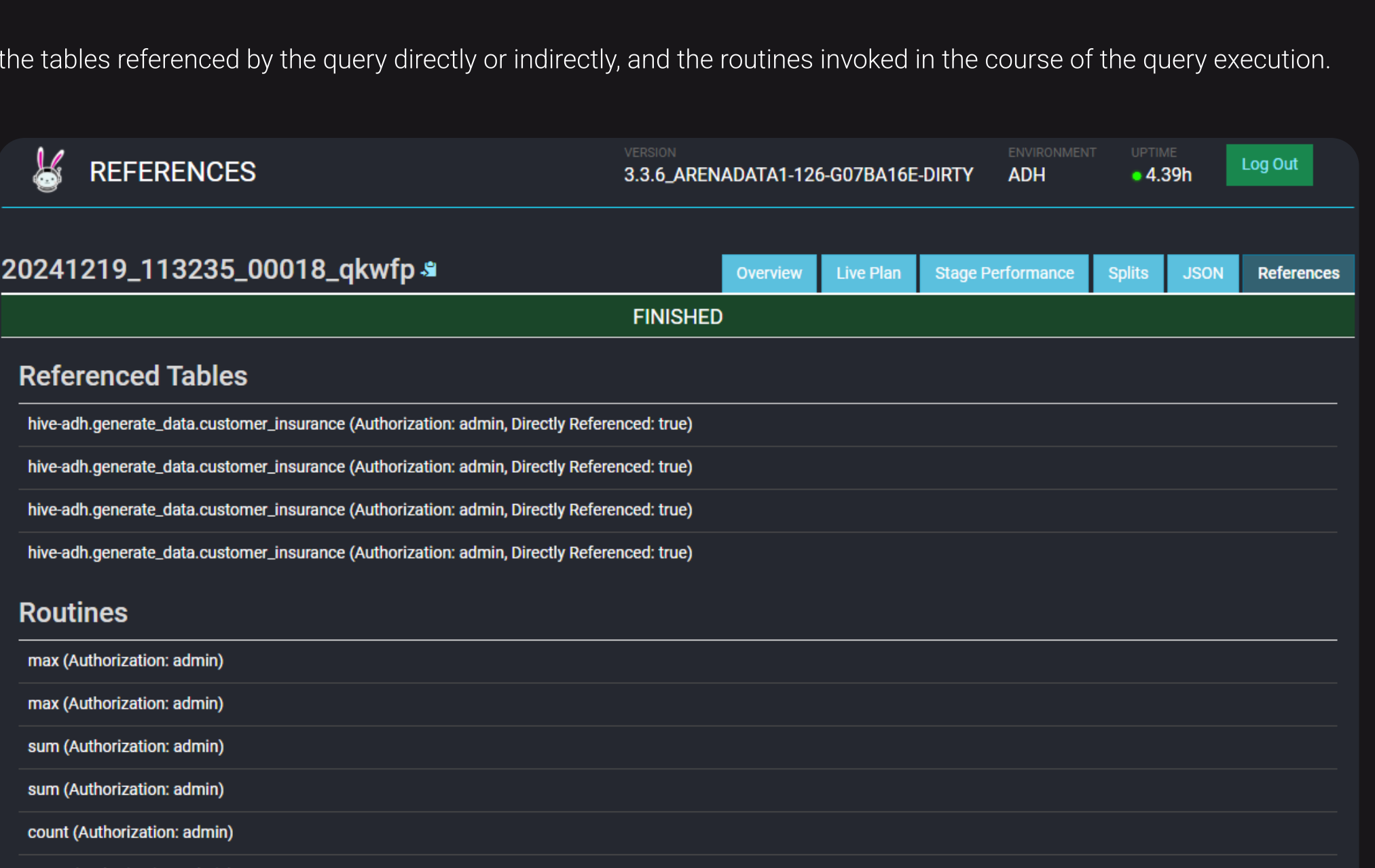


```
{
  "queryId" : "20241219_113235_00018_qkwfp",
  "session" : {
    "queryId" : "20241219_113235_00018_qkwfp",
    "querySpan" : { },
    "transactionId" : "fa7e328f-40bc-4ef2-b446-de6c3b08cdc0",
    "clientTransactionSupport" : true,
    "user" : "admin",
    "originalUser" : "admin",
    "groups" : [ ],
    "originalUserGroups" : [ ],
    "principal" : "admin",
    "enabledRoles" : [ ],
    "source" : "DBeaver 24.0.5 - SQLEditor <Script-1.sql>",
    "path" : {
      "path" : [ {
        "catalogName" : "system",
        "schemaName" : "$query"
      }, {
        "catalogName" : "system",
        "schemaName" : "builtin"
      } ],
      "rawPath" : ""
    },
    "timeZoneKey" : 2079,
    "locale" : "ru",
    "remoteUserAddress" : "10.92.12.8",
    "userAgent" : "Trino JDBC Driver/468",
    "clientTags" : [ ],
    "clientCapabilities" : [ "PATH", "PARAMETRIC_DATETIME", "SESSION_AUTHORIZATION" ],
    "resourceEstimates" : { },
    "start" : "2024-12-19T11:32:35.973492714Z",
    "systemProperties" : { },
    "catalogProperties" : { },
    "catalogRoles" : { },
    "preparedStatements" : { },
    "protocolName" : "Trino",
    "timeZone" : "Europe/Moscow"
  },
```

Excerpt of the JSON representation of the query

References

The ***References*** tab enumerates the tables referenced by the query directly or indirectly, and the routines invoked in the course of the query execution.



REFERENCES
VERSION
3.3.6_ARENADATA1-126-G07BA16E-DIRTY
ENVIRONMENT
ADH
UPTIME
4.39h
Log Out
20241219_113235_00018_qkwfp
Overview
Live Plan
Stage Performance
Splits
JSON
References
FINISHED

Referenced Tables

- hive-adh.generate_data.customer_insurance (Authorization: admin, Directly Referenced: true)
- hive-adh.generate_data.customer_insurance (Authorization: admin, Directly Referenced: true)
- hive-adh.generate_data.customer_insurance (Authorization: admin, Directly Referenced: true)
- hive-adh.generate_data.customer_insurance (Authorization: admin, Directly Referenced: true)

Routines

- max (Authorization: admin)
- max (Authorization: admin)
- sum (Authorization: admin)
- sum (Authorization: admin)
- count (Authorization: admin)
- count (Authorization: admin)
- count (Authorization: admin)

The References tab of the Query details page

Catalog management in Trino

Alpashkin Konstantin

Contents

To Table of Contents

Catalog configuration

View available catalogs

Preconfigured catalogs

Catalog management via ADCM

Catalog management modes

Catalog management properties

A Trino catalog is a collection of configuration properties that represent a single data source (for example, a database) to be accessed by the Trino engine. Every catalog configuration specifies a connector for accessing the underlying data source as well as other details required to establish the connection.

With Trino catalogs, you can refer to the underlying data source in your SQL queries by using a catalog name. For example:

```
SELECT * FROM "hive-adh"."default"."transactions";
```

Where:

- `hive-adh` is the name of the built-in Hive catalog.
- `default` is the name of the schema, used to group tables, views, etc. For example, a schema in Trino maps to a Hive database. For other data sources, the schema can map to a different entity.
- `transactions` is the table to fetch data from.

Although the example above refers to Hive, catalogs can use any other connector to work with many data storage systems. With catalogs, the underlying data store is therefore abstracted for Trino clients.

You can use several catalogs within a single SQL statement to retrieve information from different data sources with one query. For example, the following SQL makes Trino fetch data from two independent data sources — a MySQL database and a Hive warehouse. After getting the result sets from both sources using corresponding connectors, Trino applies a `JOIN` operation and returns the final result set to the client.

```
SELECT txns.*, ords.*
FROM "mysql_test_catalog"."demo_schema"."transactions" txns
JOIN "hive-adh"."demo_schema"."orders" ords
ON txns.acc_id = ords.account_id;
```

Catalog configuration

Trino catalogs are defined using *<catalog_name>.properties* files stored in the Trino catalog directory. By default, this directory is located at */etc/trino/conf/catalog/* on the hosts with Trino components. To change the default catalog directory, use the ***catalog.config-dir*** setting in ADCM (***Clusters → <clusterName> → Services → Trino → Components → Trino Coordinator → node.properties***). The example of */etc/trino/conf/catalog/* content is below:

```
$ ls -l /etc/trino/conf/catalog
```

```
-rw-r--r--. 1 trino hadoop 433 Dec 19 14:34 hive-adh.properties
-rw-r--r--. 1 trino hadoop 280 Dec 20 10:21 iceberg-adh.properties
```

The *.properties* catalog configuration files include the following information:

- The connector used by the catalog (`connector.name`). This is a mandatory property in a *<catalog_name>.properties* file.
- Connectivity details to access the data source, such as connection URLs, credentials, etc.
- Security parameters.
- Other properties required for interaction with the data source. For example, data formats, compression codecs, etc.

Below is an example of the built-in Hive catalog configuration (*hive-adh.properties*), which is used by the Trino service to work with Hive:

```
connector.name=hive
fs.hadoop.enabled=True
hive.compression-codec=SNAPPY
hive.config.resources=/etc/hadoop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml
hive.hdfs.impersonation.enabled=False
hive.metastore.thrift.client.ssl.enabled=False
hive.metastore.thrift.impersonation.enabled=True
hive.metastore.uri=thrift://ka-adh-3.ru-central1.internal:9083
hive.non-managed-table-writes-enabled=True
hive.storage-format=PARQUET
```

The name of the *<catalog_name>.properties* file determines the name of the catalog. For example, *hive-adh.properties* stores settings for the catalog named `hive-adh`.

View available catalogs

To get a list of catalogs available in the Trino service, use the command:

```
SHOW CATALOGS;
```

Sample output:

```
Catalog
---------
hive-adh
iceberg-adh
test-postgresql-catalog
system
(4 rows)
```

If you need to get all catalogs that use a specific connector, submit the following query to Trino:

```
SELECT *
FROM system.metadata.catalogs
WHERE connector_name='hive';
```

The sample result:

```
catalog_name|connector_id|connector_name|
------------+------------+--------------+
hive-adh    |hive-adh    |hive          |
```

Preconfigured catalogs

The Trino service comes with the following preconfigured catalogs out-of-the-box:

- A Hive catalog named `hive-adh`. Used for working with the Hive service deployed in the ADH cluster.
- An Iceberg catalog named `iceberg-adh`. Used for working with Iceberg tables stored in the ADH cluster.

If Hive is installed in the ADH cluster, the configurations for the built-in catalogs are generated automatically using up-to-date values for accessing your Hive service. You can edit the configurations of the built-in Hive/Iceberg catalogs in ADCM. For this, use the ***Hive configuration*** and ***Iceberg configuration*** sections in ADCM (***Clusters → <clusterName> → Services → Trino → Primary configuration***).

> **TIP**
>
> To overwrite configuration properties for the default Hive/Iceberg catalogs, use ***Custom Hive configuration*** and ***Custom Iceberg configuration*** fields respectively.

Catalog management via ADCM

In Trino, adding a new catalog assumes creating a *<catalog_name>.properties* file in the Trino catalog directory (*/etc/trino/conf/catalog/*) on the hosts of a Trino cluster. For this, ADCM provides a convenient UI for adding new catalogs instead of manual manipulations with *.properties* files.

The following steps show how to create a sample Trino catalog that interacts with Arenadata Postgres (ADPG) through the PostgreSQL connector:

1. In ADCM, go to ***Clusters → <clusterName> → Services → Trino → Primary configuration*** and enable the ***Custom catalogs*** option.
2. Expand ***Custom catalogs*** and specify the properties listed in the table.

catalog_name	key	value
test_catalog_adpg	connector.name	postgresql
	connection-url	jdbc:postgresql://<ADPG_master_host>:5432
	connection-user	postgres
	connection-password	postgres

3. Save the configuration and restart the Trino service. During the restart, ADCM creates *test_catalog_adpg.properties* files on all ADH hosts with Trino components. Verify that the *.properties* files have been created:

   ```
   $ ls /etc/trino/conf/catalog/
   ```

   The output:

   ```
   hive-adh.properties  iceberg-adh.properties  test_catalog_adpg.properties
   ```

4. Connect to Trino (for example, via Trino CLI or an SQL client, like DBeaver) and query available catalogs:

   ```
   SHOW CATALOGS;
   ```

   The result set includes the new catalog:

   ```
   Catalog          |
   -----------------+
   hive-adh         |
   iceberg-adh      |
   system           |
   test_catalog_adpg|
   ```

Now the data on the target ADPG server can be accessed via the newly created catalog. For example:

```
SELECT * FROM "test_catalog_adpg"."public"."book";
```

The result:

```
book_id|title                |author_id|in_stock|
-------+---------------------+---------+--------+
      1|Mrs. Dalloway        |        1|      25|
      2|To the Lighthouse    |        1|      13|
      3|To Kill a Mockingbird|        2|       0|
      4|The Great Gatsby     |        3|       3|
      5|The Lord of the Rings|        4|      45|
```

Catalog management modes

Trino provides several catalog management modes:

- `static`. Reads configuration files and creates catalog entities only on the server startup. Adding a catalog requires a restart of the Trino service.
- `dynamic`. Allows managing catalogs dynamically using the `CREATE CATALOG` and `DROP CATALOG` commands.
- `DBCatalogManager`. Allows managing catalogs dynamically and stores catalog metadata in an ADPG/PostgreSQL database. Allows updating catalog configurations without restarting the Trino service.

To change the catalog management mode, use the ***catalog.management*** option in ADCM (***Clusters → <clusterName> → Services → Trino → Components → Trino Coordinator → Trino catalog management***).

Catalog management properties

The following properties are used for managing catalogs in Trino. By default, the Trino service starts using the default parameter values. To override a configuration parameter, use the ***Custom config.properties*** section of the ***Trino Coordinator*** component in ADCM.

Property name	Description	Default value
catalog.prune.update-interval	Requires `catalog.management=dynamic`. Specifies an interval for pruning dropped catalogs. Dropping a catalog does not interrupt running queries (if any), but makes the catalog unavailable for new queries	5s
catalog.store	Possible values: `file`, `memory`. Requires `catalog.management=dynamic`. If set to `file`, creating and dropping catalogs using SQL commands also creates/removes *<catalog_name>.properties* files. Existing catalog files are also read on the Trino service startup. When set to `memory`, catalogs exist only in memory, and the *.properties* files are ignored on startup	file
catalog.config-dir	Requires `catalog.management=static` or `catalog.store=file`. Specifies the directory with catalog property files	/etc/trino/conf/catalog
catalog.disabled-catalogs	Requires `catalog.management=static` or `catalog.store=file`. Specifies a comma-separated list of catalogs to ignore while reading *<catalog_name>.properties* during the startup	—
catalog.read-only	Requires `catalog.store=file`. If set to `true`, the existing *<catalog_name>.properties* files cannot be removed with the `DROP CATALOG` command, and no new catalog files can be written with identical names using `CREATE CATALOG` operation. As a result, the restart of the Trino service resets all the catalogs to the existing files only	false

Use DBCatalogManager

Konstantin Alpashkin

Contents

To Table of Contents

- Overview
- Enable DBCatalogManager
- Configuration
  - Catalog versioning and cleanup
  - Encryption
  - Migration
- ADPG connection example

Overview

DBCatalogManager is a Trino catalog management implementation that stores catalog configurations in an ADPG/PostgreSQL database rather than in *.properties* files. Keeping catalog metadata in a database has several benefits, including improved collaboration, version control, synchronization, and enhanced security.

The implementation allows you to manage Trino catalogs dynamically using the `CREATE CATALOG` / `DROP CATALOG` commands. It allows updating catalog properties without having to restart the Trino service. DBCatalogManager provides encryption capabilities to store sensitive configuration properties in the encrypted form.

By default, DBCatalogManager is preconfigured to work with the ADPG service of an ADH cluster. DBCatalogManager can connect to the ADPG service out-of-the-box, without any configurations; it also can initialize all the necessary DB objects (database, user, role) on the ADPG side. For more details on using DBCatalogManager with the ADPG service, see ADPG connection example.

Enable DBCatalogManager

You can enable DBCatalogManager using ADCM. For this, select the `DB` value for the ***catalog.management*** parameter (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Trino*** → ***Components*** → ***Trino Coordinator*** → ***Trino catalog management***) and restart the Trino service.

> TIP
>
> When you enable DBCatalogManager and restart the Trino service afterwards, ADCM asks whether to initialize necessary DB entities (database, user, role) in ADPG automatically.

Once enabled, you can create new catalogs using the `CREATE CATALOG` command, and Trino will store catalog configurations in the database specified by the ***Postgres JDBC URL*** parameter (by default, the DB name is `catalog`).

DBCatalogManager stores catalog configurations in tables belonging to the `trino` schema. For example, to view these tables using psql, use the commands:

```
\c catalog
\dt trino.*
```

Sample output:

```
                   List of relations
 Schema |             Name             | Type  |  Owner
--------+------------------------------+-------+---------
 trino  | catalog                      | table | catalog
 trino  | catalog_property             | table | catalog
 trino  | catalog_scheduler_job_launch | table | catalog
```

Configuration

ADCM provides a list of settings for tuning the DBCatalogManager behavior. Their description is below.

Trino catalog management

Parameter	Description	Default value
catalog.management	Catalog manager type	static
Postgres JDBC URL	JDBC connection URL of the Postgres catalog store	jdbc:postgresql://{{ groups['adpg.adpg'][0] | d(omit) }}:5432/catalog
Postgres username	Postgres catalog store JDBC connection username	catalog
Postgres password	Postgres catalog store JDBC connection password	—
Enable cleanup	Defines whether the cleanup of the old version catalogs is enabled	true
Encryption password	Password for the properties encryption	—
RegEx pattern	Regular expression pattern for defining properties to be encrypted	—
Cleanup old version catalogs	Old version catalogs cleanup interval. Minimal value is `10d`	30d
Time to live of old version catalogs	Time-to-live (TTL) of old version catalogs. Minimal value is `1d`	365d

Catalog versioning and cleanup

Running `DROP CATALOG` with DBCatalogManager does not remove the catalog data immediately but marks the catalog as inactive, so that subsequent `SHOW CATALOGS` operations will not include the catalog in the results.

Such inactive catalogs can be retained in the database for a configurable period of time specified by the ***Time to live of old version catalogs*** ADCM property. After that, catalogs exceeding the time-to-live value get removed by a scheduled job running at regular intervals and specified by the ***Cleanup old version catalogs*** property.

Encryption

DBCatalogManager allows encrypting specific catalog configuration fields before writing to the database. To specify the fields, use the ***RegEx pattern*** property in ADCM. Its value should be a regular expression matching one or more configuration property keys that have to be encrypted. For example, the following expression can be used to encrypt passwords:

```
.*password
```

Migration

You can migrate existing catalog configurations stored in *.properties* files to ADPG. For this, use the command:

```
MIGRATE CATALOG [IF NOT EXISTS] ALL | <catalog name>
```

An example of migrating a single catalog and all available catalogs is below:

```
MIGRATE CATALOG IF NOT EXISTS test_trino_catalog
MIGRATE CATALOG IF NOT EXISTS ALL
```

ADPG connection example

The following steps show how to use DBCatalogManager with the ADPG service running in the same ADH cluster. The example assumes that both Trino and ADPG services use default settings, which allows establishing the connection without any manipulations on the ADPG side. If you need to use a different DB user, password, or database for connecting to ADPG, specify them using the corresponding settings.

1. In ADCM, enable DBCatalogManager using the ***catalog.management*** parameter (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Trino*** → ***Components*** → ***Trino Coordinator*** → ***Trino catalog management***) and restart the Trino service. When running the ***Restart*** action, ensure that the ***Init DB credentials*** option is selected.

2. Create a new Trino catalog. For this, connect to Trino Coordinator and run the command:

   ```
   CREATE CATALOG test_trino_catalog USING memory
   WITH ("memory.max-data-per-node"='128MB');
   ```

3. Ensure that the catalog has been created:

   ```
   SHOW CATALOGS;
   ```

   The output:

   ```
   information_schema
   test_trino_catalog
   ```

4. Check the ADPG database for new writes. For example, by using psql, query the tables belonging to the `trino` schema:

   ```
   \c catalog
   SELECT * FROM trino.catalog;
   ```

   Sample output:

   ```
    id |        name        | connector_name |         date_from          | date_to | is_active
   ----+--------------------+----------------+----------------------------+---------+-----------
     1 | test_trino_catalog | memory         | 2025-06-03 21:12:11.193509 |         | t
   ```

5. To migrate an existing catalog stored as a *.properties* file (for example, */etc/trino/catalogs/hive-adh.properties*), use the command:

   ```
   MIGRATE CATALOG "hive-adh"
   ```

   Ensure that the catalog becomes available:

   ```
   SHOW CATALOGS;
   ```

   The output:

   ```
   Catalog           |
   ------------------+
   hive-adh          |
   system            |
   test_trino_catalog|
   ```

   Check the new record in ADPG:

   ```
   \c catalog
   SELECT * FROM trino.catalog;
   ```

   The output:

   ```
    id |        name        | connector_name |         date_from          | date_to | is_active
   ----+--------------------+----------------+----------------------------+---------+-----------
     1 | test_trino_catalog | memory         | 2025-06-03 21:12:11.193509 |         | t
     2 | hive-adh           | hive           | 2025-06-03 22:10:43.222419 |         | t
   ```

Manage fault-tolerant execution in Trino

Konstantin Alpashkin

Contents

To Table of Contents

Enable FTE

Exchange manager and exchange data

FTE settings

- Spool data to HDFS, Ozone, and local FS

By default, when a Trino node runs out of resources while executing a query or fails due to some other reason, the entire query fails and has to be re-submitted manually. Typically, the longer a query execution lasts, the more likely it is to fail.

The Trino service provides a fault-tolerant execution (FTE) mechanism that mitigates query failures by retrying failed queries and their subtasks. If FTE is enabled and a query fails, Trino persists intermediate data in a file system, allowing other Trino workers to perform a retry using this data.

NOTE

- The support for FTE is connector-dependent. The list of Trino connectors that support FTE is available in the Trino documentation.
- The Trino FTE mechanism does not apply to broken queries or other user errors. For example, Trino does not retry a query that has failed due to invalid SQL syntax or incompatible column types.

Enable FTE

By default, the FTE mechanism is disabled for the Trino service. You can enable FTE using ADCM (***Clusters** → **<clusterName>** → **Services** → **Trino** → **Primary Configuration** → **Fault-tolerant execution***). Particularly, the retry-policy parameter should be specified. The parameter accepts the following values:

- `NONE` — disables FTE (default).
- `QUERY` — if a Trino worker fails, Trino retries the entire query. This policy is recommended when the majority of Trino requests are small queries.
- `TASK` — if a Trino worker fails, Trino retries individual query tasks. This policy is best suited for retrying large batch queries, however, it can result in higher latency for big number of short-running queries.

NOTE

Selecting the `TASK` retry policy automatically adjusts some relevant Trino configurations to follow best practices for a fault-tolerant Trino cluster. For the `TASK` retry policy, it is strongly recommended to set the `task.low-memory-killer.policy=total-reservation-on-blocked-nodes` parameter; otherwise, queries may need to be killed manually if Trino worker components run out of memory.

Exchange manager and exchange data

In Trino, Exchange manager is a component responsible for transferring exchange data between Trino workers while executing a distributed query. When FTE is enabled, this component becomes responsible for spooling exchange data to a file system like HDFS, Ozone, etc. When activating FTE, you can select an Exchange manager implementation to save intermediate query data in a specific file system.

FTE settings

The Trino service provides several settings groups in ADCM, which are used for tuning the FTE behavior. These are:

- Fault-tolerant execution. The settings define the retry behavior, including retry policies, intervals, encryption, etc.
- Exchange manager settings. The settings are used to control the intermediate spooling data generated by an Exchange manager (storage type and location, permissions, buffers, etc.).

Spool data to HDFS, Ozone, and local FS

By default, the Trino service provides two Exchange manager implementations that can spool data to HDFS, Ozone, and to a local file system.

When you enable FTE, the Trino service automatically detects whether HDFS or Ozone is installed in the ADH cluster, and updates all the required settings to work with the current storage type.

To store exchange data to HDFS, set the `exchange-manager.name=hdfs` parameter and specify the path for storing spooling data by using `exchange.base-directories` with the `hdfs:///` schema in the path.

To store exchange data in Ozone, set the `exchange-manager.name=hdfs` parameter and specify the storage path using the `ofs:///` schema.

NOTE

Specifying both HDFS and Ozone paths in `exchange.base-directories` is not allowed.

To store exchange data in the local file system, set the `exchange-manager.name=local` parameter and specify the path for storing spooling data using `exchange.base-directories` (use the `file:///` scheme).

The Trino service can automatically generate directories for spooling data. Using FTE settings, you can specify the location, directory permissions, etc.

Credentials propagation in Trino

Elena Kostyuchenko

Contents

To Table of Contents

Overview
Enable credentials propagation
Example

Overview

Credentials propagation allows Trino to authenticate connectors — such as ADB, PostgreSQL, ClickHouse, and others, on behalf of the currently logged-in user. This ensures secure, user-specific access to data sources.

This article explains how to enable and configure credentials propagation for Trino in Arenadata clusters.

> **NOTE**
>
> This feature works for JDBC-like connectors and when using HTTP authentication with password (the `http-server.authentication.type=PASSWORD` property).

Enable credentials propagation

When `arenadata.http-server.authentication.password.populate-extra-credentials` is set to `true`, Trino automatically adds the username and password from the current session to the extra credentials. The configured connector reads these credentials and authenticates the query on behalf of the user.

To enable credentials propagation via ADCM:

1. In ADCM, select your ADH cluster.
2. Navigate to ***Services*** → ***Trino*** → ***Components*** → ***Trino Coordinator/Trino Worker*** and toggle ***Show advanced***.
3. Open the ***Custom node.properties*** section and click ***Add property***.
4. Enter `arenadata.http-server.authentication.password.populate-extra-credentials` as the field name and `true` as the value.
5. Save the configuration by clicking ***Save*** and restart a component by selecting ***Actions*** → ***Restart***.

For each connector that should use the propagated credentials, add the following properties to the connector configuration file:

```
user-credential-name=arenadata.username
password-credential-name=arenadata.password
```

For more information on how to configure connectors, see Overview of Trino connectors.

Example

To verify that credentials propagation works correctly:

1. Connect to Trino using CLI. The example command for connection:

```
$ ./trino https://<coordinator_host>:18188 --user=<username> --password
```

2. Enter the password when prompted.
3. Run a query against a connector that uses the propagated credentials:

```
SHOW TABLES FROM adb.default;
```

The connector will authenticate using the provided user credentials.

Trino Coordinator high availability

Konstantin Alpashkin

Contents

To Table of Contents

Haproxy Trino component

Haproxy Trino configuration

Logging

Haproxy Trino statistics

Kerberos tips and limitations

Trino service supports a high availability (HA) mode that allows running multiple Trino Coordinator components. If one Trino Coordinator becomes unavailable, the standby Trino Coordinators kick in without interrupting the entire Trino cluster operation.

Haproxy Trino component

Switching between Trino Coordinators is done with the help of the Haproxy Trino component, which acts as a reverse proxy between Trino clients and Trino Coordinators in ADH. This component is based on the HAProxy project, which is used for ensuring high availability, load balancing, and routing HTTP/TCP traffic among servers.

IMPORTANT

- To connect to the Trino service running in the HA mode, the Trino client (Trino CLI, JDBC, or a browser) must connect to the Haproxy Trino component, but not directly to Trino Coordinators. The up-to-date web UI link and JDBC connection string are available in ADCM (***Clusters → <ADH_cluster> → Services → Trino → Info***).
- The Haproxy Trino and Trino Coordinator components must be installed on different hosts to avoid port conflicts.

An ADH cluster may contain one or more Haproxy Trino components. Adding several Haproxy Trino components can be used for implementing a full-fledged high availability mode at the network level.

Haproxy Trino configuration

When you use the Trino service in the HA mode, ADCM automatically configures the Haproxy Trino component to route incoming requests to Trino Coordinators, so the service requires no manual configuration. Additionally, you can tune the Haproxy Trino behavior as described below.

The main settings file for Haproxy Trino is */etc/adh-haproxy/conf/haproxy-trino.cfg*. It is a HAProxy configuration file located on the ADH host with the Haproxy Trino component installed.

Sample haproxy-trino.cfg

```
#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global
    log /dev/log    local0
    log 127.0.0.1:514  local0

    chroot      /var/lib/adh-haproxy/trino
    maxconn     1024
    user        haproxy
    group       haproxy
    daemon

    ssl-server-verify none

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s

listen stats
    bind *:7001
    stats enable
    stats uri /

frontend trino_http_in
    bind *:18188
    default_backend trino_http_servers

backend trino_http_servers
    balance first

    server trino-coordinator0 ka-adh-1.ru-central1.internal:18188 check inter 1s fall 2 rise 5
    server trino-coordinator1 ka-adh-3.ru-central1.internal:18188 check inter 1s fall 2 rise 5 backup
```

This file should not be edited manually — instead, to change the Haproxy Trino configuration, use the Jinja template available in ADCM (***Clusters → <ADHclusterName> → Services → Trino → Components → Haproxy Trino → Configuration***). During the Trino service startup, *haproxy-trino.cfg* is generated from this template.

Below is the default *haproxy-trino.cfg* template with major settings highlighted.

```
{% set ssl_sert = '' %}
{%- if trino_ssl_enable -%}
    {% set ssl_sert = haproxy_conf_ssl_vars %}
{% endif %}
#---------------------------------------------------------------------
# Global settings
#---------------------------------------------------------------------
global ①
    log /dev/log    local0
    log 127.0.0.1:514  local0

    chroot      /var/lib/adh-haproxy/trino
    maxconn     1024
    user        haproxy
    group       haproxy
    daemon

    ssl-server-verify none

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults ②
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    timeout http-request    10s
    timeout queue           1m
    timeout connect         10s
    timeout client          1m
    timeout server          1m
    timeout http-keep-alive 10s
    timeout check           10s

listen stats ③
    bind *:7001 {{ ssl_sert }}
    stats enable
    stats uri /

frontend trino_http_in ④
    bind *:{{ roles_trino_vars_component_ports['haproxy']['web'] }} {{ ssl_sert }}
    default_backend trino_http_servers

backend trino_http_servers ⑤
    balance first

{% for host in haproxy_conf_hosts %}
    {{ host }}
{% endfor %}
```

① Global HAProxy parameters, such as logs destination, root directory definition, certificate verification, etc.

② Default HAProxy parameters, such as operation mode, logging, and various timeouts.

③ Enables HAProxy real-time statistics and monitoring. Defines a port and URI for the statistics endpoint (default to *http://<Haproxy_Trino_host>:7001/*).

④ Describes a set of listening sockets to accept incoming connections.

⑤ Lists Trino Coordinator servers to which Haproxy Trino will forward incoming requests. In this template, the `{% for host in … %}` loop produces a list of Trino Coordinator servers available in the ADH cluster. `balance first` indicates that the first server is used as an active (main) Trino Coordinator, other servers are used as backup Coordinators if the first one becomes unavailable.

For more details on HAProxy configuration, see the HAProxy reference.

Logging

By default, the Haproxy Trino component logs its activity and stores logs at */var/log/adh-haproxy/*. The logging settings are specified in *haproxy-trino.cfg* and the default configuration ensures dual logging, namely:

```
global
    log /dev/log    local0 ①
    log 127.0.0.1:514  local0 ②
```

① Sends logs to the local syslog socket (*/dev/log*) to be processed by the system's syslog daemon.

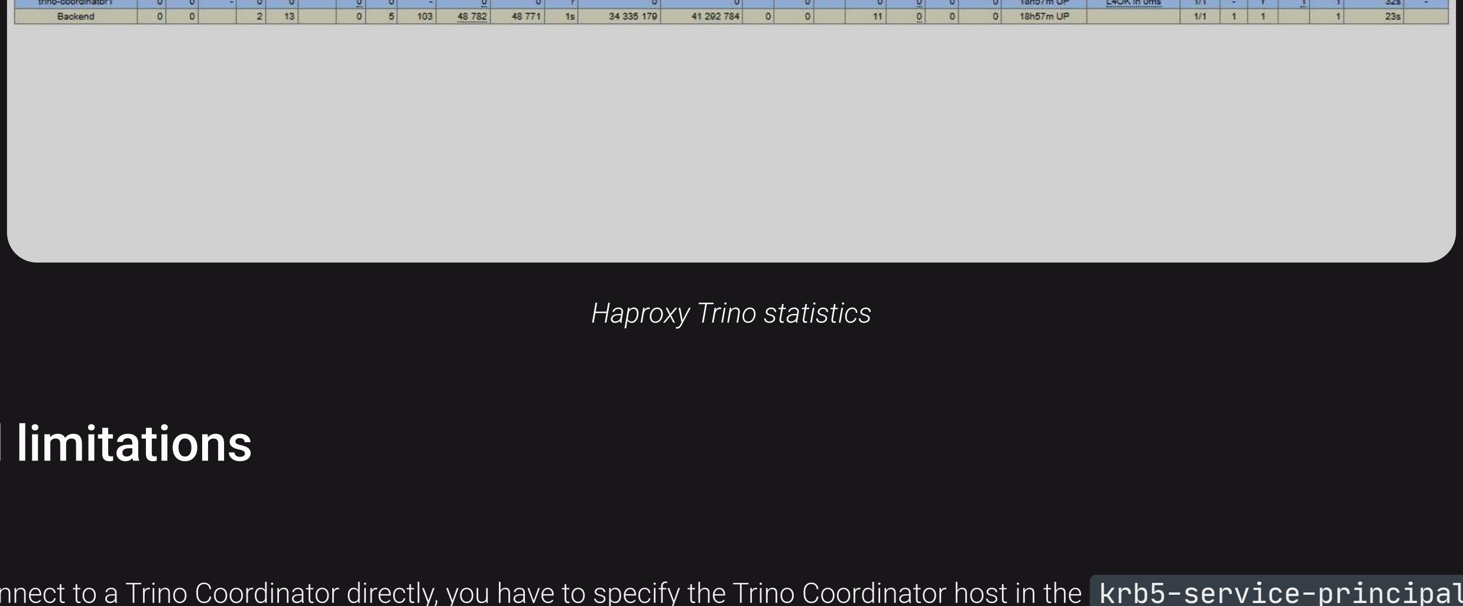
② Sends logs to a syslog daemon like rsyslog or syslog-ng listening to the `514` port on localhost. By default, these daemons are configured to write logs to */var/log/adh-haproxy/*.

TIP

The Haproxy Trino component has corresponding rsyslog/syslog-ng settings in ADCM (***/etc/syslog-ng/conf.d/haproxy-trino.conf***, ***/etc/rsyslog.d/haproxy-trino.conf***) where you can configure logs templates, filters, log files location, etc.

Haproxy Trino statistics

The Haproxy Trino component provides statistics about its operation. The statistics collection is enabled by default (the `listen stats` block in *haproxy-trino.cfg*) and is available at the following endpoint: *http://<Haproxy_Trino_host>:7001/*.



HAProxy version 3.2.0, released 2025/05/28

Statistics Report for pid 2081

> General process information

Haproxy Trino statistics

Kerberos tips and limitations

- In a kerberized cluster, to connect to a Trino Coordinator directly, you have to specify the Trino Coordinator host in the `krb5-service-principal-pattern` parameter for the Trino console client, for example:

  ```
  krb5-service-principal-pattern=HTTP@ka-adh-1.ru-central1.internal
  ```

  For JDBC, use the parameter:

  ```
  KerberosServicePrincipalPattern=HTTP@ka-adh-1.ru-central1.internal
  ```

- Direct connection to Trino Coordinator web UI via a browser (Chrome, Firefox) or cURL is not supported.

Overview of Trino connectors

Konstantin Alpashkin

Contents

To Table of Contents

Overview

Supported connectors

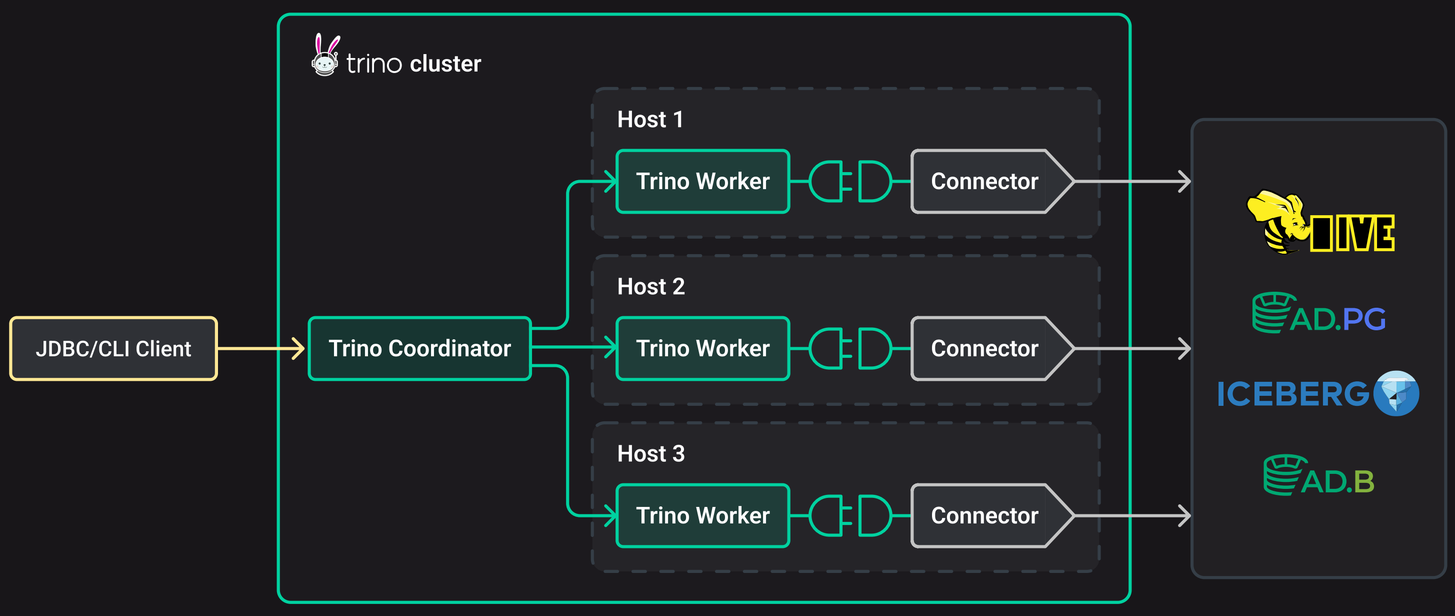
Configure connectors

Create a custom connector

Overview

Trino connectors are Java-based plugins through which Trino can interact with various data sources, such as Hadoop data lakes, Hive warehouses, Apache Iceberg tables, PostgreSQL databases, and many others. Trino accesses data in a data source by using a connector, which is configured in a catalog. Connectors are responsible for managing the communication between Trino and the underlying data source. They also translate ANSI SQL statements into operations understood by the underlying data source. One connector can be used to access one or more data sources of the same type.

At a high level, the connectors' role in Trino interaction is presented in the following diagram.



Trino connectors

Supported connectors

The full list of connectors supported by Trino can be found in Trino documentation.

Configure connectors

In Trino, connectors can be configured using catalog configuration files. The connector to be used for accessing the catalog's data source is specified in the *<catalog_name>.properties* configuration file using the `connector.name` property. For example:

```
connector.name=hive
...
```

NOTE

The same connector can be used by several catalogs. For example, if you have two Hive warehouses with separate Metastore databases, you can configure two catalogs that use the same connector for Hive (`connector.name=hive`) but provide different connection properties for each catalog.

Catalog configuration files may also include auxiliary connector-specific properties for fine-tuning of the interaction with the underlying data source. For example:

```
hive.metastore.thrift.impersonation.enabled=True
iceberg.catalog.type=hive_metastore
iceberg.compression-codec=SNAPPY
iceberg.file-format=PARQUET
...
```

For more information on connector-specific properties, see the corresponding connector section of the Trino documentation.

Create a custom connector

Apart from the connectors list supported by the Trino service out-of-the-box, you can develop your own connectors to access almost any data source. Even if your data source does not utilize concepts like tables, columns, rows, partitions, etc., once you adapt your data source to the API expected by Trino, you can write queries in ANSI SQL and fetch results through the connector.

Technically, new Trino connectors are added as plugins that integrate with the Trino framework to connect to a new data source. Any connector must implement the Trino's service provider interface (SPI), which allows Trino to interact with a data source. Every connector must implement the following parts of the API:

- Operations on retrieving metadata about tables, views, and schemas.
- Operations to produce logical units of data partitioning to allow Trino to parallelize read and write statements.
- Data sources and sinks that convert the source data to/from the in-memory objects expected by the Trino query engine.

After implementing the SPI for a connector, Trino can use standard operations internally to connect to any data source and perform operations on any data source. The particular connector's implementation takes care of the details relevant to the specific features of the underlying data source.

For example, a basic Trino connector that supports reading data from a data source must implement a set of methods, including `listTables()`. The Trino engine invokes this method for all connectors to get a list of available tables in a schema/database. The Trino engine is not aware of how each particular connector gets the tables list. Some connectors may retrieve the list from an information schema, others may query a metastore, and some have to request the list via the API of the data source. It is only the connector that knows how to get the list for the particular data source.

For details and examples on creating custom connectors, see the following sections in the Trino documentation:

- HTTP connector example
- JDBC connector example

Also, you can explore existing Trino connectors source code in GitHub.

Iceberg catalog in Trino

Konstantin Alpashkin

Contents

To Table of Contents

- Enable Iceberg catalog
- Supported table formats
- Data types mapping
- Iceberg catalog configuration properties
- SQL support
  - Schema management
  - DDL examples
  - DML examples

The Trino service comes with a built-in catalog for working with Iceberg tables. This catalog configuration uses the Iceberg connector and out-of-the-box is ready to work with Iceberg tables stored in your ADH cluster. The built-in Iceberg catalog uses Hive Metastore as an Iceberg catalog implementation. That is, to execute a query on an Iceberg table, Trino requests metadata about the table from Hive Metastore.

In the built-in Iceberg catalog, all the configuration properties are auto-generated to ensure access to Hive Metastore and Hive warehouse directories of your ADH cluster. The built-in Iceberg catalog name is `iceberg-adh`. ADH forms this catalog automatically and generates the *iceberg-adh.properties* file at */etc/trino/conf/catalog/* if:

- Your ADH cluster has the Hive service installed. If Hive was installed after Trino, the Trino service must be restarted.
- The Iceberg catalog configuration is active.

Enable Iceberg catalog

By default, the built-in Iceberg catalog is disabled and can be activated using ADCM. For this, toggle the ***Iceberg configuration*** option of the Trino service (***Clusters → <clusterName> → Services → Trino → Configuration***) and then restart the service.

Once enabled, check the catalog availability. For this, connect to Trino Coordinator (for example, via Trino CLI or an SQL client like DBeaver) and run the command:

```
SHOW CATALOGS;
```

The built-in Iceberg catalog appears in the list:

```
Catalog    |
-----------+
iceberg-adh|
system     |
```

TIP

The up-to-date JDBC connection string for connecting to the Trino service is available on the service configuration page in ADCM (***Clusters → <clusterName> → Services → Trino → Info***).

Supported table formats

Trino's Iceberg connector can work with Iceberg tables in the following formats:

- Parquet
- ORC
- Avro

Data types mapping

When reading and writing data to Iceberg tables, Trino automatically converts certain data types to adhere to the Iceberg specification and Avro/ORC/Parquet formats. The mapping rules may vary depending on the direction.

For information on mapping Iceberg types to Parquet/ORC/Avro types, follow the links to the Iceberg spec:

- Iceberg → Avro
- Iceberg → ORC
- Iceberg → Parquet

Iceberg to Trino type mapping

Iceberg type	Trino type
BOOLEAN	BOOLEAN
INT	INTEGER
LONG	BIGINT
FLOAT	REAL
DOUBLE	DOUBLE
DECIMAL(p,s)	DECIMAL(p,s)
DATE	DATE
TIME	TIME(6)
TIMESTAMP	TIMESTAMP(6)
TIMESTAMPTZ	TIMESTAMP(6) WITH TIME ZONE
STRING	VARCHAR
UUID	UUID
BINARY	VARBINARY
FIXED (L)	VARBINARY
STRUCT(...)	ROW(...)
LIST(e)	ARRAY(e)
MAP(k,v)	MAP(k,v)

Trino to Iceberg type mapping

Trino type	Iceberg type
BOOLEAN	BOOLEAN
INTEGER	INT
BIGINT	LONG
REAL	FLOAT
DOUBLE	DOUBLE
DECIMAL(p,s)	DECIMAL(p,s)
DATE	DATE
TIME(6)	TIME
TIMESTAMP(6)	TIMESTAMP
TIMESTAMP(6) WITH TIME ZONE	TIMESTAMPTZ
VARCHAR	STRING
UUID	UUID
VARBINARY	BINARY
ROW(...)	STRUCT(...)
ARRAY(e)	LIST(e)
MAP(k,v)	MAP(k,v)

Iceberg catalog configuration properties

The Trino's Iceberg catalog supports numerous configuration properties for fine-tuning of the interaction with Iceberg tables. The up-to-date configuration properties list is available in Trino documentation. You can specify new or override the default property values using ADCM. For this, use the ***Custom Iceberg configuration*** settings section (***Clusters → <clusterName> → Services → Trino → Configuration***).

SQL support

IMPORTANT

To allow Trino to write to Iceberg tables, the user name specified while connecting to Trino (for example, via Trino CLI or DBeaver) must match the Unix user on the ADH host with sufficient permissions to write to Hive warehouse directories in HDFS. Otherwise, you may observe a similar error:

```
Failed to create external path hdfs://adh/apps/hive/warehouse/trino_demo_schema1.db for database
trino_demo_schema1. This may result in access not being allowed if the StorageBasedAuthorizationProvider is
enabled: null
```

Schema management

A schema in Trino is used to group several tables/views and is equivalent to a database in Hive parlance. Below are basic commands for managing Trino schemas:

```
CREATE SCHEMA "iceberg-adh"."trino_demo_schema_iceberg"; ①
CREATE SCHEMA "iceberg-adh"."trino_demo_schema_iceberg_with_loc"
    WITH (location='/user/hive/warehouse/some_custom_path'); ②
SHOW SCHEMAS;
USE "iceberg-adh"."trino_demo_schema_iceberg"; ③
DROP SCHEMA "trino_demo_schema_iceberg"; ④
```

① Creates a new schema (Hive database). `iceberg-adh` is the name of the built-in Iceberg catalog. Notice the double quotes used for identifiers.

② Creates a schema (Hive database) using a custom HDFS location.

③ Specifies a session schema. All the subsequent SQL commands submitted to Trino within the current session will run on the selected Hive database.

④ Deleting a non-empty schema is prohibited. To delete a schema, you have to remove all nested tables first.

To run Trino queries on arbitrary tables that belong to different schemas, you can specify a catalog, schema, and table name within your SQL command, for example:

```
SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions"
```

TIP

Using Trino UI, you can view and analyze SQL queries submitted to Trino.

DDL examples

Trino supports common DDL operations like `CREATE`, `DROP`, `ALTER`, etc., for managing Iceberg tables and materialized views. Trino supports a set of Iceberg table properties that specify metadata for Iceberg tables. These properties can be applied to an Iceberg table using the `WITH` clause as shown in the examples below. The following snippet provides examples of basic DDL operations using the Iceberg catalog.

```
CREATE TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions" ( ①
    txn_id BIGINT NOT NULL,
    txn_value DOUBLE,
    acc_id BIGINT,
    txn_date DATE,
    comment VARCHAR
);

CREATE TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd" ( ②
    txn_id BIGINT,
    txn_value DOUBLE,
    acc_id BIGINT,
    txn_date DATE,
    comment VARCHAR
) WITH (
    format = 'PARQUET',
    partitioning = ARRAY['acc_id']
);

CREATE TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd_ts" ( ③
    txn_id BIGINT,
    txn_value DOUBLE,
    acc_id BIGINT,
    txn_date DATE,
    comment VARCHAR
) WITH (
    partitioning = ARRAY['month(txn_date)', 'bucket(acc_id, 5)']
);

CREATE TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions_sorted" ( ④
    txn_id BIGINT,
    txn_value DOUBLE,
    acc_id BIGINT,
    txn_date DATE,
    comment VARCHAR
) WITH (
    sorted_by = ARRAY['txn_value']
);

CREATE TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions_orc_as" ⑤
WITH (
    format = 'ORC',
    orc_bloom_filter_columns = ARRAY['acc_id','txn_date'],
    orc_bloom_filter_fpp = 0.1
) AS
    SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions"
    WHERE txn_id < 10;

ALTER TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd" ⑥
SET PROPERTIES partitioning = ARRAY['acc_id', 'txn_date'];

ALTER TABLE "iceberg-adh"."trino_demo_schema_iceberg"."transactions" ⑦
EXECUTE expire_snapshots(retention_threshold => '1d');


CREATE MATERIALIZED VIEW "iceberg-adh"."trino_demo_schema_iceberg"."transactions_mv"  ⑧
WITH (
    partitioning = ARRAY['txn_date']
) AS
```

① Creates an Iceberg table (`table_type=ICEBERG`) in the default Hive warehouse location. Notice that Trino accepts queries in ANSI SQL with corresponding data types, whereas the Iceberg specification may use slightly different data types. For example, when running this query, the Trino's `VARCHAR` data type will be cast to the Iceberg's `STRING`.

② Creates a Parquet-formatted Iceberg table with partitions. Notice the `ARRAY['col1', 'col2']` syntax for defining partitions.

③ Creates a table partitioned by months of `txn_date` and also by hashes of `acc_id` (5 partitions).

④ Creates a table, whose content is stored in Iceberg files sorted by a column (`txn_value`). Storing sorted table data can be beneficial in terms of performance for future read operations.

⑤ Creates an ORC-formatted table using the CTAS (`CREATE TABLE AS`) syntax. Additionally, sets ORC-specific bloom filtering properties.

⑥ Adds more partition columns to a partitioned table using `ALTER TABLE SET`. For more information about Iceberg table properties that can be set via `ALTER TABLE SET`, see Trino documentation.

⑦ Removes all the snapshots, metadata, and data files of an Iceberg table, which are older than the specified retention value (7 days in this example). Running this command periodically allows cleaning up unnecessary historical data files and helps to keep your table metadata small. For information on other Iceberg-specific commands (`remove_orphan_files`, `drop_extended_stats`, etc.) that can be invoked using `ALTER TABLE EXECUTE`, see Trino documentation.

⑧ Creates a materialized view based on results of a `SELECT` query.

For more information on DDL commands supported by the Iceberg catalog in Trino, see Trino documentation.

DML examples

Trino supports common DDL operations like `INSERT`, `UPDATE`, `DELETE`, `MERGE`, etc. The following snippet provides examples of DML operations via the Iceberg catalog.

```
SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd$partitions"; ①

SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd$snapshots"; ②

SELECT * FROM TABLE( ③
    system.table_changes(
      schema_name => "trino_demo_schema_iceberg",
      table_name => "transactions",
      start_snapshot_id => 3196536194725230246,
      end_snapshot_id => 248634242462154958
    )
  );

SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd"
FOR VERSION AS OF 2602443793554441031; ④

SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd" FOR TIMESTAMP AS OF TIMESTAMP '2025-01-17 01:41:38.000
+0300' ⑤

SELECT * FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions" txns_icb
JOIN "hive-adh"."trino_demo_schema_hive"."transactions" txns_hive
ON txns_icb.txn_id = txns_hive.txn_id; ⑥

DELETE FROM "iceberg-adh"."trino_demo_schema_iceberg"."transactions_pttnd"
WHERE acc_id = 1001; ⑦
```

① Retrieves detailed information about the partitions of the given Iceberg table by using the hidden `$partitions` metadata table. Similarly, you can make use of other metadata tables like `$manifests`, `$files`, `$history`, etc. For more information on fetching from Iceberg metadata tables, see Trino documentation.

② Gets detailed information about the snapshots available for the given Iceberg table.

③ Returns row-level changes between two versions of an Iceberg table.

④ A time-travel `SELECT` using a snapshot ID. The state of the table to read data from is determined by an exact snapshot. Such a query ignores all the changes made to the table after the given snapshot was created.

⑤ A time-travel `SELECT` using a timestamp. The state of the table to read data from is determined by a snapshot closest to the provided timestamp. Such a query ignores all the changes made to the table after the given timestamp.

⑥ Selects data from two independent Trino catalogs — an Iceberg catalog (`iceberg-adh`) and a Hive catalog (`hive-adh`). After Trino gets result sets from both catalogs, it applies a `JOIN` operation and returns the final result set to the client. For this sample query to work, both catalogs must be enabled.

⑦ Deletes the entire partition if the `WHERE` clause specifies filters that exactly match a partition.

Hive catalog in Trino

Konstantin Alpashkin

Contents

To Table of Contents

- Enable Hive catalog
- Supported Hive formats
- Hive catalog configuration properties
- SQL support
  - Schema management
  - DDL examples
  - DML examples
- Performance optimization

The Trino service comes with a built-in catalog configuration for working with Hive. This catalog uses the Hive connector and out-of-the-box is ready to interact with the Hive service available in your ADH cluster.

> The architecture of the Trino connector for Hive differs from DBMS-oriented and other connector implementations as it does not utilize the Hive runtime for read/write operations. That is, when Trino executes a query through the Hive connector, it does not invoke any functions of the Hive engine. Instead, Trino interacts directly with the Hive Metastore and employs an HDFS client to perform operations on the raw files stored in a Hive warehouse location in HDFS. This brings forth a requirement that the HDFS Client component must be installed on each host alongside with Trino Workers.

In the built-in Hive catalog, all the configuration properties are auto-generated to ensure access to Hive Metastore and Hive warehouse directories of your ADH cluster. The built-in Hive catalog name is `hive-adh`. ADH forms this catalog automatically and generates the *hive-adh.properties* file at */etc/trino/conf/catalog/* if:

- Your ADH cluster has the Hive service installed. If Hive was installed after Trino, the Trino service must be restarted.
- The Hive catalog configuration is active.

Enable Hive catalog

By default, the built-in Hive catalog of the Trino service is disabled and can be activated using ADCM. For this, toggle the ***Hive configuration*** option of the Trino service (***Clusters → <clusterName> → Services → Trino → Configuration***) and then restart the Trino service.

Once enabled, check the catalog accessibility. For this, connect to Trino Coordinator (for example, via Trino CLI or an SQL client like DBeaver) and run the command:

```
SHOW CATALOGS;
```

The built-in Hive catalog appears in the list:

```
Catalog |
--------+
hive-adh|
system  |
```

> **TIP**
>
> The up-to-date JDBC connection string for connecting to the Trino service is available on the service configuration page in ADCM (***Clusters → <clusterName> → Services → Trino → Info***).

Supported Hive formats

The Trino's Hive catalog can work with Hive data files in the following formats:

- ORC
- Parquet
- Avro

As for serializable formats, the following SerDes are allowed:

- TextFile
- CSV using `org.apache.hadoop.hive.serde2.OpenCSVSerde`
- JSON using `org.apache.hive.hcatalog.data.JsonSerDe`
- RCFile using `ColumnarSerDe`
- RCFile using `LazyBinaryColumnarSerDe`
- SequenceFile

Hive catalog configuration properties

The Trino connector for Hive supports numerous configuration parameters for fine-tuning the interaction with Hive data files. The up-to-date properties list is available in Trino documentation. You can specify new or override the default property values using ADCM. For this, use the ***Custom Hive configuration*** settings section (***Clusters → <clusterName> → Services → Trino → Configuration***).

SQL support

> **IMPORTANT**
>
> To allow Trino to write to a Hive database, the user name specified while connecting to Trino (for example, via Trino CLI tool or DBeaver) must match the Unix user on the ADH host with sufficient permissions to write to Hive warehouse directories in HDFS. Otherwise, you may observe a similar error:
>
> ```
> Failed to create external path hdfs://adh/apps/hive/warehouse/trino_demo_schema1.db for database
> trino_demo_schema1. This may result in access not being allowed if the StorageBasedAuthorizationProvider is
> enabled: null
> ```

Schema management

A schema in Trino is used to group several tables/views and is equivalent to a database in Hive parlance. Below are examples of base commands for managing Trino schemas:

```
CREATE SCHEMA "hive-adh"."trino_demo_schema_hive"; ①
USE "hive-adh"."trino_demo_schema_hive"; ②
SHOW SCHEMAS;
DROP "hive-adh"."trino_demo_schema_hive"; ③
```

① Creates a new schema (Hive database). `hive-adh` is the name of the built-in Hive catalog. Notice the double quotes used for identifiers.

② Specifies a session schema. All the subsequent Trino SQL commands will run against the selected Hive database.

③ Deleting a non-empty schema is prohibited. To delete a schema, you have to remove all nested tables first.

To run Trino queries against arbitrary tables that belong to different schemas, you can specify a catalog, schema, and table name within your SQL command, for example:

```
SELECT * FROM "hive-adh"."trino_demo_schema_hive"."transactions" txns_hive
JOIN "iceberg-adh"."trino_demo_schema_iceberg"."transactions" txns_icb
ON txns_hive.txn_id = txns_icb.txn_id;
```

> **TIP**
>
> Using Trino UI, you can view and analyze SQL queries submitted to Trino.

DDL examples

Trino allows running most of the Hive DDL operations to manage Hive entities like tables, databases, materialized views, etc. Trino also supports a set of Hive table properties that specify metadata for Hive tables. These properties can be applied to a Hive table using the `WITH` clause as shown in the examples below. The following snippet provides basic DDL operations on Hive tables using the Trino catalog for Hive.

```
CREATE TABLE "hive-adh"."trino_demo_schema_hive"."transactions" (id BIGINT, value VARCHAR); ①

CREATE TABLE "hive-adh"."trino_demo_schema_hive"."transactions_with" (id BIGINT, value VARCHAR) ②
WITH (
    format = 'ORC',
    transactional=true
);

CREATE TABLE "hive-adh"."trino_demo_schema_hive"."transactions_pttnd" ( ③
    txn_id BIGINT,
    txn_amount DOUBLE,
    acc_id BIGINT
 ) WITH (
    partitioned_by=ARRAY['acc_id']
);

CREATE TABLE "hive-adh"."trino_demo_schema_hive"."transactions_bckttd" ( ④
    txn_id BIGINT,
    txn_amount DOUBLE,
    acc_id BIGINT
 ) WITH (
    bucketed_by=ARRAY['acc_id'],
    bucket_count=20
);

CREATE TABLE "hive-adh"."trino_demo_schema_hive"."transactions_ext" (
        id BIGINT,
        payload VARCHAR,
        request_time TIMESTAMP
) WITH (
  format = 'TEXTFILE',
  external_location = 'hdfs://adh/apps/hive/ext/test_ext_table' ⑤
);

SET SESSION "hive-adh".non_transactional_optimize_enabled=true; ⑥
ALTER TABLE "hive-adh"."trino_demo_schema_hive"."transactions_pttnd"
EXECUTE optimize (file_size_threshold => '128MB'); ⑦

ANALYZE "hive-adh"."trino_demo_schema_hive"."transactions_pttnd"; ⑧

CALL system.create_empty_partition( ⑨
    schema_name => "hive-adh",
    table_name => "trino_demo_schema_hive",
    partition_columns => ARRAY['acc_id'],
    partition_values => ARRAY['1005']
);
```

① Creates a basic Hive table. Notice that Trino accepts queries in ANSI SQL with corresponding data type, whereas Hive uses HiveQL, which differs from the traditional ANSI SQL. For example, when running this query, the `VARCHAR` data type will be cast to Hive's `STRING`.

② Creates a transactional, ORC-formatted table. Creating transactional Hive tables using Trino requires the ORC format to be used. If a table format is not specified explicitly, Trino creates a non-transactional table using the format specified by the `hive.storage-format` catalog configuration property.

③ Creates a partitioned Hive table. Notice the `ARRAY['col1', 'col2']` syntax for defining partitions.

④ Creates a table with a specific number of buckets.

⑤ Creates an external table that points to a custom HDFS location.

⑥ Sets a session property. Particularly, `non_transactional_optimize_enabled=true` is required to enable the `OPTIMIZE` command.

⑦ Runs the `OPTIMIZE` command that rewrites the content of the table so that it is merged into fewer but larger files.

⑧ Computes statistics for a table. Trino does not support gathering table statistics for transactional Hive tables.

⑨ Calls a system procedure to create an empty partition. This operation creates an empty partition directory in the Hive warehouse, e.g. */apps/hive/warehouse/trino_demo_schema.db/transactions_pttnd/acc=1005*. For more information about procedures supported by the Hive catalog, see Trino documentation.

DML examples

Trino supports most of the Hive DML operations, like `INSERT`, `UPDATE`, `DELETE`, `MERGE`, etc. The actual behavior of these commands depends on the table format and the Metastore being used. Below are some known limitations and peculiarities of running Hive CRUD operations via Trino:

- The `UPDATE` command is only supported for transactional tables. The target table must be transactional, ACID, and ORC-formatted.
- The `DELETE` command applied to a non-transactional table is only supported if the table is partitioned and the `WHERE` clause fully matches a partition. Transactional Hive tables in the ORC format support row-by-row deletion, in which `WHERE` may match arbitrary sets of rows.
- `MERGE` is only supported for ACID tables.
- ACID tables created with Hive Streaming Ingest are not supported.
- Gathering table statistics using `ANALYZE` is not supported for Hive transactional tables.
- `TRUNCATE` is not supported by the Hive catalog.

For more information about best practices and limitations on using SQL with the Trino catalog for Hive, see Trino documentation. The following snippet provides examples of DML operations with comments.

```
INSERT INTO "hive-adh"."trino_demo_schema_hive"."transactions_pttnd"
    VALUES (1, 15, 1001),
    (2, 100, 1002),
    (3, 75, 1001);

SELECT * FROM "hive-adh"."trino_demo_schema_hive"."transactions_pttnd"; ①

SELECT * FROM "hive-adh"."trino_demo_schema_hive"."transactions_pttnd$properties"; ②

SELECT * FROM "hive-adh"."trino_demo_schema_hive"."transactions_pttnd$partitions"; ③

SELECT *, "$path", "$file_size", "$partition" FROM "hive-adh"."trino_demo_schema_hive"."transactions_pttnd"; ④

SELECT *, "$path", "$file_size" FROM "hive-adh"."trino_demo_schema_hive"."transactions_pttnd"
WHERE "$partition" = 'acc_id=1001'; ⑤

SELECT * FROM "hive-adh"."trino_demo_schema_hive"."transactions" txns_hv ⑥
JOIN "iceberg-adh"."trino_demo_schema_iceberg"."transactions" txns_icb
ON txns_hv.txn_id = txns_icb.txn_id;

DELETE FROM "hive-adh"."trino_demo_schema_hive"."transactions_pttnd" ⑦
WHERE txn_id = 2;
```

① An ANSI SQL `SELECT` against the selected Hive database.

② Returns table metadata from the hidden `$properties` table.

③ Returns a list of all partitions for a partitioned table.

④ Returns table metadata for metadata columns.

⑤ Returns the file path and file size of all the row records that belong to a specific partition (`acc_id=1001`).

⑥ Selects data from two independent Trino catalogs — the Hive catalog `hive-adh` and the Iceberg catalog `iceberg-adh`. After Trino gets result sets from both catalogs, it applies a `JOIN` operation and returns the final result set to the client.

⑦ Deletes a row. The target table must be transactional. Otherwise, Trino throws an error.

For detailed information on Hive DML operations supported by Trino, see Trino documentation.

Performance optimization

Trino provides several optimizations to speed up the interaction with Hive tables. These are the following:

- Table statistics. Trino supports gathering table statistics (for transactional tables only) to improve query processing performance.
- Dynamic filtering. Provides a significant performance boost for queries with selective `JOIN` operations by avoiding reading the "unnecessary" data that would be filtered by the `JOIN` condition.
- File system caching. Caches the chunks of data, which are frequently transmitted from HDFS and accessed by workers while executing similar queries.
- Table redirection. A technique that allows redirecting CRUD operations to a different catalog.

Trino ADB connector

Konstantin Alpashkin

To Table of Contents

Arenadata DB (ADB) is a distributed analytical data storage built on the open-source massively parallel Greenplum DBMS.

The ADH Trino service provides an SQL query engine used for processing data in parallel, distributed over multiple storages, such as object storages, databases, and file systems. The Trino service comes with a connector that allows you to execute queries in an ADB cluster.

This section shows you how to use the Trino connector for ADB and includes the following articles:

- Overview
- Prerequisites
- Usage example
- Configuration

Trino ADB connector overview

Konstantin Alpashkin

Contents

To Table of Contents

Connector operating modes
  Enable parallel retrieve cursor mode

Connecting to an ADB cluster through the Trino connector begins with creating a new Trino catalog. The catalog configuration must use the `adb` connector type and must contain the following set of properties (at minimum):

```
connector.name=adb ①
connection-url=jdbc:postgresql://<adb_master>:5432/database ②
connection-user=<adb_username> ③
connection-password=<password>
```

① Sets the connector type to work with ADB.

② Sets the JDBC connection string to connect to the ADB server. The connector uses the PostgreSQL JDBC driver and supports all its configuration properties as well as inherits its limitations. For more information, see Trino PostgreSQL connector.

③ Specifies the ADB user to connect to ADB. The user must be pre-created on the ADB side and must have sufficient permissions to perform operations in the ADB cluster.

For more details about the ADB catalog configuration, see ADB catalog parameters and the usage example.

Connector operating modes

The Trino connector can interact with an ADB cluster in two modes:

- **Both read/write operations are performed via gpfdist**.

  `gpfdist` is a file distribution utility that allows loading and unloading data in parallel from external ADB tables. Under the hood, the Trino connector spins up `gpfdist` servers on ADH hosts, and ADB segments use `gpfdist` clients to exchange data with the Trino connector in parallel.

- **Write data to ADB via gpfdist, read data using gp_parallel_retrieve_cursor**.

  The `gp_parallel_retrieve_cursor` module is an enhanced cursor implementation that allows fetching query results in parallel directly from ADB segments. This cursor type provides faster parallel reads, adds extra flexibility by fetching data in chunks, and generally is more efficient for retrieving large result sets.

Enable parallel retrieve cursor mode

To enable fetching data from ADB with `gp_parallel_retrieve_cursor`, do the following:

1. In the ADB cluster, create the `gp_parallel_retrieve_cursor` extension:

   ```
   CREATE EXTENSION gp_parallel_retrieve_cursor;
   ```

2. When creating a new Trino catalog for ADB, set the `adb.connector.gp-parallel-cursor.enabled=true` property.

Trino ADB connector prerequisites

Konstantin Alpashkin

Contents

To Table of Contents

ADB requirements

ADH requirements

General requirements

This article describes the requirements for ADB and ADH clusters that have to be fulfilled to allow the Trino connector to interact with an ADB cluster. These requirements are mandatory for running the connector usage example.

ADB requirements

- The ADB cluster is installed according to the Online installation guide.
- The ADB cluster uses the default port (`5432`) for incoming JDBC connections. SSL is disabled.
- The ADB user, which is used by the Trino connector to access an ADB cluster, has sufficient read/write permissions. More details on granting permissions to the ADB user are described in the example.

ADH requirements

- The ADH cluster is installed according to the Online installation guide. The minimum ADH version is 4.0.0.
- The ADH cluster has the Trino service installed.

General requirements

- All ADB and ADH hosts have network access to each other. TCP ports used by ADB/ADH components are not in use.
- A JDBC client (for example, DBeaver) to connect to the Trino service.

Trino ADB connector usage example

Konstantin Alpashkin

Contents

To Table of Contents

Prepare ADB cluster

Run the example

This article describes a step-by-step scenario that highlights major operations with an ADB cluster using the Trino connector.

Prepare ADB cluster

To run the example scenario, ensure that your ADB cluster is configured according to the following requirements:

- The `adb_trino_demo_user` user with the `CREATEEXTTABLE` privileges and `123` password is created in the ADB cluster.

  To add the test user, run `psql` on the ADB master host:

  ```
  $ sudo su - gpadmin
  $ psql adb
  ```

  In the `psql` terminal, run the following query:

  ```
  CREATE ROLE adb_trino_demo_user
  WITH  CREATEEXTTABLE(protocol='gpfdist',type='readable')
        CREATEEXTTABLE(protocol='gpfdist',type='writable')
        LOGIN
        PASSWORD '123';
  ```

- The `adb_trino_demo_db` database exists in the ADB cluster. To create the database, use the query:

  ```
  CREATE DATABASE adb_trino_demo_db;
  ```

- The `adb_trino_demo_user` user is allowed to connect to the `adb_trino_demo_db` database. For this purpose, you can add the following record to the *pg_hba.conf* file:

  ```
  host    all  adb_trino_demo_user       0.0.0.0/0     md5
  ```

  Note that it is not a production-ready configuration. Use it only for test needs. Instead of `0.0.0.0/0`, you can specify the IP addresses (with a subnet number) of the ADH hosts with Trino service components.

> TIP
>
> You can modify the *pg_hba.conf* file via the ADCM web interface. To do this, fill in the ***Custom pg_hba section*** parameter on the ***Configuration*** tab of the ADB service in the ADB cluster. To apply the changes, click ***Save*** and run the ***Reconfigure & Restart*** service action.

- The `adb_trino_demo_db` database contains the `author` test table (in the `public` schema). To create and populate the test table, run the following queries in `psql`.

  First, connect to the test database:

  ```
  \c adb_trino_demo_db
  ```

  Then, run the queries:

  ```
  CREATE TABLE public.author (id INT NOT NULL, name TEXT NOT NULL)
  WITH (appendoptimized=true)
  DISTRIBUTED BY(id);
  ```

  ```
  INSERT INTO public.author(id, name) VALUES
  (1,'Virginia Woolf'),
  (2,'J.R.R. Tolkien'),
  (3,'Harper Lee'),
  (4,'J.D. Salinger'),
  (5,'George Orwell'),
  (6,'John Steinbeck'),
  (7,'Margaret Mitchell'),
  (8,'Alan Moore'),
  (9,'Jack Kerouac'),
  (10,'Ernest Hemingway');
  ```

- The `adb_trino_demo_user` ADB user has sufficient permissions to work with the `author` table. To grant the read/write permissions, use the query:

  ```
  GRANT SELECT, INSERT ON public.author TO adb_trino_demo_user;
  ```

Run the example

1. Create a new Trino catalog to work with ADB. For this, open the Trino service configuration in ADCM, enable ***Custom catalogs***, and specify the following catalog properties:

catalog_name	key	value
adbcatalog	connector.name	adb
	connection-url	*jdbc:postgresql://10.92.40.79:5432/adb_trino_demo_db* Where: ■ `10.92.40.79` is the sample IP address of the ADB master host. Replace it with your own value. ■ `adb_trino_demo_db` is the test ADB database to work with. The DB has to be created in advance, the creation steps are described in prerequisites.
	connection-user	adb_trino_demo_user
	connection-password	123

2. Save the configuration and restart the Trino service.

   > TIP
   >
   > With dynamic catalog management, you can create/delete catalogs using `CREATE` / `DELETE` commands without having to restart the Trino service.

3. Connect to Trino using a JDBC client like DBeaver (or Trino CLI client). When connecting, provide Trino JDBC URL, for example:

   ```
   jdbc:trino://ka-adh-2.ru-central1.internal:18188
   ```

   The up-to-date JDBC connection string is available in ADCM (***Clusters → <ADH_cluster> → Services → Trino → Info***).

4. Once connected to Trino, list Trino catalogs using the command:

   ```
   SHOW CATALOGS;
   ```

   Verify that the test ADB catalog is available:

   ```
   Catalog   |
   ----------+
   adbcatalog|
   system    |
   ```

   > TIP
   >
   > To analyze queries submitted by the Trino connector to ADB, use Trino UI.

5. Select the test catalog and the schema for further queries:

   ```
   USE adbcatalog.public;
   ```

6. Read data from the test ADB table. The steps to create the test table are described in the prerequisites.

   ```
   SELECT * FROM author;
   ```

   The output:

   ```
   id|name             |
   --+-----------------+
    2|J.R.R. Tolkien   |
    1|Virginia Woolf   |
    6|John Steinbeck   |
    9|Jack Kerouac     |
    5|George Orwell    |
    3|Harper Lee       |
    7|Margaret Mitchell|
    8|Alan Moore       |
    4|J.D. Salinger    |
   10|Ernest Hemingway |
   ```

7. Write data to the test table:

   ```
   INSERT INTO author (id, name)
   VALUES (11, 'L.N. Tolstoy');
   ```

8. Check the write operation results on the ADB side.

   For this, run `psql` on the ADB master host:

   ```
   $ sudo su - gpadmin
   $ psql adb_trino_demo_db
   ```

   In the `psql` terminal, run the query:

   ```
   SELECT * FROM public.author
   WHERE name LIKE '%Tolstoy%';
   ```

   The output contains the record submitted to ADB by the Trino connector:

   ```
    id |     name
   ----+--------------
    11 | L.N. Tolstoy
   (1 row)
   ```

9. Create a new ADB table using the Trino connector. For this, run the following query in DBeaver:

   ```
   CREATE TABLE public.transactions (
     txn_id integer,
     acc_id varchar,
     txn_value double,
     txn_date date
   )
   WITH (
     distributed_by = ARRAY['acc_id', 'txn_date'], ①
     appendoptimized = true ②
   );
   ```

   ① Specifies the hash distribution by several columns.

   ② Sets the append-optimized table type.

   For more information on setting table properties via the connector, see ADB table properties.

   ```
   INSERT INTO public.transactions
   (txn_id, acc_id, txn_value, txn_date)
   VALUES
       (1, '1001', cast(20.00 as double), cast('2025-01-02' AS date)),
       (2, '1002', cast(110.50 as double), cast('2025-01-01' AS date)),
       (3, '1003', cast(23 as DOUBLE), cast('2025-01-01' AS date));
   ```

10. Check the newly created table using `psql` on the ADB master host.

    ```
    \dt;
    ```

    The new table is available:

    ```
                             List of relations
     Schema |     Name     | Type  |        Owner        |   Storage
    --------+--------------+-------+---------------------+-------------
     public | author       | table | gpadmin             | append only
     public | transactions | table | adb_trino_demo_user | append only
    (2 rows)
    ```

11. To delete the test ADB table, use the command:

    ```
    DROP TABLE IF EXISTS public.transactions;
    ```

    > NOTE
    >
    > The Trino connector can drop tables only if the ADB table owner is specified in the ADB catalog settings.

Trino ADB connector configuration

Konstantin Alpashkin

Contents

To Table of Contents

- ADB catalog parameters
- ADB table properties
- Data types mapping
  - Type mapping configuration
  - DECIMAL type processing
  - ARRAY type processing

This article describes configuration parameters of the Trino connector for ADB.

ADB catalog parameters

The following configuration parameters can be set when you create a Trino catalog for working with an ADB cluster. These properties can be specified using ADCM or within the `WITH` clause, for example:

```
CREATE CATALOG test_adb_catalog USING adb
WITH (
  "connection-url" = 'jdbc:postgresql://<adb_master>:5432/trino',
  ...
  "configuration_param1" = "value"
  "configuration_param2" = "value"
);
```

Configuration parameter	Description	Default value
adb.fetch-size	The maximum number of rows to fetch. If not set, the value is calculated using the formula: $\max\left(\frac{100000}{\text{column\_count}}, 1000\right)$	—
adb.array-mapping	The type of array mapping for the array data type. Possible values: `DISABLED`, `AS_ARRAY`	DISABLED
adb.enable-string-pushdown-with-collate	Enables string pushdown with collate	false
adb.include-system-tables	Indicates whether to include system tables	false
adb.max-scan-parallelism	The maximum level of parallelism for scanning tables	1
adb.connector.write-buffer-size	The maximum amount of memory allocated per sink for write queries	64MB
adb.connector.read-buffer-size	The maximum amount of memory allocated per record cursor for read queries	64MB
adb.connector.gpfdist.retry-timeout	The value of the ADB `gpfdist_retry_timeout` property. If not set, the ADB default value is used	—
adb.connector.gp-parallel-cursor.enabled	Enables gp-parallel-cursor for unloading data	false

ADB table properties

The following ADB table properties can be set using the `WITH` clause of the `CREATE TABLE` / `CREATE TABLE AS` commands. The properties described below are `NULL` by default. This means that if a property is not set when creating a table via the connector, the ADB cluster default values will be used.

Table property	Description
distributed	Specifies a way to distribute table records, which does not depend on specific columns. Valid values are `randomly` and `replicated`. For the hash distribution type, use the `distributed_by` property
distributed_by	Specifies the hash distribution of table entries. The value must be an array of column names, for example `ARRAY['col1', 'col2']`
appendoptimized	If set to `true`, the connector creates an append-optimized table. Setting `false` creates a heap table
blocksize	Sets the block size for append-optimized tables in bytes
orientation	Sets the orientation for append-optimized tables. Valid values are `column` and `row`
checksum	Setting `true` enables CRC checksum for append-optimized tables
compresstype	Specifies the data compression algorithm. Valid values are `zlib`, `zstd`, `rle_type`, and `none`. The actual support for specific algorithms depends on the ADB version
compresslevel	Sets the compression level. Valid values vary for each compression algorithm: ▪ zlib: `1` — `9`; ▪ zstd: `1` — `19`; ▪ rle_type: `1` — `4`.
fillfactor	Sets a fill factor for heap tables. Valid values: from `10` to `100`

An example of setting ADB table properties is below.

```
CREATE TABLE nations (
  n_nationkey integer,
  n_name char(25),
  n_regionkey integer,
  n_comment varchar(152)
)
WITH (
  distributed_by = ARRAY['nationkey', 'regionkey'], ①
  appendoptimized = true, ②
  orientation = 'column', ③
  compresstype = 'zstd' ④
)
```

① Specifies the hash distribution by several columns.

② Sets the append-optimized table type.

③ Sets the orientation to be columnar.

④ Specifies the compression using the ZSTD algorithm.

Data types mapping

When exchanging data between ADB and the Trino connector, the data types are mapped according to the following rules.

ADB to Trino types mapping

ADB type	Trino type
BIT	BOOLEAN
BOOLEAN	BOOLEAN
SMALLINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
REAL	REAL
DOUBLE	DOUBLE
NUMERIC(p, s)	DECIMAL(p, s) DECIMAL(p, s) is an alias of NUMERIC(p, s)
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
ENUM	VARCHAR
BYTEA	VARBINARY
DATE	DATE
TIME(n)	TIME(n)
TIMESTAMP(n)	TIMESTAMP(n)
TIMESTAMPTZ(n)	TIMESTAMP(n) WITH TIME ZONE
MONEY	VARCHAR
UUID	UUID
JSON	JSON
JSONB	JSON
HSTORE	MAP(VARCHAR, VARCHAR)
ARRAY	DISABLED, ARRAY For more information, see ARRAY type processing

Trino to ADB types mapping

ADB type	Trino type
BOOLEAN	BOOLEAN
SMALLINT	SMALLINT
TINYINT	SMALLINT
INTEGER	INTEGER
BIGINT	BIGINT
REAL	REAL
DOUBLE	DOUBLE
DECIMAL(p, s)	NUMERIC(p, s) DECIMAL(p, s) is an alias of NUMERIC(p, s)
CHAR(n)	CHAR(n)
VARCHAR(n)	VARCHAR(n)
VARBINARY	BYTEA
DATE	DATE
TIME(n)	TIME(n)
TIMESTAMP(n)	TIMESTAMP(n)
TIMESTAMP(n) WITH TIME ZONE	TIMESTAMPTZ(n)
UUID	UUID
JSON	JSONB
ARRAY	ARRAY For more information, see ARRAY type processing

Type mapping configuration

The following configuration parameters can be used to change the type casting logic.

Configuration parameter	Description	Default value
unsupported-type-handling	Defines how to process columns of unsupported types. The possible values: ▪ `IGNORE` — the column is ignored. ▪ `CONVERT_TO_VARCHAR` — convert the column value to `VARCHAR` of unlimited length. The corresponding session parameter is `unsupported_type_handling`.	IGNORE
jdbc-types-mapped-to-varchar	Defines a list of source data types to be forcibly cast to `VARCHAR` of unlimited length (even if the specified type is supported by the connector). For example, `jdbc-types-mapped-to-varchar=VARBINARY,UUID,`	—

DECIMAL type processing

By default, the connector throws an error for columns of the `DECIMAL` type without an explicit precision or scale definition. If the `decimal-mapping` connector configuration parameter (or the `decimal_mapping` session parameter) is set to `allow_overflow`, then columns of the `DECIMAL` type without explicit precision/scale will be converted to the connector `DECIMAL(38,0)` data type. Using the `decimal-default-scale` connector configuration parameter (or the `decimal_default_scale` session parameter), you can specify a custom scale value. The precision value is always `38`.

If a `DECIMAL` value does not fit into the specified precision and scale without rounding, the connector throws an error. You can change this behavior by enabling automatic rounding using the `decimal-rounding-mode` connector parameter (or the `decimal_rounding_mode` session parameter). The supported rounding values are `UNNECESSARY` (default), `UP`, `DOWN`, `CEILING`, `FLOOR`, `HALF_UP`, `HALF_DOWN`, and `HALF_EVEN`. These values correspond to the rounding behavior rules as of Java 17.

ARRAY type processing

The ADB `ARRAY` data type does not support fixed-dimensional arrays, while the Trino connector for ADB only supports one-dimensional arrays. You can instruct the connector how to handle the `ARRAY` types using the `adb.array-mapping` connector configuration parameter (the `array_mapping` session parameter). The supported values are:

- `DISABLED` (default) — throws an error when processing columns of the `ARRAY` type.
- `AS_ARRAY` — the connector treats ADB arrays as the `ARRAY` data type. This value is suitable for ADB arrays, which are actually one-dimensional. For a multi-dimensional array, an exception will be thrown.

OpenAPI connector overview

Konstantin Alpashkin

Contents

To Table of Contents

OpenAPI schema parsing

OpenAPI extensions

Error messages

Features and limitations

- Limitations

The Trino OpenAPI connector allows querying OpenAPI-compatible REST APIs via HTTP. OpenAPI is a standard way to describe the functionality of a REST API — available endpoints, supported HTTP methods, response types, request parameters, etc. The connector accepts an OpenAPI schema (JSON/YAML) and maps detected endpoints to tables. Then, the connector translates SQL queries into HTTP requests, handles the response, and returns the result set to the Trino client.

> **NOTE**
>
> It is strongly recommended to use the connector only for read operations. Inserts and updates are implemented in experimental mode.

Trino OpenAPI connector is bundled with ADH and can be used out of the box. To get started with the connector, see OpenAPI connector usage example. For configuration details, see OpenAPI connector configuration.

OpenAPI schema parsing

The connector maps an OpenAPI schema to SQL entities (tables, columns, requests) according to the following rules:

- Endpoint paths are mapped to separate tables. For example, the *[BaseURL]/transactions* endpoint is available for querying as a table named `transactions`. The camel case (`fooBar`) is converted to snake case (`foo_bar`) and special characters like `/` are converted to `_`.
- SQL operations are mapped to HTTP methods as follows:
  - `SELECT` ⇒ `GET` or `POST` (if `GET` is unavailable);
  - `INSERT` ⇒ `POST` / `PUT`;
  - `UPDATE` ⇒ `PATCH` / `POST`;
  - `DELETE` ⇒ `DELETE`.
- All request parameters are mapped to columns, including path, query, and header parameters.
- Fields of the `HTTP OK (200)` response type are mapped to columns.
- All columns are disambiguated:
  - If a field with the same name but a different data type appears in the response, it is postfixed with `_2`, `_3`, etc.
  - If a field with the same name appears both in the request and response, the request body field is postfixed with `_req`.

OpenAPI extensions

OpenAPI supports custom extensions, which are defined in the OpenAPI schema as blocks prefixed with `x-`. Such extensions can be used for fine-tuning and adding extra capabilities to the connector (for example, x-unwrap, x-pagination).

> **TIP**
>
> If an OpenAPI schema returned by a REST service cannot be modified to include `x-` extensions, you can save the schema file locally and modify it as needed.

Error messages

When the connector makes an API call and gets an error, the connector returns an SQL error to the Trino client. Typically, such errors contain only the HTTP status code, and the response message is not displayed (it can be non-human-readable or might contain sensitive data).

To extract human-readable error messages from error responses, set the `errorPath` property in the `x-trino` block of the OpenAPI schema. For example:

```
paths:
  /records:
    get:
      responses:
        # ...
      x-trino:
       errorPath: "$response.body#/message"
```

Features and limitations

The connector implementation distributed with ADH includes several improvements and features, namely:

- Array unwrapping. Allows for automatic expansion of array fields in the HTTP response.
- Pagination for the `LIMIT` pushdown. Allows querying paginated responses from a REST API.

Limitations

The connector has the following limitations:

- Limited support for `INSERT` / `DELETE` / `UPDATE` operations. The connector is recommended only for read operations.
- Limited predicate pushdown support. Predicate pushdown works well only for APIs that support query parameters filtering, and which are fully described in the OpenAPI schema. The connector may cause performance issues on large datasets, since due to the lack of predicate pushdowns, the connector fetches all the dataset and only then applies in-memory filtering.

OpenAPI connector usage example

Konstantin Alpashkin

Contents

To Table of Contents

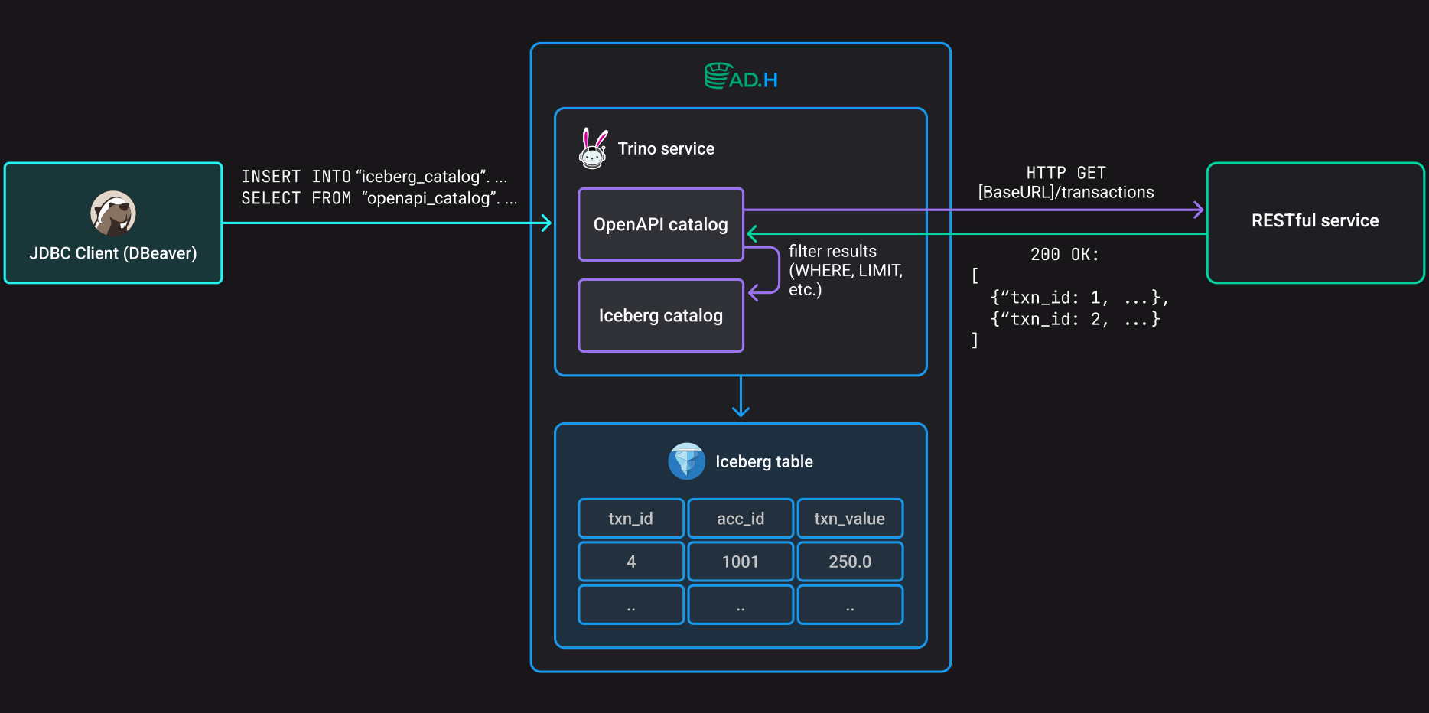
Prerequisites

Prepare a test OpenAPI server

- Run the test server

Run the example

This article describes a step-by-step scenario that shows how to use the Trino OpenAPI connector. The high-level scenario workflow is presented in the following diagram.



Trino OpenAPI connector use case

Prerequisites

To run the scenario, the following prerequisites must be met:

- An ADH cluster with the 4.1.0 version or higher is installed and running. The cluster includes the following services:
  - ADPG
  - Core configuration
  - HDFS
  - Hive
  - Trino
  - YARN
  - ZooKeeper
- A test REST API service with an OpenAPI schema. In this example, a simple Python application emulates a mock RESTful server with bank transactions. The sample code and the instructions on running the app are provided in the Prepare a test OpenAPI server section.
- Two Trino catalogs are created:
  - `openapi-catalog`. The catalog for fetching data from the mock server.
  - `iceberg-adh`. The catalog for writing filtered data to an Iceberg table.

  Catalog creation steps are described in the scenario.
- A JDBC client (for example, DBeaver) to connect to the Trino service.

Prepare a test OpenAPI server

In this example, a Python application built with the FastAPI framework emulates a REST server that returns bank transactions over HTTP. The mock server exposes the following REST endpoints:

GET [BaseURL]/transactions

Returns a list of all transactions, for example:

```
    [
        {
            "txn_id": 1,
            "acc_id": 1002,
            "txn_value": 184.0,
            "txn_type": "withdrawal"
        },
        {
            "txn_id": 2,
            "acc_id": 1000,
            "txn_value": 282.95,
            "txn_type": "spb"
        },
        {
            "txn_id": 3,
            "acc_id": 1001,
            "txn_value": 100.00,
            "txn_type": "deposit"
        }
    ]
```

GET [BaseURL]/transactions_batch

Returns a batch of transactions of a fixed size:

```
{
    "transactions_count": 2,
    "items":
    [
        {
            "txn_id": 1,
            "acc_id": 1002,
            "txn_value": 184.0,
            "txn_type": "withdrawal"
        },
        {
            "txn_id": 2,
            "acc_id": 1000,
            "txn_value": 282.95,
            "txn_type": "spb"
        }
    ]
}
```

GET [BaseURL]/openapi.yaml

Returns an OpenAPI schema of the mock server. FastAPI provides automatic schema generation and custom `x-…` fields are injected into the schema programmatically. For example:

```
openapi: 3.1.0
info:
  title: Transaction API
  description: Transaction API with x-unwrap demo
  version: 1.0.0
paths:
  /transactions:
    get:
      summary: Get Transactions
      operationId: get_transactions_transactions_get
      responses:
        '200':
          description: Successful Response
          content:
            application/json:
              schema:
                items:
                  $ref: '#/components/schemas/Transaction'
                type: array
                title: Response Get Transactions Transactions Get
  /transactions_batch:
    get:
      summary: Get Transactions Batch
      operationId: get_transactions_batch_transactions_batch_get
      responses:
        '200':
          description: Successful Response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TransactionBatch'
      x-unwrap:
        resultParam: $response.body#/items
        includeRoot: 'false'
# ...
```

Run the test server

To run the mock server, follow the steps:

> **TIP**
>
> Use the Python interpreter bundled with ADH (*/opt/python3.10/*).

1. Create and activate a new virtual environment.

   ```
   $ mkdir demo_mock_server
   $ cd demo_mock_server
   $ /opt/python3.10/bin/python3 -m venv venv
   $ source venv/bin/activate
   ```

2. Create the *server.py* file and paste the following code:

   server.py

   ```
   from fastapi import FastAPI
   from fastapi.openapi.utils import get_openapi
   from fastapi.responses import Response
   from pydantic import BaseModel
   from typing import List
   import random
   import uvicorn
   import yaml


   class Transaction(BaseModel):
       txn_id: int
       acc_id: int
       txn_value: float
       txn_type: str


   class TransactionBatch(BaseModel):
       total_count: int
       items: List[Transaction]


   app = FastAPI(title="Transaction API", version="1.0.0")


   def generate_transactions(n: int = 10) -> List[Transaction]:
       acc_ids = [1000, 1001, 1002]
       txn_types = ["deposit", "withdrawal", "spb"]
       transactions = []
       for txn_id in range(1, n + 1):
           transactions.append(
               Transaction(
                   txn_id=txn_id,
                   acc_id=random.choice(acc_ids),
                   txn_value=round(random.uniform(0, 500), 2),
                   txn_type=random.choice(txn_types),
               )
           )
       return transactions


   transactions = generate_transactions(10)


   @app.get("/transactions", response_model=List[Transaction])
   def get_transactions():
       return transactions


   # solely for x-unwrap demonstration
   @app.get("/transactions_batch", response_model=TransactionBatch)
   def get_transactions_batch():
       return {
           "total_count": len(transactions) / 2,
           "items": transactions[:5],
       }


   def custom_openapi():
       if app.openapi_schema:
           return app.openapi_schema

       openapi_schema = get_openapi(
           title=app.title,
           version=app.version,
           description="Transaction API with x-unwrap demo",
           routes=app.routes,
       )

       # Add x-unwrap for /transactions_batch
       openapi_schema["paths"]["/transactions_batch"]["get"]["x-unwrap"] = {
           "resultParam": "$response.body#/items",
           "includeRoot": "false"
       }

       app.openapi_schema = openapi_schema
       return app.openapi_schema


   app.openapi = custom_openapi


   @app.get("/openapi.yaml", include_in_schema=False)
   def get_openapi_yaml():
       openapi_schema = custom_openapi()
       yaml_schema = yaml.dump(openapi_schema, allow_unicode=True,
                               sort_keys=False)
       return Response(content=yaml_schema, media_type="application/yaml")


   if __name__ == "__main__":
       uvicorn.run(app, host="0.0.0.0", port=8003)
   ```

3. Install the dependencies using the following *requirements.txt* file:

   ```
   $ pip install -r requirements.txt
   ```

   requirements.txt

   ```
   annotated-types==0.7.0
   anyio==4.10.0
   click==8.2.1
   exceptiongroup==1.3.0
   fastapi==0.116.1
   h11==0.16.0
   idna==3.10
   pydantic==2.11.7
   pydantic_core==2.33.2
   PyYAML==6.0.2
   sniffio==1.3.1
   starlette==0.47.2
   typing-inspection==0.4.1
   typing_extensions==4.14.1
   uvicorn==0.35.0
   ```

4. Run the application:

   ```
   $ python server.py
   ```

   The output:

   ```
   INFO:     Started server process [26147]
   INFO:     Waiting for application startup.
   INFO:     Application startup complete.
   INFO:     Uvicorn running on http://0.0.0.0:8003 (Press CTRL+C to quit)
   ...
   ```

Run the example

1. Ensure the mock server is up and running.
2. In ADCM, enable the built-in Iceberg catalog (***Clusters → <clusterName> → Services → Trino → Iceberg configuration***).
3. (Optional) Enable dynamic catalog management (***Clusters → <clusterName> → Services → Trino → Components → Trino Coordinator → Trino catalog management → catalog.management***). This allows creating new catalogs using the `CREATE CATALOG` command. Restart the Trino service.
4. Connect to Trino Coordinator using DBeaver. The JDBC connection string is available in ADCM (***Clusters → <clusterName> → Services → Trino → Info***). In the connection string, specify the name of a user with HDFS write permissions. This is required to allow certain write operations by the Iceberg catalog in further steps (writing raw data to HDFS).
5. In DBeaver, create another Trino catalog with the following SQL:

   ```
   CREATE CATALOG "openapi-catalog" USING openapi
   WITH (
       "spec-location" = 'http://10.92.41.251:8003/openapi.yaml', ①
       "base-uri" = 'http://10.92.41.251:8003', ②
       "authentication.type" = 'none' ③
   );
   ```

   ① The endpoint that returns an OpenAPI schema of the mock server. In this scenario, the schema is generated by FastAPI.

   ② Base URL of the mock server's REST API.

   ③ No authentication is used in this example for simplicity. For more information on supported authentication types, see the OpenAPI connector configuration article.

6. Verify the new catalogs:

   ```
   SHOW CATALOGS;
   ```

   The output:

   ```
   Catalog             |
   --------------------+
   iceberg-adh         |
   system              |
   openapi-catalog     |
   ```

7. Create a test Iceberg table:

   ```
   USE "iceberg-adh".default;
   CREATE TABLE transactions_spb_only(txn_id int, acc_id int, txn_value double);
   ```

8. Fetch transactions from the mock server, filter by transaction type, and write filtered result set to the Iceberg table with one SQL command:

   ```
   INSERT INTO "iceberg-adh".default.transactions_spb_only (txn_id, acc_id, txn_value)
   SELECT txn_id, acc_id, txn_value
   FROM "openapi-catalog".default.transactions
   WHERE txn_type = 'spb';
   ```

9. Verify the data in the Iceberg table:

   ```
   SELECT * FROM "iceberg-adh".default.transactions_spb_only;
   ```

   Sample output:

   ```
   txn_id|acc_id|txn_value|
   ------+------+---------+
        1|  1001|   275.32|
        2|  1001|    12.24|
        6|  1000|    89.72|
        7|  1000|    40.71|
        8|  1002|   470.56|
   ```

   Verify data in HDFS:

   ```
   $ hdfs dfs -ls /apps/hive/warehouse/transactions_spb_only-7947dbdc9e8d4cf6b5db7558a64f16c5
   ```

   Sample output:

   ```
   drwxrwxr-x   - hdfs hadoop          0 2025-09-17 16:01 hdfs://adh/apps/hive/warehouse/transactions_spb_only-
   7947dbdc9e8d4cf6b5db7558a64f16c5/data
   drwxrwxr-x   - hdfs hadoop          0 2025-09-17 16:01 hdfs://adh/apps/hive/warehouse/transactions_spb_only-
   7947dbdc9e8d4cf6b5db7558a64f16c5/metadata
   ```

OpenAPI connector configuration

Konstantin Alpashkin

Contents

To Table of Contents

Array unwrapping

Pagination

The following configuration parameters can be used to tune the connector's behavior. You can specify these parameters when creating a new Trino catalog using ADCM or inside the `WITH` clause, for example:

```
CREATE CATALOG "openapi-catalog" USING openapi
WITH (
    "spec-location" = ...,
    "base-uri" = ...,
    ...
);
```

Configuration parameter	Description
spec-location	URL or file name containing the OpenAPI specification (JSON/YAML)
base-uri	Base URL of the REST API
authentication.type	Default authentication type to use if it is not specified in the OpenAPI specification. Possible values: `none`, `http`, `api_key`, `oauth`
authentication.scheme	Authentication schema for the `http` authentication type. Possible values: `basic`, `bearer`
authentication.client-id	OAuth Client ID
authentication.client-secret	OAuth Client secret
authentication.username	Username for the `http` and `oauth` authentication types
authentication.password	Password for the `http` and `oauth` authentication types
authentication.bearer-token	Bearer token for the `http` authentication
authentication.api-key-name	API key name
authentication.api-key-value	API key value
max-requests-per-second	Maximum number of HTTP requests per second that can be submitted from a single Trino node
max-splits-per-second	Maximum number of splits per second generated when executing a query

Under the hood the connector uses the Airlift HTTP client, which accepts its own set of properties, for example:

```
openApi.http-client.log.enabled=true
openApi.http-client.log.path=logs
```

For more information on tuning the client, see Airlift HTTP client properties.

Array unwrapping

When processing an HTTP response, the connector can unwrap fields with array payloads. For example, assume the connector receives the following HTTP-response:

```
{
    "transactions_count": 10,
    "items":
    [
        {
            "txn_id": 1,
            "acc_id": 1002,
            "txn_value": 184.0,
            "txn_type": "withdrawal"
        },
        {
            "txn_id": 2,
            "acc_id": 1000,
            "txn_value": 282.95,
            "txn_type": "spb"
        },
        {
            "txn_id": 3,
            "acc_id": 1001,
            "txn_value": 100.00,
            "txn_type": "deposit"
        }
    ]
}
```

By default, Trino would convert `items` to a single table row with the array elements:

```
total_count|                                          items|
-----------+-----------------------------------------------+
         10|[{"txn_id": 1, ...}, {"txn_id": 1, ...}, ...]|
```

To force Trino to "explode" the array and present its contents as a separate table, in your OpenAPI schema, specify the `x-unwrap` section. For the HTTP response above, the `x-unwrap` section looks as follows:

```
/transactions_batch:
   get:
     summary: Get Transactions Batch
     operationId: get_transactions_batch_transactions_batch_get
     responses:
       '200':
         description: Successful Response
         content:
           application/json:
             schema:
               $ref: '#/components/schemas/TransactionBatch'
     x-unwrap:
       resultParam: $response.body#/items ①
       includeRoot: 'false' ②
```

① The response field holding an array that has to be exploded and displayed as a separate table.

② Indicates whether to include duplicate fields (`transactions_count`).

With such `x-unwrap` configuration, the connector maps the content of the `items` field to a separate table:

```
txn_id|acc_id|txn_value|txn_type  |
------+------+---------+----------+
     1|  1002|184.00   |withdrawal|
     2|  1000|282.95   |spb       |
     3|  1001|100.00   |deposit   |
```

Pagination

The connector supports pagination for the `LIMIT` pushdown. To enable pagination, in your OpenAPI schema, use `x-pagination` for the required endpoint:

```
x-pagination:
  pageParam: "page"
  limitParam: "size"
  resultsPath: "$response.body#/elements"
  totalResultsPath: "$response.body#/totalCount"
```

Using pagination is not recommended as it may return inconsistent results, especially on large datasets.

LDAP authentication in Trino

Sergei Tikhomirov

Contents

To Table of Contents

Enable LDAP

Check

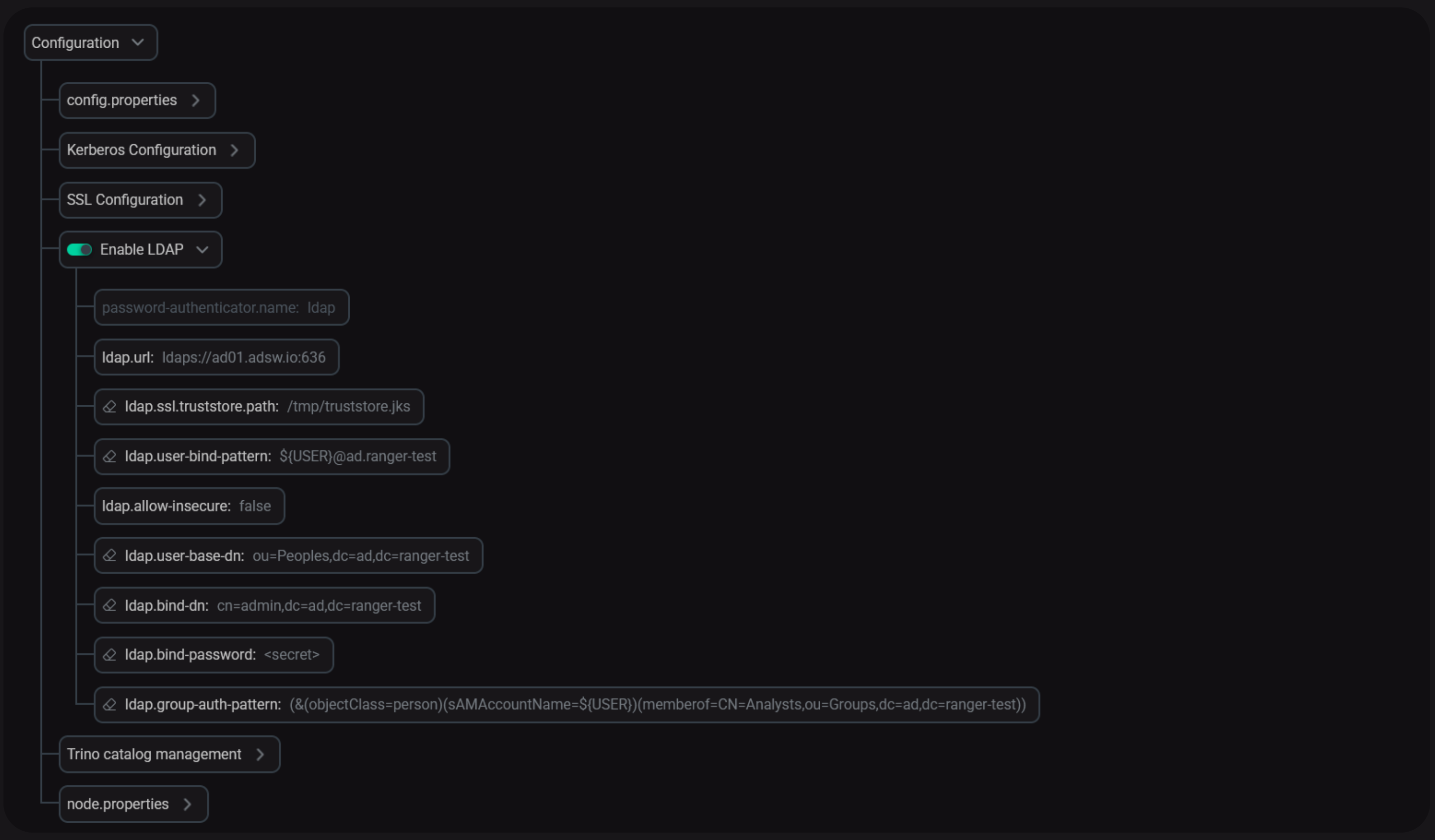
Enable LDAP

NOTE

The LDAP authentication only works in clusters with active SSL.

To enable LDAP authentication, follow the steps below:

1. On the ***Clusters*** page, select your ADH cluster and proceed to the ***Services*** tab in the cluster menu.
2. Select the ***Trino*** service and go to the ***Components*** tab.
3. Click the ***Trino Coordinator*** component and activate the ***Enable LDAP*** parameter group.
4. Fill in the LDAP parameters for Active Directory or FreeIPA and restart ***Trino Coordinator***.



```
password-authenticator.name: ldap
ldap.url: ldaps://ad01.adsw.io:636
ldap.ssl.truststore.path: /tmp/truststore.jks
ldap.user-bind-pattern: ${USER}@ad.ranger-test
ldap.allow-insecure: false
ldap.user-base-dn: ou=Peoples,dc=ad,dc=ranger-test
ldap.bind-dn: cn=admin,dc=ad,dc=ranger-test
ldap.bind-password: <secret>
ldap.group-auth-pattern: (&(objectClass=person)(sAMAccountName=${USER})(memberof=CN=Analysts,ou=Groups,dc=ad,dc=ranger-test))
```

LDAP AD configuration

Check

To check that the authentication works, try using the Trino CLI tool:

1. Connect to Trino Coordinator with specific parameters:

```
$ ./trino https://<coordinator_host>:18188 --keystore-path <keystore-path> --keystore-password <keystore_password> --truststore-path <truststore-path> --truststore-password <truststore_password> --user=<username> --password
```

2. Enter a password when prompted.
3. Once the Trino shell is active, run the `SHOW CATALOGS;` command. The output should be similar to the following:

```
Catalog
-------------
hive-adh
iceberg-adh
system
(3 rows)

Query 20250306_005901_00001_6smm5, FINISHED, 1 node
Splits: 7 total, 7 done (100.00%)
0.28 [0 rows, 0B] [0 rows/s, 0B/s]
```

Kerberos authentication in Trino

Sergei Tikhomirov

To Table of Contents

Kerberos authentication pipeline for Trino requires performing several additional actions which depend on whether the Trino Ranger plugin is enabled or not. This articles demonstrates the differences of the approaches.

To setup Kerberos authentication for Trino, follow the steps below:

1. Activate SSL for your ADH cluster.
2. Enable Kerberos for your ADH cluster.
3. Go to the Trino Coordinator component settings and edit the ***user-mapping.json*** file parameter so that its contents are as follows:

```
{
  "rules": [
    {
      "pattern": "(.+)@(.*)"
    },
    {
      "pattern": "^(.*?)/.*$"
    }
  ]
}
```

4. Configure impersonation settings.

With Ranger plugin

1. In Ranger, create a user for the Kerberos principal that will be executing Trino CLI commands.
2. Add that user to the `all - trinouser`, `all - catalog, schema, table, column`, and `all - queryid` policies in Trino.

Without Ranger plugin

On the Trino configuration page in ADCM, expand the ***access-control.properties*** parameter group and edit the ***rules_json_patter*** parameter so that it includes permissions for impersonation and catalogs, for example:

```
{
  "system": {
    "impersonation": [
      {
        "user": "<user with permissions, like `trino`>",
        "allow": {
          "user": "<user who will run trino-cli>"
        }
      }
    ]
  },
  "catalogs": [
    {
      "role": "admin",
      "catalog": "hive-adh",
      "allow": "all"
    },
    {
      "user": "trino",
      "catalog": "iceberg-adh",
      "allow": "all"
    },
    {
      "user": "<user who will run trino-cli>",
      "catalog": "iceberg-adh",
      "allow": "all"
    }
  ]
}
```

Check that everything works correctly by running a Trino CLI command:

```
$ trino --execute 'select * from "hive-adh"."default"."solar_objects";'
```

The output is expected to be the table's content:

```
"1","Sun","1989100000","1392000"
"2","Mercury","330","4879"
"3","Venus","4867","12104"
"4","Earth","5972","12742"
"5","Mars","642","6780"
"6","Jupiter","1898187","139822"
"7","Saturn","568317","116464"
"8","Uranus","86813","50724"
"9","Neptune","102413","49244"
```

Ranger authorization with Hive policies

Sergei Tikhomirov

Contents

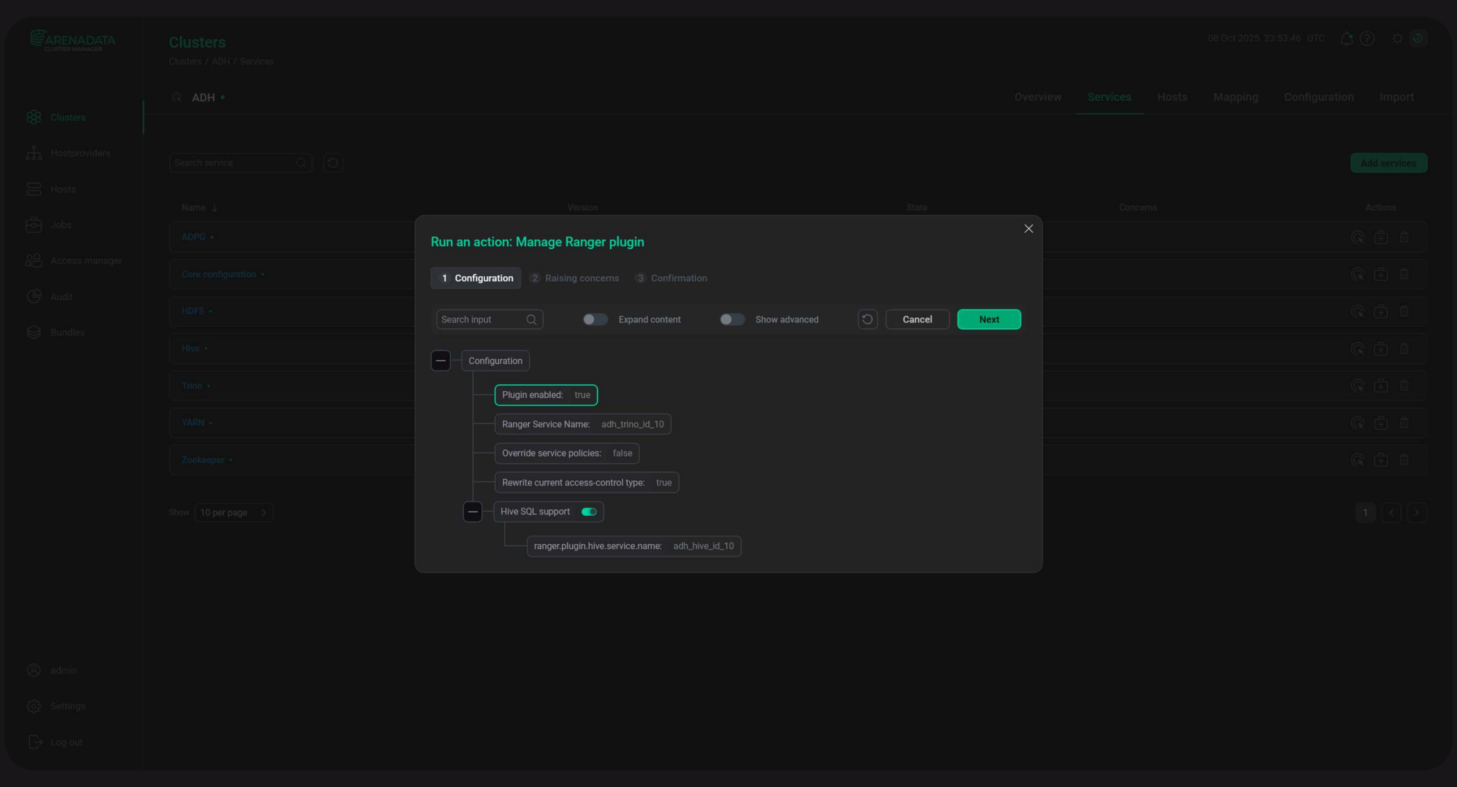
To Table of Contents

Setup
Check

The Trino Ranger plugin can be configured for the Hadoop SQL policies. This article describes how to configure and test this feature.

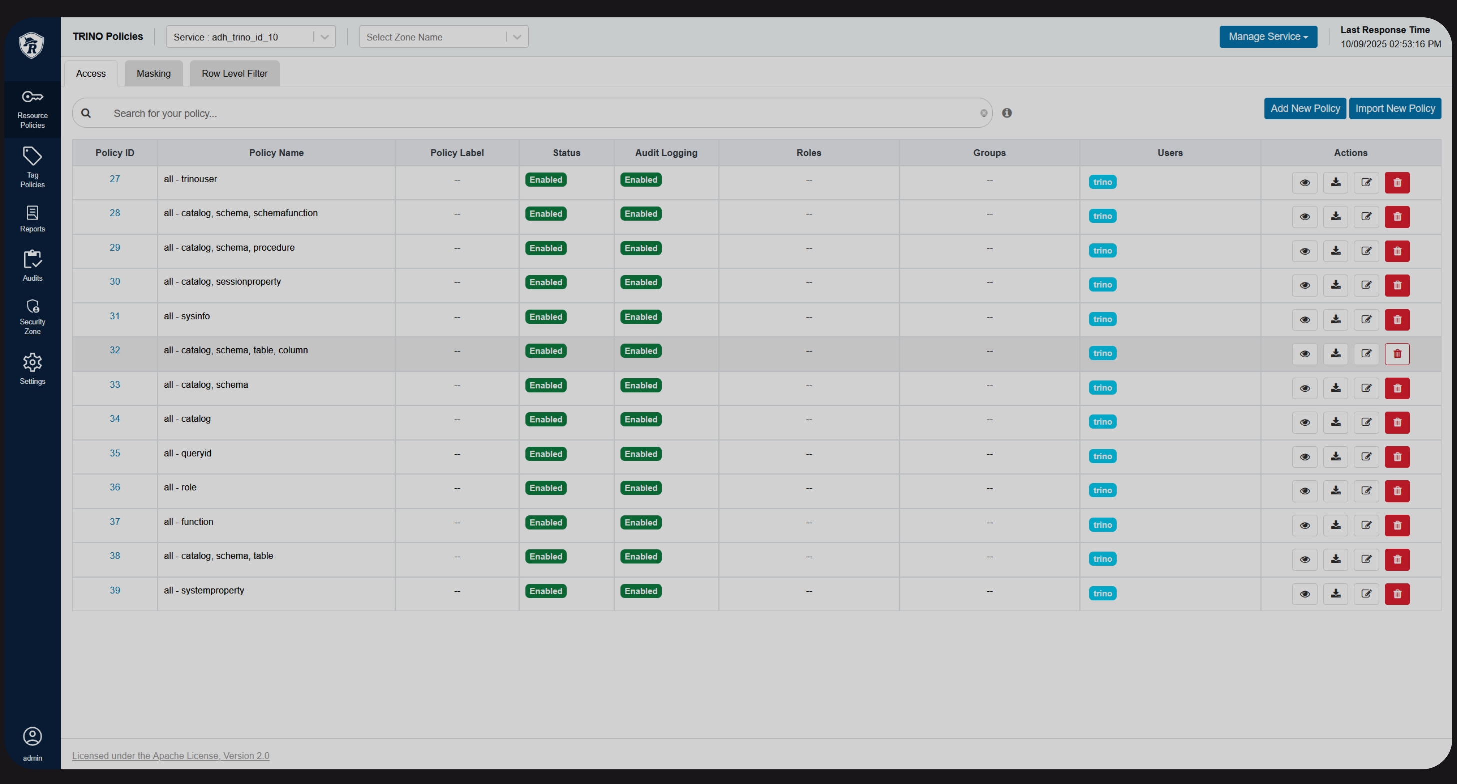
Setup

1. Enable the Hive Ranger plugin.
2. Enable the Trino Ranger plugin with the ***Hive SQL Support*** parameter activated. Make sure that the ***ranger.plugin.hive.service.name*** parameter matches the Hive service name in Ranger.



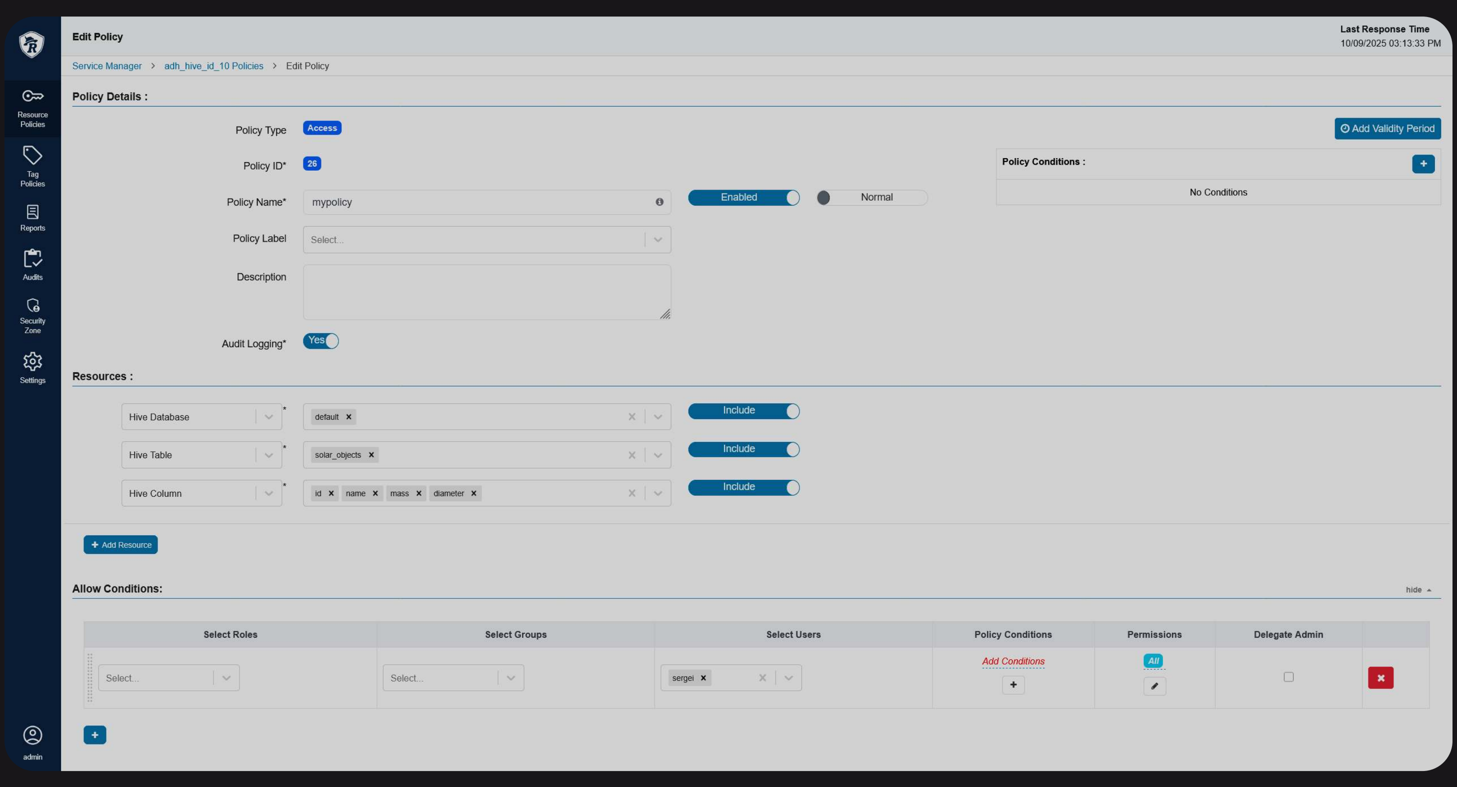
The Trino Ranger plugin parameters

3. Remove the `all - catalog, schema, table, column` policy for Trino in Ranger.



The Trino policy to remove

4. Add a user that should be able to authenticate into the `all - trinouser`, `all - catalog, schema`, and `all - queryid` policies in Trino.
5. In Hive, create a policy that would allow the user access to all columns in a specific database and table.



Create a Hive policy

Check

To check that the setup works correctly, follow the steps below:

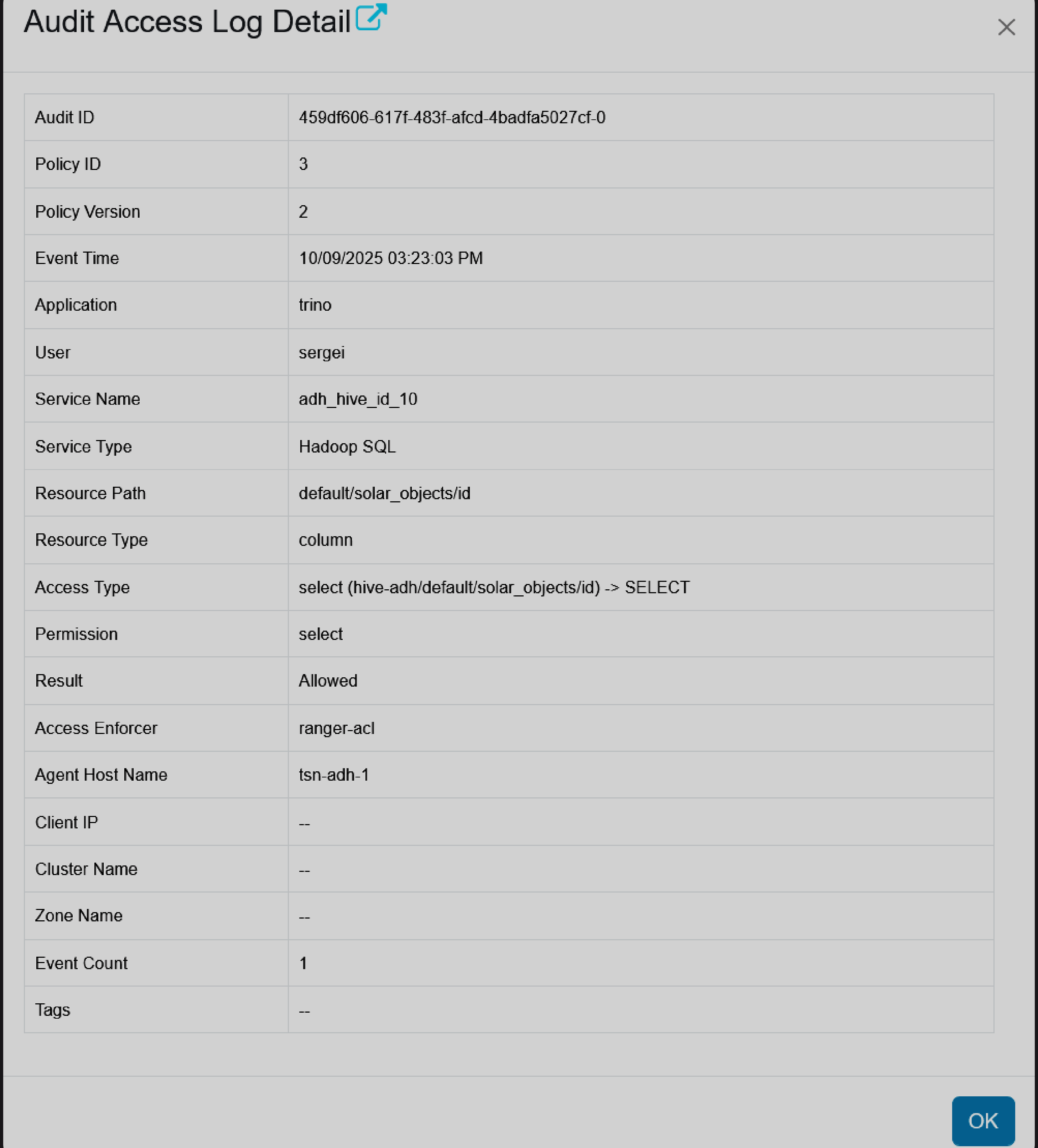
1. On an ADH cluster host, execute a query using the Trino shell:

```
$ trino --execute 'select * from "hive-adh"."default"."solar_objects";'
```

The output is expected to be the table's content:

```
"1","Sun","1989100000","1392000"
"2","Mercury","330","4879"
"3","Venus","4867","12104"
"4","Earth","5972","12742"
"5","Mars","642","6780"
"6","Jupiter","1898187","139822"
"7","Saturn","568317","116464"
"8","Uranus","86813","50724"
"9","Neptune","102413","49244"
```

2. In the Ranger Admin UI, check the ***Audits*** → ***Access*** page to see that the access was allowed and the impersonation works correctly.



Audit Access Log Detail

Audit ID	459df606-617f-483f-afcd-4badfa5027cf-0
Policy ID	3
Policy Version	2
Event Time	10/09/2025 03:23:03 PM
Application	trino
User	sergei
Service Name	adh_hive_id_10
Service Type	Hadoop SQL
Resource Path	default/solar_objects/id
Resource Type	column
Access Type	select (hive-adh/default/solar_objects/id) -> SELECT
Permission	select
Result	Allowed
Access Enforcer	ranger-acl
Agent Host Name	tsn-adh-1
Client IP	--
Cluster Name	--
Zone Name	--
Event Count	1
Tags	--

The Ranger Audits page

Trino service management via ADCM

Vladimir Adamenkov

Contents

To Table of Contents

Overview

Overview

The ADCM UI provides actions to manage the Trino service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Trino service are listed in the table below.

Trino service actions

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Trino components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM. The action provides an additional ***Init DB credentials*** option. If selected, the database entities (user, database, role) will be initialized in the ADPG service, used by DBCatalogManager
Manage Ranger plugin	Enables or disables Ranger plugin for Trino

The Trino service has the following components:

- Trino Coordinator
- Trino Worker
- Trino Client
- HAProxy Trino

The components have a similar set of available actions listed in the table below.

Trino components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

Trino Client actions

Action	Description
Apply client configs	Applies Trino Client configurations specified in ADCM on all hosts where the component is installed

YARN UI overview

Elena Kostyuchenko

Contents

To Table of Contents

MapReduce History Server

NodeManager

ResourceManager

Timeline server

YARN web interfaces (UI) display diagnostic information and enable access to the logs, configuration, and job status. The UIs are available by the *http://<server-hostname>:<port>/* URL, where `<server-hostname>` is the IP address or, if you have configured the redirect in the *hosts* file, FQDN of a host. And `<port>` is the port of the YARN component's UI.

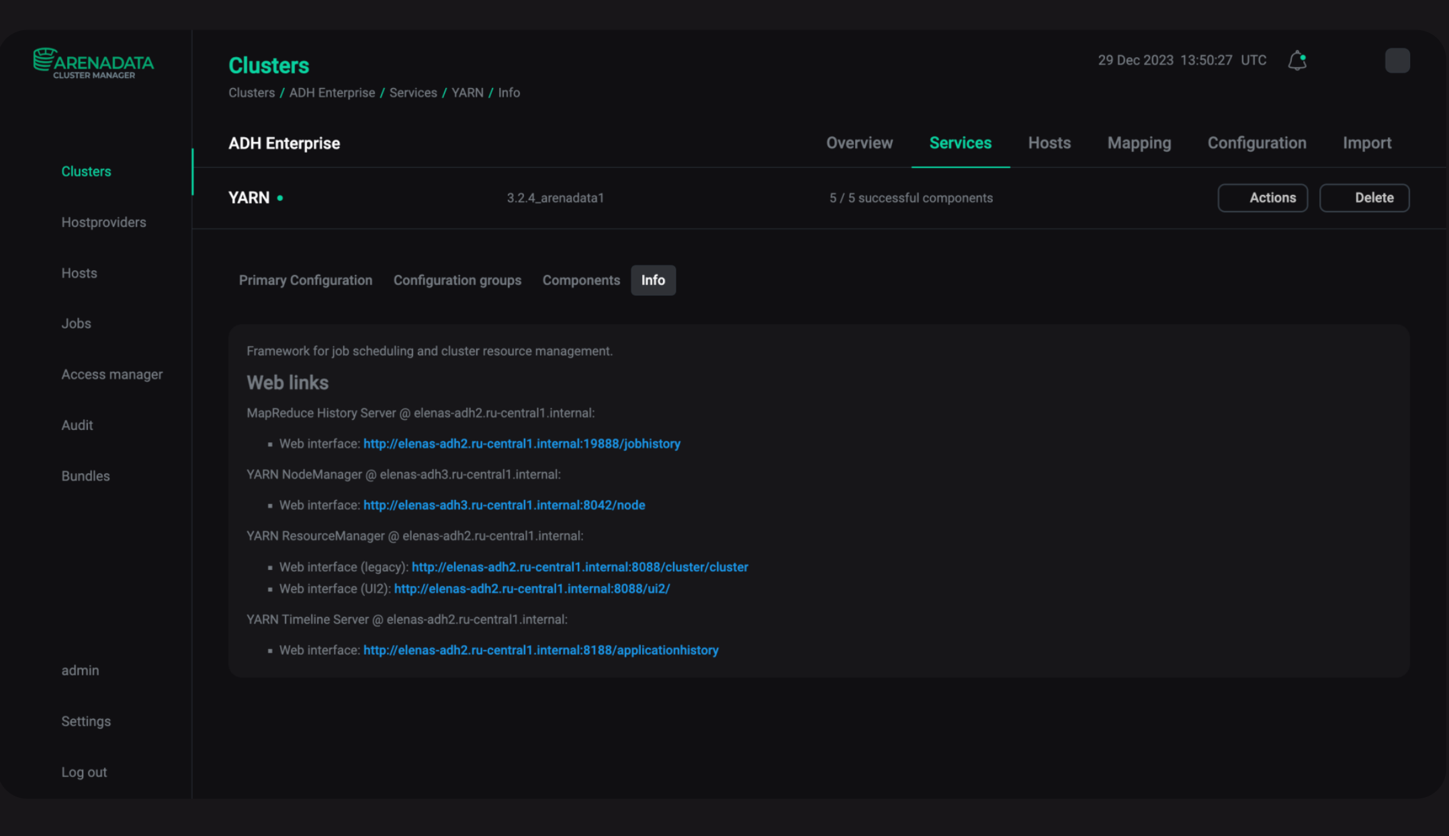
If you have an SSL certificate configured, use *https://* instead of *http://*.

YARN web UIs

Component	URL
MapReduce History Server	http://<IP>:19888/jobhistory
YARN NodeManager	http://<IP>:8042/node
YARN ResourceManager (legacy)	http://<IP>:8088/cluster/cluster
YARN ResourceManager (UI2)	http://<IP>:8088/ui2/
YARN Timeline Server	http://<IP>:8188/applicationhistory/

You can find links to YARN UIs in ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click ***YARN***.
3. Select ***Info***.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH Enterprise / Services / YARN / Info
29 Dec 2023 13:50:27 UTC
ADH Enterprise
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
YARN
3.2.4_arenadata1
5 / 5 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info

Framework for job scheduling and cluster resource management.

Web links

MapReduce History Server @ elenas-adh2.ru-central1.internal:

- Web interface: http://elenas-adh2.ru-central1.internal:19888/jobhistory

YARN NodeManager @ elenas-adh3.ru-central1.internal:

- Web interface: http://elenas-adh3.ru-central1.internal:8042/node

YARN ResourceManager @ elenas-adh2.ru-central1.internal:

- Web interface (legacy): http://elenas-adh2.ru-central1.internal:8088/cluster/cluster
- Web interface (UI2): http://elenas-adh2.ru-central1.internal:8088/ui2/

YARN Timeline Server @ elenas-adh2.ru-central1.internal:

- Web interface: http://elenas-adh2.ru-central1.internal:8188/applicationhistory

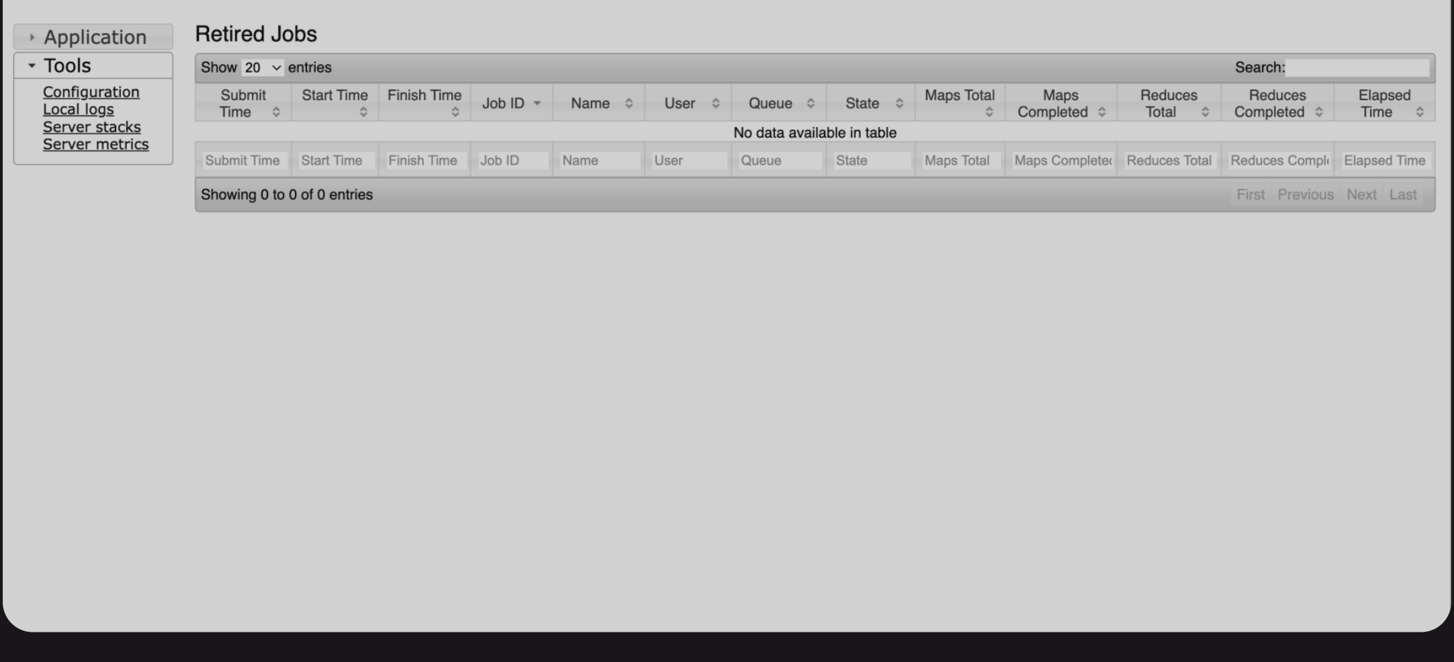
YARN service web interface links in ADCM

MapReduce History Server

The main MapReduce History Server UI page has a navigation menu on the left with two sections: ***Applicaion***, which contains information about the build version and the server's start time, and ***Tools***.

The ***Tools*** section contains links to the following information:

- ***Configuration*** — an XML file that includes all HDFS and YARN configuration files;
- ***Local logs*** — the list of MapReduce History Server logs located on the host;
- ***Server stacks*** — the list of active threads and their status;
- ***Server metrics*** — the list of JMX metrics.



hadoop
JobHistory
Application
Tools
Configuration
Local logs
Server stacks
Server metrics
Retired Jobs
Show 20 entries
Search:
First Previous Next Last

Submit Time	Start Time	Finish Time	Job ID	Name	User	Queue	State	Maps Total	Maps Completed	Reduces Total	Reduces Completed	Elapsed Time
No data available in table												

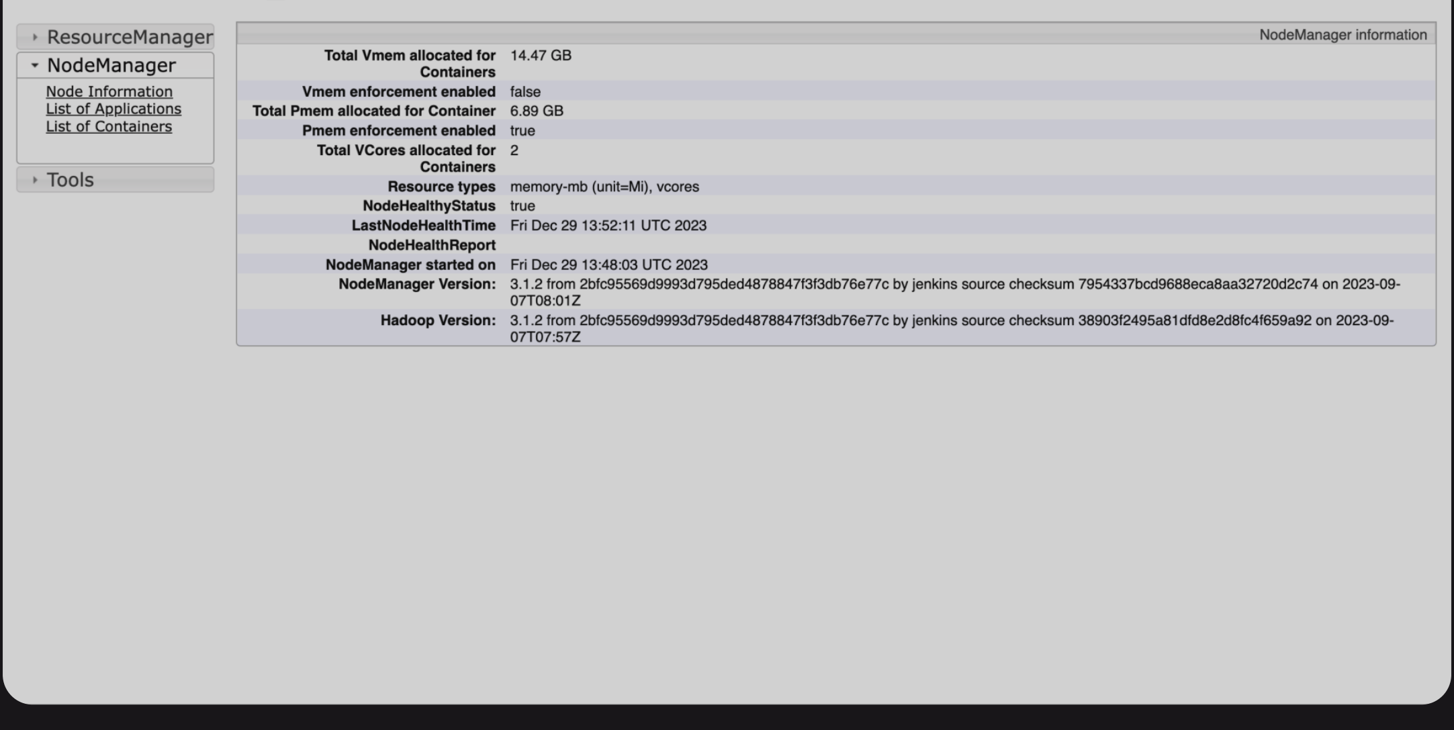
Showing 0 to 0 of 0 entries

MapReduce History Server: main page

NodeManager

The main NodeManager UI page has a navigation menu on the left with three sections:

- ***ResourceManager*** — contains a link to the ResourceManager UI.
- ***NodeManager*** — contains NodeManager information, the list of running applications, and the list of running containers.
- ***Tools*** — contains links to the following information:
  - ***Configuration*** — an XML file that includes all HDFS and YARN configuration files;
  - ***Local logs*** — the list of NodeManager logs located on the host;
  - ***Server stacks*** — the list of active threads and their status;
  - ***Server metrics*** — the list of JMX metrics.



hadoop
ResourceManager
NodeManager
Node Information
List of Applications
List of Containers
Tools
NodeManager information

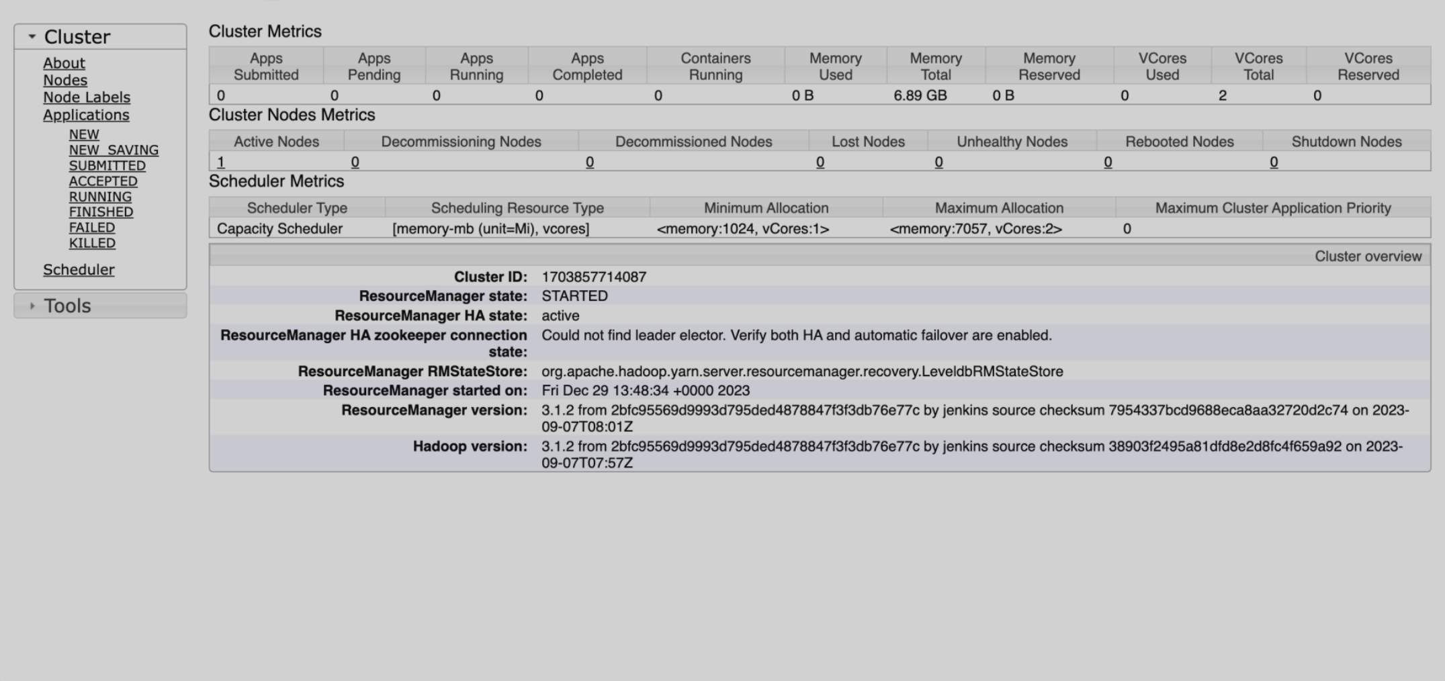
Total Vmem allocated for Containers 14.47 GB
Vmem enforcement enabled false
Total Pmem allocated for Container 6.89 GB
Pmem enforcement enabled true
Total VCores allocated for Containers 2
Resource types memory-mb (unit=Mi), vcores
NodeHealthyStatus true
LastNodeHealthTime Fri Dec 29 13:52:11 UTC 2023
NodeHealthReport
NodeManager started on Fri Dec 29 13:48:03 UTC 2023
NodeManager Version: 3.1.2 from 2bfc95569d9993d795ded4878847f3f3db76e77c by jenkins source checksum 7954337bcd9688eca8aa32720d2c74 on 2023-09-07T08:01Z
Hadoop Version: 3.1.2 from 2bfc95569d9993d795ded4878847f3f3db76e77c by jenkins source checksum 38903f2495a81dfd8e2d8fc4f659a92 on 2023-09-07T07:57Z

NodeManager UI: main page

ResourceManager

There are two ResourceManager UIs:

- Legacy UI: *http://<IP>:8088/cluster/cluster*.
- UI2: *http://<IP>:8088/ui2/*.



hadoop
About the Cluster
Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools
Cluster overview

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	6.89 GB	0 B	0	2	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

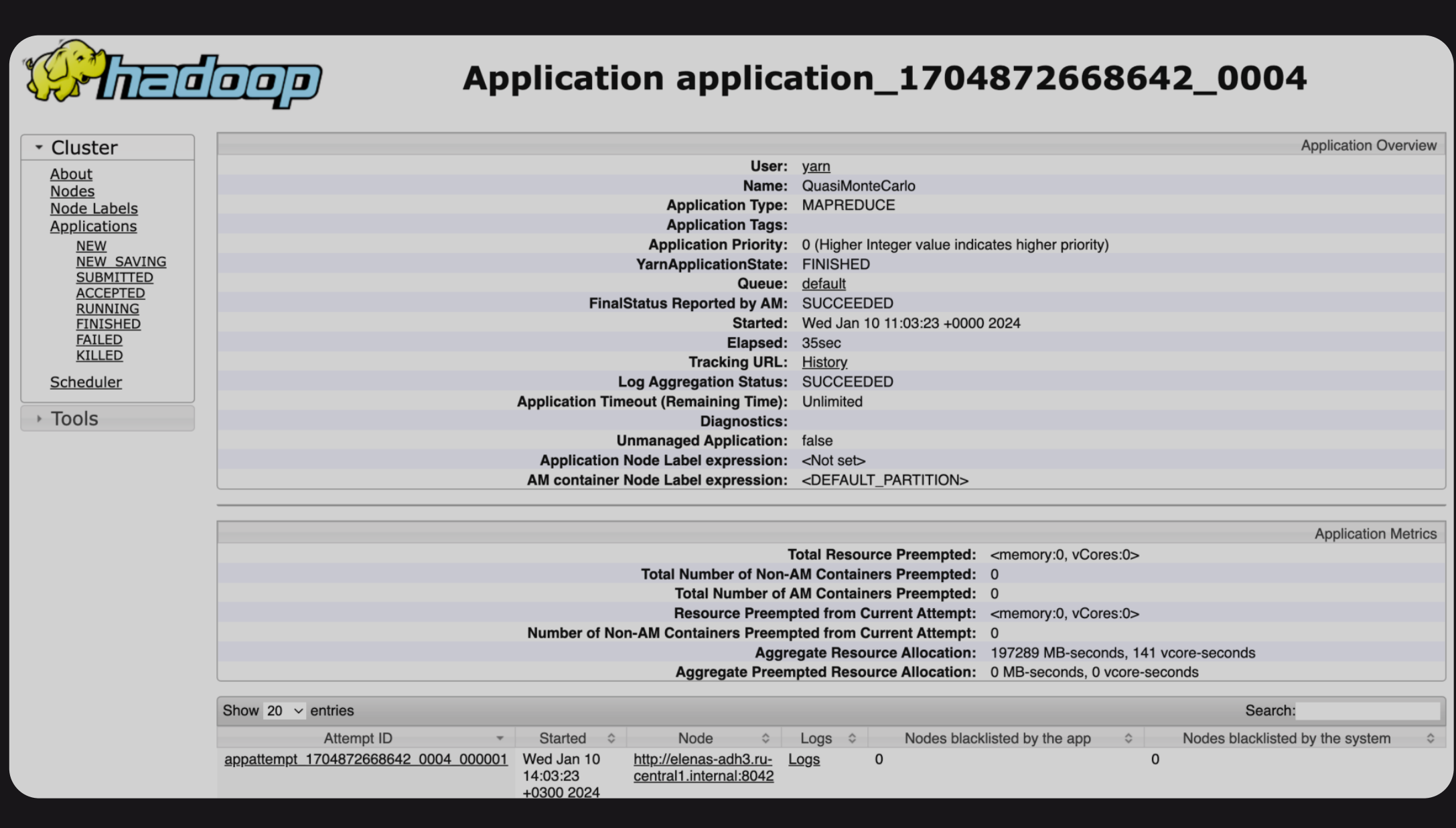
Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:7057, vCores:2>	0

Cluster ID: 1703857714087
ResourceManager state: STARTED
ResourceManager HA state: active
ResourceManager HA zookeeper connection state: Could not find leader elector. Verify both HA and automatic failover are enabled.
ResourceManager RMStateStore: org.apache.hadoop.yarn.server.resourcemanager.recovery.LeveldbRMStateStore
ResourceManager started on: Fri Dec 29 13:48:34 +0000 2023
ResourceManager version: 3.1.2 from 2bfc95569d9993d795ded4878847f3f3db76e77c by jenkins source checksum 7954337bcd9688eca8aa32720d2c74 on 2023-09-07T08:01Z
Hadoop version: 3.1.2 from 2bfc95569d9993d795ded4878847f3f3db76e77c by jenkins source checksum 38903f2495a81dfd8e2d8fc4f659a92 on 2023-09-07T07:57Z

Legacy ResourceManager UI: main page

Both web interfaces provide information about the cluster:

- ***About*** — contains information about the cluster and the list of its metrics as well as the nodes and scheduler metrics.
- ***Nodes*** — shows NodeManager information and metrics table from the ***About*** section.
- ***Node Labels*** — lists node labels of the cluster and information about them.
- ***Applications*** — contains the list of applications with information about them and links to each application's page.



hadoop
Application application_1704872668642_0004
Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools
Application Overview
Application Metrics

User: yarn
Name: QuasiMonteCarlo
Application Type: MAPREDUCE
Application Tags:
Application Priority: 0 (Higher Integer value indicates higher priority)
YarnApplicationState: FINISHED
Queue: default
FinalStatus Reported by AM: SUCCEEDED
Started: Wed Jan 10 11:03:23 +0000 2024
Elapsed: 35sec
Tracking URL: History
Log Aggregation Status: SUCCEEDED
Application Timeout (Remaining Time): Unlimited
Diagnostics:
Unmanaged Application: false
Application Node Label expression: <Not set>
AM container Node Label expression: <DEFAULT_PARTITION>

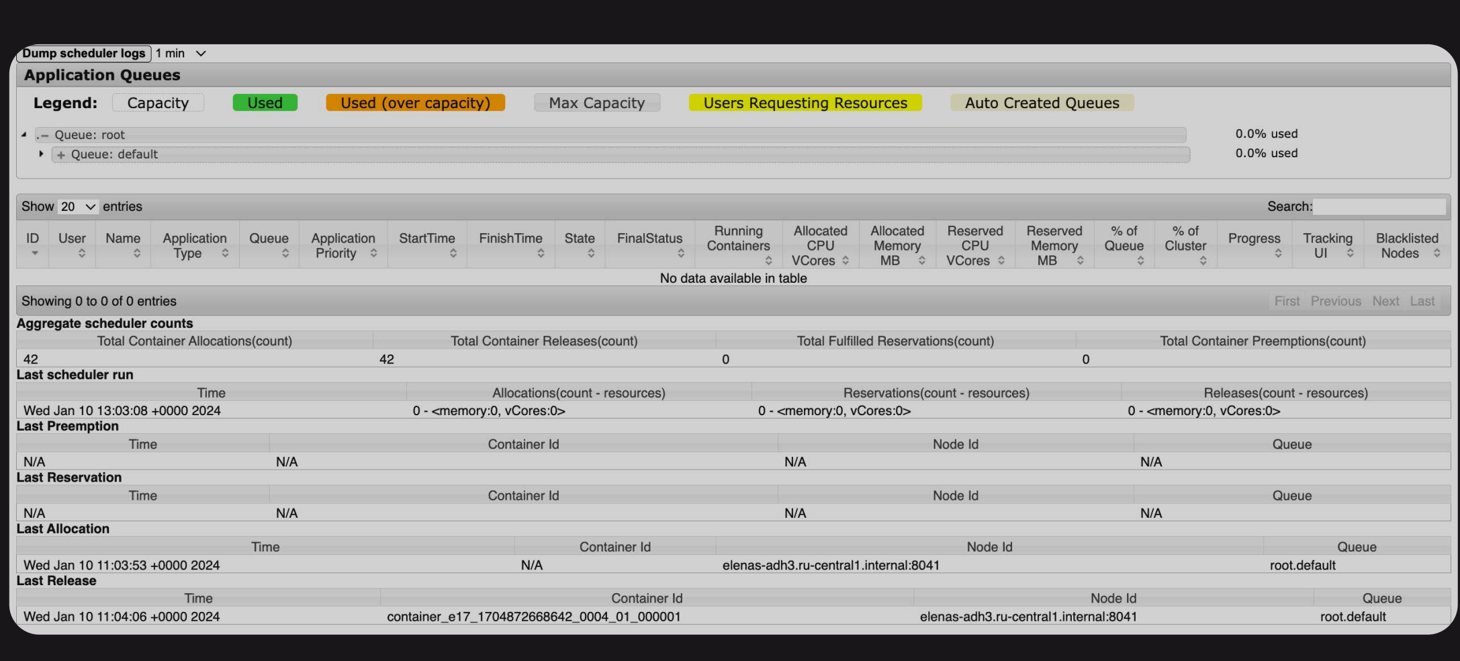
Total Resource Preempted: <memory:0, vCores:0>
Total Number of Non-AM Containers Preempted: 0
Total Number of AM Containers Preempted: 0
Resource Preempted from Current Attempt: <memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt: 0
Aggregate Resource Allocation: 197289 MB-seconds, 141 vcore-seconds
Aggregate Preempted Resource Allocation: 0 MB-seconds, 0 vcore-seconds

Show 20 entries Search:

Attempt ID	Started	Node	Logs	Nodes blacklisted by the app	Nodes blacklisted by the system
appattempt_1704872668642_0004_000001	Wed Jan 10 14:03:23 +0300 2024	http://elenas-adh3.ru-central1.internal:8042	Logs	0	0

Legacy ResourceManager UI: application page

- ***Scheduler*** — displays application queues information and metrics table from the ***About*** section.



Dump scheduler logs
1 min
Application Queues
Legend:
Capacity
Used
Used (over capacity)
Max Capacity
Users Requesting Resources
Auto Created Queues
Queue: root
Queue: default
0.0% used
0.0% used
Show 20 entries
Search:
First Previous Next Last

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Reserved CPU VCores	Reserved Memory MB	% of Queue	% of Cluster	Progress	Tracking UI	Blacklisted Nodes
No data available in table																			

Showing 0 to 0 of 0 entries

Aggregate scheduler counts

Total Container Allocations(count)	Total Container Releases(count)	Total Fulfilled Reservations(count)	Total Container Preemptions(count)
42	42	0	0

Last scheduler run

Time	Allocations(count - resources)	Reservations(count - resources)	Releases(count - resources)
Wed Jan 10 13:03:08 +0000 2024	0 - <memory:0, vCores:0>	0 - <memory:0, vCores:0>	0 - <memory:0, vCores:0>

Last Preemption

Time	Container Id	Node Id	Queue
N/A	N/A	N/A	N/A

Last Reservation

Time	Container Id	Node Id	Queue
N/A	N/A	N/A	N/A

Last Allocation

Time	Container Id	Node Id	Queue
Wed Jan 10 11:03:53 +0000 2024	N/A	elenas-adh3.ru-central1.internal:8041	root.default

Last Release

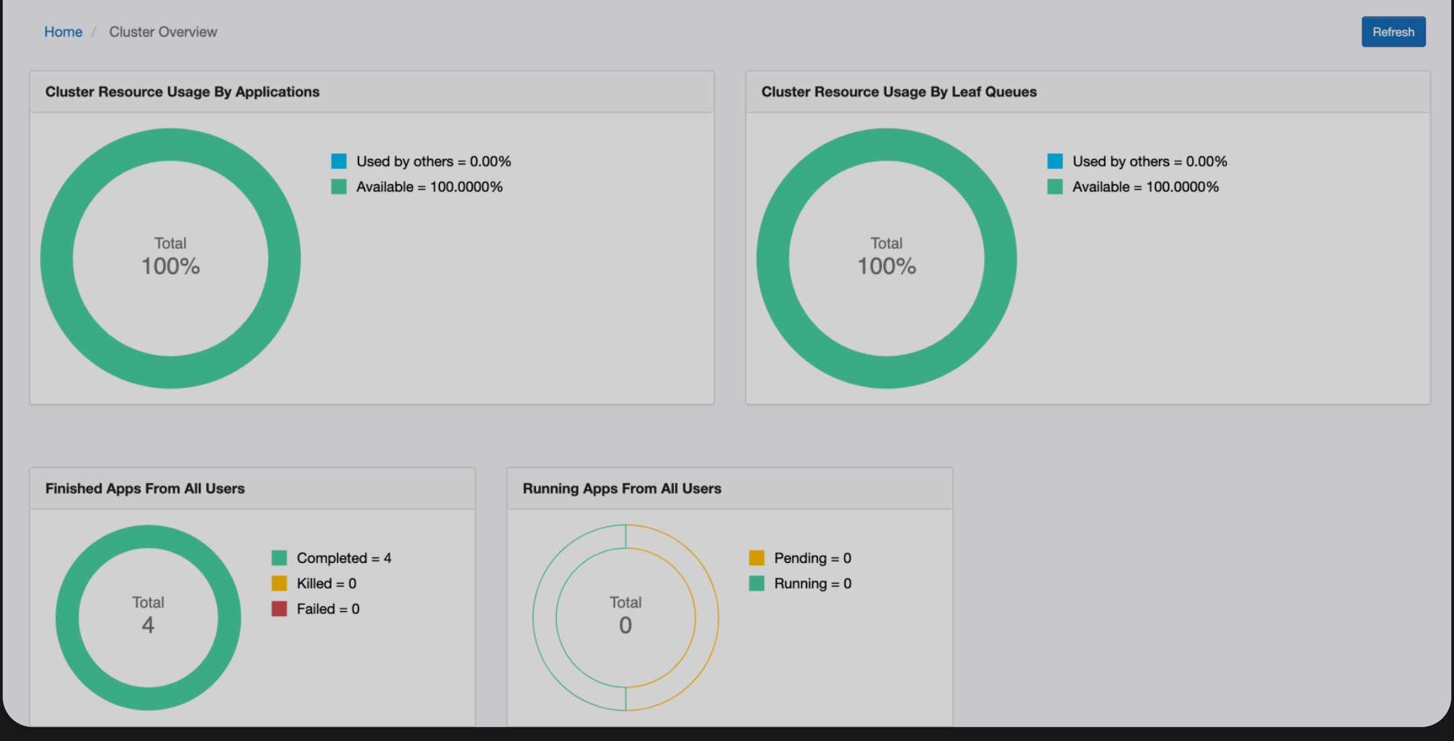
Time	Container Id	Node Id	Queue
Wed Jan 10 11:04:06 +0000 2024	container_e17_1704872668642_0004_01_000001	elenas-adh3.ru-central1.internal:8041	root.default

Legacy ResourceManager UI: Application Queues table

Same as other YARN web interfaces, both ResourceManager UIs contain links to YARN tools:

- ***Configuration*** — an XML file that includes all HDFS and YARN configuration files;
- ***Local logs*** — the list of YARN logs located on the host;
- ***Server stacks*** — the list of active threads and their status;
- ***Server metrics*** — the list of JMX metrics.

The new ResourceManager web interface, UI2, provides the same information but in the form of tabs instead of links. Additionally, some information from tables is represented as graphs.

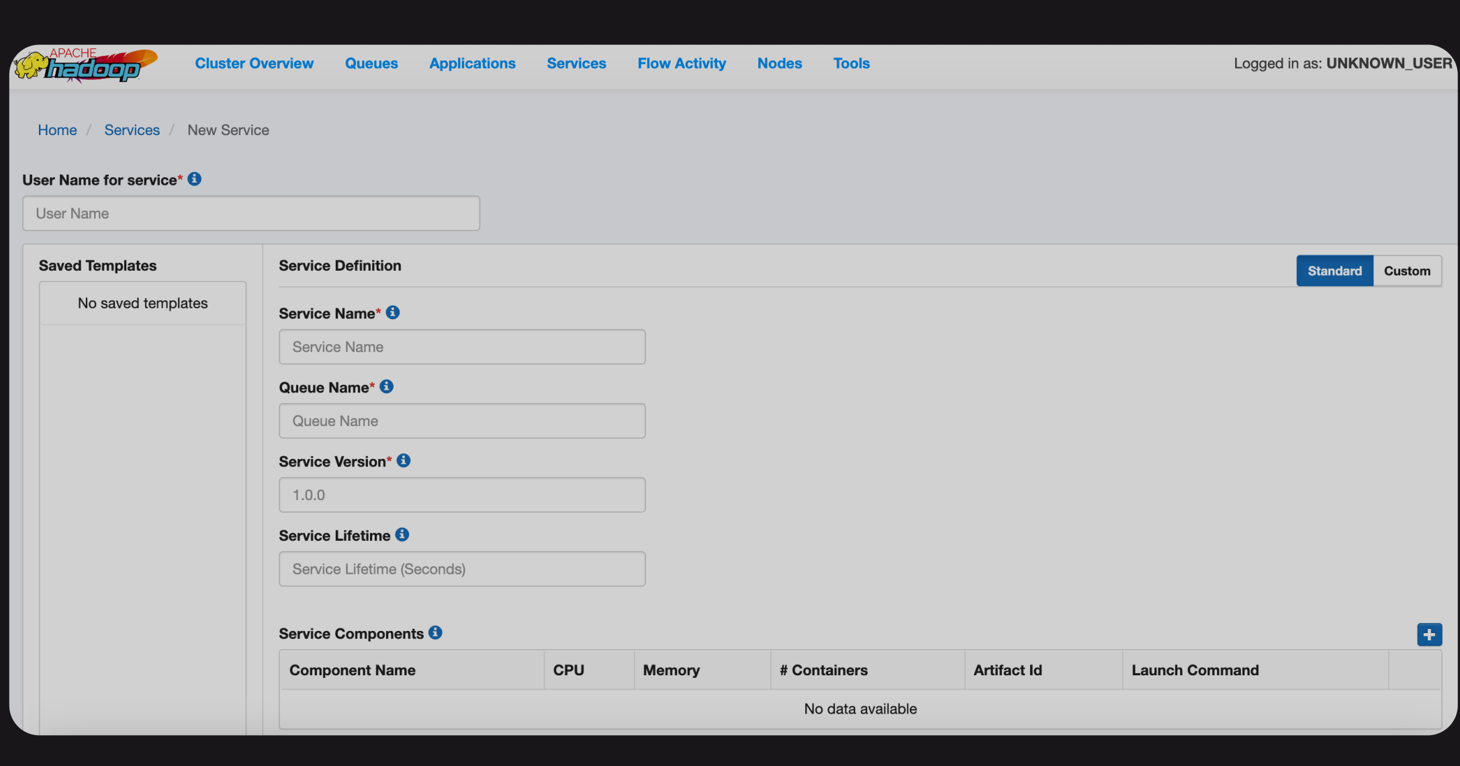


APACHE hadoop
Cluster Overview
Queues
Applications
Services
Flow Activity
Nodes
Tools
Logged in as: UNKNOWN_USER
Home / Cluster Overview
Refresh
Cluster Resource Usage By Applications
Used by others = 0.00%
Available = 100.0000%
Total 100%
Cluster Resource Usage By Leaf Queues
Used by others = 0.00%
Available = 100.0000%
Total 100%
Finished Apps From All Users
Completed = 4
Killed = 0
Failed = 0
Total 4
Running Apps From All Users
Pending = 0
Running = 0
Total 0

ResourceManager UI2: main page

The tabs from the ResourceManager UI2 contain the following information:

- ***Cluster Overview*** — cluster resource statistics and application metrics as graphs.
- ***Queues*** — scheduler queues graphs and resource statistics.
- ***Applications*** — the list of applications with information about them and links to each application's page.
- ***Services*** — the list of services that run applications and a tool for creating them.



APACHE hadoop
Cluster Overview
Queues
Applications
Services
Flow Activity
Nodes
Tools
Logged in as: UNKNOWN_USER
Home / Services / New Service
Saved Templates
No saved templates
Service Definition
Standard
Custom

User Name for service*: User Name

Service Name*: Service Name

Queue Name*: Queue Name

Service Version*: 1.0.0

Service Lifetime: Service Lifetime (Seconds)

Service Components

Component Name	CPU	Memory	# Containers	Artifact Id	Launch Command
No data available					

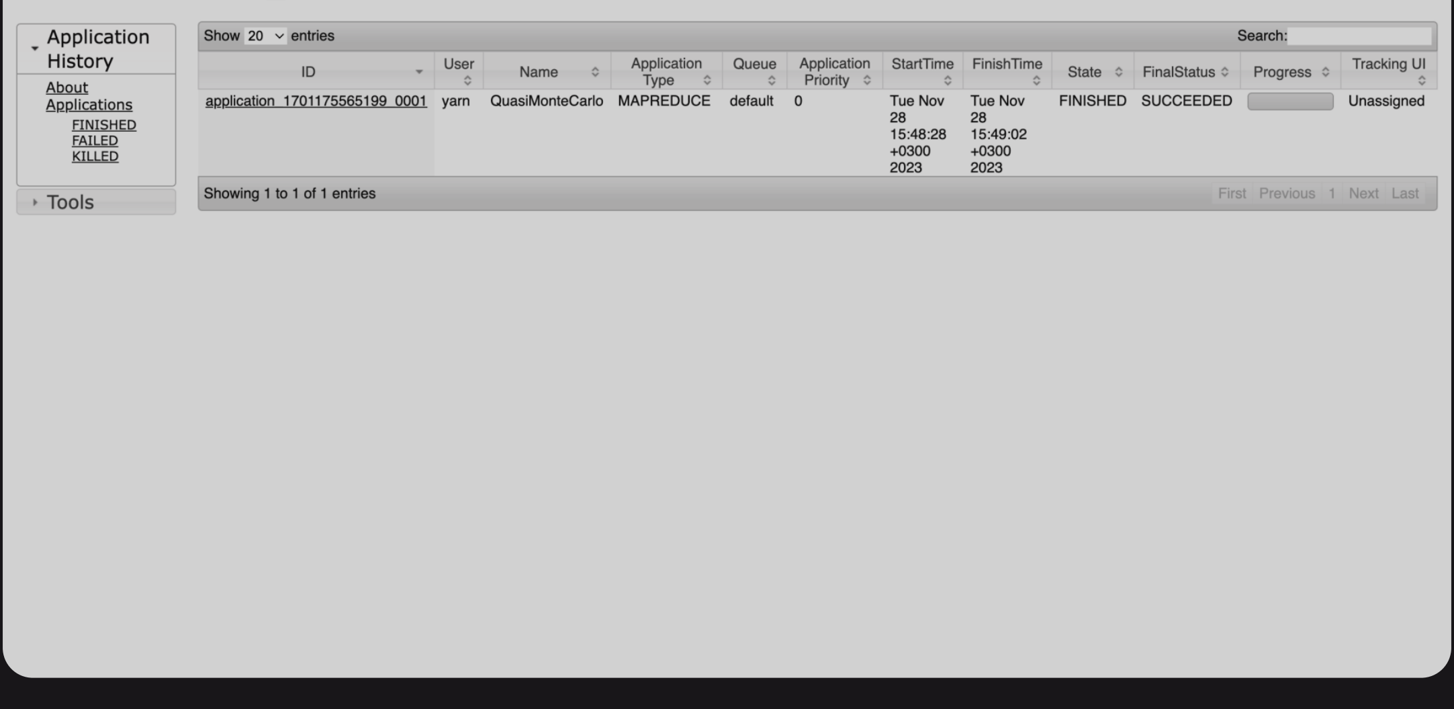
ResourceManager UI2: service creation page

- ***Flow Activity*** — timeline server information page.
- ***Nodes*** — details about the cluster nodes that host NodeManagers.
- ***Tools*** — a web interface for viewing YARN logs and configuration files.

Timeline server

The main Timeline server UI page has a navigation menu on the left with two sections:

- ***Application History*** — contains information about the timeline server and the list of all launched applications that can be filtered by state: successfully finished, failed, and killed (stopped).
- ***Tools*** — contains links to the following information:
  - ***Configuration*** — an XML file that includes all HDFS and YARN configuration files;
  - ***Local logs*** — the list of Timeline server logs located on the host;
  - ***Server stacks*** — the list of active threads and their status;
  - ***Server metrics*** — the list of JMX metrics.



hadoop
All Applications
Application History
About
Applications
FINISHED
FAILED
KILLED
Tools
Show 20 entries
Search:
First Previous 1 Next Last

ID	User	Name	Application Type	Queue	Application Priority	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1701175565199_0001	yarn	QuasiMonteCarlo	MAPREDUCE	default	0	Tue Nov 28 15:48:28 +0300 2023	Tue Nov 28 15:49:02 +0300 2023	FINISHED	SUCCEEDED		Unassigned

Showing 1 to 1 of 1 entries

Timeline server UI: main page

Connect to YARN via CLI

Elena Kostyuchenko

Contents

To Table of Contents

- Connect
- Useful commands
  - Monitor cluster load
  - List running applications
  - Terminate a frozen application

Connect

YARN provides a command line interface implemented as a bin/yarn script. To interact with YARN via CLI, connect to a cluster host with YARN via SSH and run the desired YARN command.

For example, check the YARN version by running:

```
$ yarn version
```

Output example:

```
Hadoop 3.1.2
Source code repository git@ssh.gitlab.adsw.io:arenadata/infrastructure/code/ci/prj_adh.git -r
2bfc95569d9993d795ded4878847f3f3db76e77c
Compiled by jenkins on 2023-09-07T07:57Z
Compiled with protoc 2.5.0
From source with checksum 38903f2495a81dfd8e2d8fc4f659a92
This command was run using /usr/lib/hadoop/hadoop-common-3.1.2.jar
```

Running `yarn` without any arguments prints the list of all YARN commands.

The commands have the following syntax:

```
$ yarn [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [SUB_COMMAND] [COMMAND_OPTIONS]
```

Useful commands

Here are examples of typical commands used when working with YARN.

Monitor cluster load

To see the dynamic list of running applications and the resources they consume, run:

```
$ yarn top
```

Output example:

```
YARN top - 11:54:03, up 0d, 2:9, 1 active users, queue(s): root
NodeManager(s): 1 total, 1 active, 0 unhealthy, 0 decommissioned, 0 lost, 0 rebooted
Queue(s) Applications: 1 running, 1 submitted, 0 pending, 0 completed, 0 killed, 0 failed
Queue(s) Mem(GB): 0 available, 6 allocated, 9 pending, 0 reserved
Queue(s) VCores: 0 available, 5 allocated, 9 pending, 0 reserved
Queue(s) Containers: 5 allocated, 9 pending, 0 reserved

                  APPLICATIONID USER             TYPE      QUEUE PRIOR   #CONT  #RCONT  VCORES RVCORES     MEM    RMEM  VCORESECS
MEMSECS %PROGR       T
 application_1705311887839_0001 yarn        mapreduce    default     0       5       0       5       0      6G      0G         57
75  16.25   00:00
```

This command prints out a list of applications that is constantly updating. To see a static list of running applications, use the `application -list` command.

List running applications

To see the list of applications running on the host, run:

```
$ yarn application -list
```

You can filter applications in the list by using the following options:

- `-appTypes` — filter by application types;
- `-appStates` — filter by application states;
- `-appTags` — filter by application tags.

Output example:

```
2024-01-15 11:56:01,792 INFO client.RMProxy: Connecting to ResourceManager at elenas-adh2.ru-central1.internal/127.0.0.1:8032
2024-01-15 11:56:02,241 INFO client.AHSProxy: Connecting to Application History server at elenas-adh2.ru-central1.internal/127.0.
0.1:10200
Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTED, RUNNING] and tags: []):1
                Application-Id      Application-Name        Application-Type          User           Queue
State             Final-State        Progress                         Tracking-URL
application_1705311887839_0002       QuasiMonteCarlo               MAPREDUCE          yarn         default
RUNNING               UNDEFINED              5%  http://elenas-adh3.ru-central1.internal:33585
```

Terminate a frozen application

To terminate an application that stopped responding, run:

```
$ yarn application -kill <Application ID>
```

Where `<Application ID>` is the ID of the application you want to kill. You can provide a list of space-separated application IDs.

For more detailed information on the above commands, see YARN CLI.

Connect to YARN via REST API

Elena Kostyuchenko

Contents

To Table of Contents

Overview

ResourceManager API

NodeManager

Timeline Server

Overview

Hadoop provides REST APIs for all YARN components. It enables URI access to the cluster, nodes, applications, and application history.

Most YARN APIs support only `GET` requests, but some YARN components, like ResourceManager, support `POST`. The only fields used in the request header are `Accept` and `Accept-Encoding`; all other header fields are ignored.

In the `Accept` field, you can pass `XML` and `JSON` to indicate which format you expect in the response. The default format is JSON. In the `Accept-Encoding` field, you can specify `gzip` to get a gzip-compressed output.

The URI syntax for YARN REST APIs is as follows:

http://<IP:PORT>/ws/<version>/<resourcepath>

Where:

- `<IP:PORT>` — the IP or FQDN of the component host and its port;
- `<version>` — the API version;
- `<resourcepath>` — a path that defines which API function to call.

Authentication

Using the API requires authentication. If your cluster has no security configured, add the `?user.name=<user>` option to your requests. Where `<user>` is the username under which you want to run the request. For example, `admin`.

If security is on, authentication method is different for each security model.

When Kerberos SPNEGO is on, use the `--negotiate` and `-u` options:

```
$ curl -i --negotiate -u : "http://<HOST>:<HTTP_PORT>/ws/<version>/<resourcepath>"
```

When the authentication with the Hadoop delegation token is on, pass the token in place of the username:

```
$ curl -i "http://<HOST>:<HTTP_PORT>/ws/<version>/<resourcepath>?delegation=<TOKEN>"
```

For more information about authentication methods, see Authentication for Hadoop HTTP web-consoles.

The example request to the Timeline server:

```
$ curl http://127.0.0.1:8188/ws/v1/timeline?user.name=admin
```

The response contains information about the server:

```
{"About":"Timeline API","timeline-service-version":"3.1.2","timeline-service-build-version":"3.1.2 from
2bfc95569d9993d795ded4878847f3f3db76e77c by jenkins source checksum 7954337bcd9688eca8aa32720d2c74","timeline-service-version-
built-on":"2023-09-07T08:01Z","hadoop-version":"3.1.2","hadoop-build-version":"3.1.2 from 2bfc95569d9993d795ded4878847f3f3db76e77c
by jenkins source checksum 38903f2495a81dfd8e2d8fc4f659a92","hadoop-version-built-on":"2023-09-07T07:57Z"}
```

NOTE

The fields within the response body are not limited in number and have no specific order. When developing applications that use the YARN REST API, use parsing routines that don't depend on a particular order.

ResourceManager API

The ResourceManager REST API enables access to information about the cluster and allows managing applications. The full API specification is available at ResourceManager REST API's.

Here's an example of using ResourceManager REST API for submitting a new application:

1. Request an ID for your application by calling:

```
$ curl -X POST http://127.0.0.1:8088/ws/v1/cluster/apps/new-application?user.name=admin
```

The server's response:

```
{"application-id":"application_1705573015955_0001","maximum-resource-capability":
{"memory":7057,"vCores":2,"resourceInformations":{"resourceInformation":
[{"maximumAllocation":9223372036854775807,"minimumAllocation":0,"name":"memory-
mb","resourceType":"COUNTABLE","units":"Mi","value":7057},
{"maximumAllocation":9223372036854775807,"minimumAllocation":0,"name":"vcores","resourceType":"COUNTABLE","units":"","value":2}]}
}}[admin@elenas-adh2 ~]$
```

2. The response contains the application ID and the information on available resources. Create a JSON file with parameters of your application and pass it to the ResourceManager:

```
$ curl -v -X POST -d @example-app.json -H "Content-type: application/json"  http://127.0.0.1:8088/ws/v1/cluster/apps/?user.
name=admin
```

Application information example

```
 {
    "application-id":"application_1705573015955_0001",
    "application-name":"test",
    "am-container-spec":
    {
       "local-resources":
       {
          "entry":
         [
            {
               "key":"hadoop-mapreduce-examples.jar",
               "value":
               {
                  "resource":"hdfs://127.0.0.2:8020/tmp/hadoop-mapreduce-examples.jar",
                  "type":"FILE",
                  "visibility":"APPLICATION",
                  "size": "30897",
                  "timestamp": "1405452071209"
               }
            }
          ]
       },
       "commands":
       {
          "command":"{{JAVA_HOME}}/bin/java -Xmx10m org.apache.hadoop.yarn.applications.distributedshell.ApplicationMaster --
container_memory 10 --container_vcores 1 --num_containers 1 --priority 0 1><LOG_DIR>/AppMaster.stdout 2><LOG_DIR>/AppMaster.
stderr"
       },
       "environment":
       {
          "entry":
          [
             {
                "key": "DISTRIBUTEDSHELLSCRIPTTIMESTAMP",
                "value": "1405459400754"
             },
             {
                "key": "CLASSPATH",
                "value": "{{CLASSPATH}}<CPS>./*<CPS>{{HADOOP_CONF_DIR}}<CPS>{{HADOOP_COMMON_HOME}}/share/hadoop/common/*<CPS>
{{HADOOP_COMMON_HOME}}/share/hadoop/common/lib/*<CPS>{{HADOOP_HDFS_HOME}}/share/hadoop/hdfs/*<CPS>
{{HADOOP_HDFS_HOME}}/share/hadoop/hdfs/lib/*<CPS>{{HADOOP_YARN_HOME}}/share/hadoop/yarn/*<CPS>
{{HADOOP_YARN_HOME}}/share/hadoop/yarn/lib/*<CPS>./log4j.properties"
             },
             {
                "key": "DISTRIBUTEDSHELLSCRIPTLEN",
                "value": "6"
             },
             {
                "key": "DISTRIBUTEDSHELLSCRIPTLOCATION",
                "value": "hdfs://127.0.0.2:8020/tmp/hadoop-mapreduce-examples.jar"
             }
          ]
       }
    },
    "unmanaged-AM":"false",
    "max-app-attempts":"2",
    "resource":
    {
       "memory":"1024",
       "vCores":"1"
    },
    "application-type":"YARN",
    "keep-containers-across-application-attempts":"false"
  }
```

If the application has been submitted successfully, the server returns the `HTTP/1.1 202 Accepted` code and the application's location URL, which can be used to check the application status.

To enable cross-origin support (CORS) for the ResourceManager only, set the following configuration parameters:

- *core-site.xml* — set `hadoop.http.filter.initializers` to `org.apache.hadoop.security.HttpCrossOriginFilterInitializer`;
- *yarn-site.xml* — set `yarn.resourcemanager.webapp.cross-origin.enabled` to `true`.

NodeManager

The NodeManager REST API provides information about the node, applications, and containers running on that node. The full API specification is available at NodeManager REST API's.

Here's an example of the command that prints information about a node:

```
$ curl -X GET http://<IP>:8042/ws/v1/node/info?user.name=admin
```

Where `<IP>` is the IP address or FQDN of the host.

To enable cross-origin support (CORS) for the NodeManager, set the following configuration parameters:

- *core-site.xml* — set `hadoop.http.filter.initializers` to `org.apache.hadoop.security.HttpCrossOriginFilterInitializer`;
- *yarn-site.xml* — set `yarn.nodemanager.webapp.cross-origin.enabled` to `true`.

Timeline Server

The Timeline Server REST API supports two methods: `GET` and `POST`. Currently, there are two available API versions:

- Timeline Server REST API v1
- Timeline Server REST API v2

Here's an example of the command that prints the list of domains for the specified owner:

```
$ curl -X GET http://<IP>:8188/ws/v1/timeline/domain?owner=<owner>
```

Where `<IP>` is the IP address or FQDN of the host and `<owner>` is the domain owner name.

FairScheduler

Mikhail Serov

Contents

To Table of Contents

- Resource distribution example
- Support of hierarchical queues
- Configuration
  - Set up queues
  - Queue configuration example

FairScheduler is a pluggable scheduler for Hadoop that enables YARN applications to share resources across large clusters so that all applications get an equal share of resources on average over time.

FairScheduler distributes resources among all running applications so that short-lived applications terminate in a reasonable amount of time and long-lived applications do not starve. It organizes applications into queues and distributes resources fairly among these queues.

By default, scheduling fairness decisions are based on memory as the only primary capacity, and the only queue for applications is `default`. You can make it take both memory (mb — megabytes) and CPUs (vcores — virtual CPUs) into account. Applications supplied to the scheduler can request creating new queues as described in Setting up queues.

Resource distribution example

Let’s consider two queues:

- `sales`
- `finance`

At some point, the first job is submitted to the `sales` queue. Being a sole running job, it will get all the resources.

Then another job is submitted to the `finance` queue. This will result in the new job gradually getting half of the resources.

So each job in those queues will equally have 50% of the cluster resources. Now another job is submitted to the `finance` queue. This will result in half of the resources allocated to the `finance` queue to be allocated to this new job.

Two jobs in the `finance` queue will now share resources allocated to the `finance` queue (50% of the total resources) in equal proportions. The sole job in the `sales` queue will use all resources allocated to this queue, that is, also 50% of the cluster resources.

Support of hierarchical queues

FairScheduler supports hierarchical queues. A queue is hierarchical if it can contain subqueues. All queues descend from the root queue, although in the configuration file, you can omit "root" in the full queue name.Available resources are distributed among the children of the `root` queue using the fair scheduling concept. Then the children queues distribute the resources assigned to them to their children in the same fashion.

Configuration

To use FairScheduler in YARN, start with assigning the appropriate scheduler class in the *yarn-site.xml* file:

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
</property>
```

Set up queues

To set up queues, edit the following XML elements in the *fair-scheduler.xml* configuration file:

- `<queue>` represents a queue. A <queue> element can contain other <queue> elements. Some of the most important properties of this element are listed below:
  - `minResources`. Specifies the minimum amount of the resources required for the queue (X mb, Y vcores). If this requirement is not met, the scheduler prioritizes this queue over other queues of the same parent when distributing available resources.
  - `maxResources`. Specifies the maximum amount of the resources that can be allocated to the queue, either in absolute values (X mb, Y vcores) or as a percentage of the cluster resources (X% memory, Y% CPU).
  - `weight`. Influences the proportion of resources the scheduler will share between the queues. The default weight for a queue is `1`. If the queue weight is `2`, then this queue should receive approximately twice as many resources as a queue with the default weight.
  - `schedulingPolicy`. Sets the scheduling policy for the queue. Its value is the name of an object that extends the `org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.SchedulingPolicy` class. Such objects are: `fifo`, `fair`, `drf`, and other. The default value is `fair`.
- `<defaultQueueSchedulingPolicy>`. Sets the default scheduling policy for queues. The default value is `fair`. This element is overridden by the `schedulingPolicy` element in the queue configuration.
- `<queueMaxAppsDefault>`. Sets the default upper limit on the number of running applications in a queue. This element is overridden by the `maxRunningApps` element in the queue configuration.
- `<queuePlacementPolicy>`. Contains a list of rules (represented by elements) that tell the scheduler what queues it must use to place the incoming applications. The rules are applied in the order that they are listed. The scheduler follows the first rule it can satisfy. Every rule has the `create` attribute, which indicates whether the scheduler can create a new queue. The default value of the `create` attribute is `true`. If its value is set to `false` and the rule places applications in a queue that is not configured in the allocation file, the scheduler moves on to the next rule. The possible rules are listed below:
  - `specified`. The application is placed into the queue that the rule specifies explicitly.
  - `user`. The application is placed into the queue with the name of the user who submitted it.
  - `primaryGroup`. The application is placed into the queue with the name of the primary group of the user who submitted it.
  - `secondaryGroupExistingQueue`. The application is placed into the queue with the name that matches a secondary group of the user who submitted it.
  - `nestedUserQueue`. The application is placed into the queue with the name of the user under the queue suggested by the nested rule. In other words, this rule must have a nested rule that specifies the parent queue for the queue where the application can be placed to.

    Starting with ADH 3.3.6.1, the `nestedUserQueue` block should contain only one nested rule. If several rules are nested like in the example below, only the last rule is used (`primaryGroup`):

    ```
    <rule name="nestedUserQueue" create="true">
        <rule name="secondaryGroupExistingQueue" create="false"/>
        <rule name="primaryGroup" create="false"/>
    </rule>
    ```
  - `default`. The application is placed into the queue specified in the `queue` attribute of the default rule. If the `queue` attribute is not specified, the application is placed into the `root.default` queue.
  - `reject`. The application is rejected.

> **NOTE**
>
> For FairScheduler, you can change the scheduler configs (limits, weight, preemption timeout, scheduling policy, etc.) on the fly, without having to run the ***Reload Scheduler config*** command. For this, simply edit the allocation file and the scheduler will reload the file with a 10-15 seconds delay after the file has been modified.

Queue configuration example

This example considers two top-level queues (descending directly from `root`):

- `sales`
- `finance`

Within the `sales` queue, there are two subqueues:

- `apac`
- `emea`

The queues can be set up for a use with FairScheduler as follows:

```
<allocations>
  <queue name="sales">
      <minResources>10000 mb,0vcores</minResources>
      <maxResources>50000 mb,0vcores</maxResources>
      <weight>2.0</weight>
      <schedulingPolicy>fifo</schedulingPolicy>
      <queue name="emea" />
      <queue name="apac" />
  </queue>
  <queue name="finance">
      <minResources>10000 mb,0vcores</minResources>
      <maxResources>70000 mb,0vcores</maxResources>
      <weight>3.0</weight>
      <schedulingPolicy>fair</schedulingPolicy>
  </queue>
  <queuePlacementPolicy>
      <rule name="specified"   />
      <rule name="primaryGroup" create="false" />
      <rule name="default" queue="finance" />
  </queuePlacementPolicy>
</allocations>
```

CapacityScheduler

Mikhail Serov

Contents

To Table of Contents

- Security system
- Hierarchical queues
- Configuration
  - Setting up queues
  - Queue configuration example

CapacityScheduler is a pluggable scheduler that you can use in a Hadoop framework. It improves multi-tenancy of a shared cluster by allocating a certain capacity of the entire cluster to each organization using this cluster.

Security system

CapacityScheduler enables each organization to get its own queue with a portion of the cluster capacity allocated specifically for that queue.

In a shared cluster, security becomes very important. For each queue, there is an access control list (ACL) containing users that can submit applications to individual queues. It also ensures that users cannot view or modify applications run by other users in other queues. Also per-queue and system administrator roles are supported.

Hierarchical queues

CapacityScheduler supports hierarchical queues. A queue is hierarchical if subqueues can be created within this queue. The portion of the cluster resources allocated to the queue can be further distributed among its subqueues. CapacityScheduler has a predefined queue called `root`. All queues in the system are the children of the `root` queue.

An added benefit is that an organization can exceed its queue capacity and use more cluster resources than it was assigned, yet only if the extra capacity is available and not being used by others. This gives organizations more flexibility in a cost-effective manner.

Configuration

To use CapacityScheduler in YARN, start with assigning the appropriate scheduler class in *yarn-site.xml*:

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>
</property>
```

Setting up queues

To set up queues, edit the *capacity-scheduler.xml* configuration file:

- In the `yarn.scheduler.capacity.root.queues` property, specify a list of comma-separated top-level queues.
- To set up subqueues in a queue defined as <queue-path>, in the `yarn.scheduler.capacity.<queue-path>.queues` property of this queue, specify a list of subqueues.
- To configure queue capacity in percentage (%), use the `yarn.scheduler.capacity.<queue-path>.capacity` property of that queue. The sum of capacities for all queues (at each level) must be equal to 100.
- To set up the maximum capacity for a queue in percentage, use the `yarn.scheduler.capacity.<queue-path>.maximum-capacity` property. This property limits the elasticity for applications in the queue. It defaults to `-1`, which disables this limitation.

Queue configuration example

Let us configure two top-level queues (descending directly from `root`):

- `sales`
- `finance`

Within the `sales` queue, there are two subqueues:

- `apac`
- `emea`

The hierarchy of queues in the CapacityScheduler configuration looks as follows:

```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>sales, finance</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.sales.queues</name>
  <value>apac,emea</value>
</property>
```

To give 70% of the queue capacity to `sales` and 30% to `finance`, add the following settings:

```
<property>
  <name>yarn.scheduler.capacity.root.sales.capacity</name>
  <value>70</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.finance.capacity</name>
  <value>30</value>
</property>
```

To allocate 65% of the `sales` queue capacity to `apac` and 35% to `emea`, add the following settings:

```
<property>
  <name>yarn.scheduler.capacity.root.sales.apac.capacity</name>
  <value>65</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.sales.emea.capacity</name>
  <value>35</value>
</property>
```

To ensure that the `sales` queue doesn't use more than 80% of the cluster resources (even if the resources are available), add the following:

```
<property>
  <name>yarn.scheduler.capacity.root.sales.maximum-capacity</name>
  <value>80</value>
</property>
```

If a user, for example, `yarn` supplies some applications without specifying a queue and there is no `default` queue, then you must specify a default queue for that user. The following snippet makes the `marketing` queue the default one for the `yarn` user:

```
<property>
   <name>yarn.scheduler.capacity.queue-mappings</name>
   <value>u:yarn:marketing</value>
</property>
```

In this configuration:

- `u` means that this a user property (`g` means a property for a group);
- `yarn` is the user name;
- `marketing` is the name of the default queue for the specified user.

YARN high availability overview

Konstantin Alpashkin, Mikhail Serov

Contents

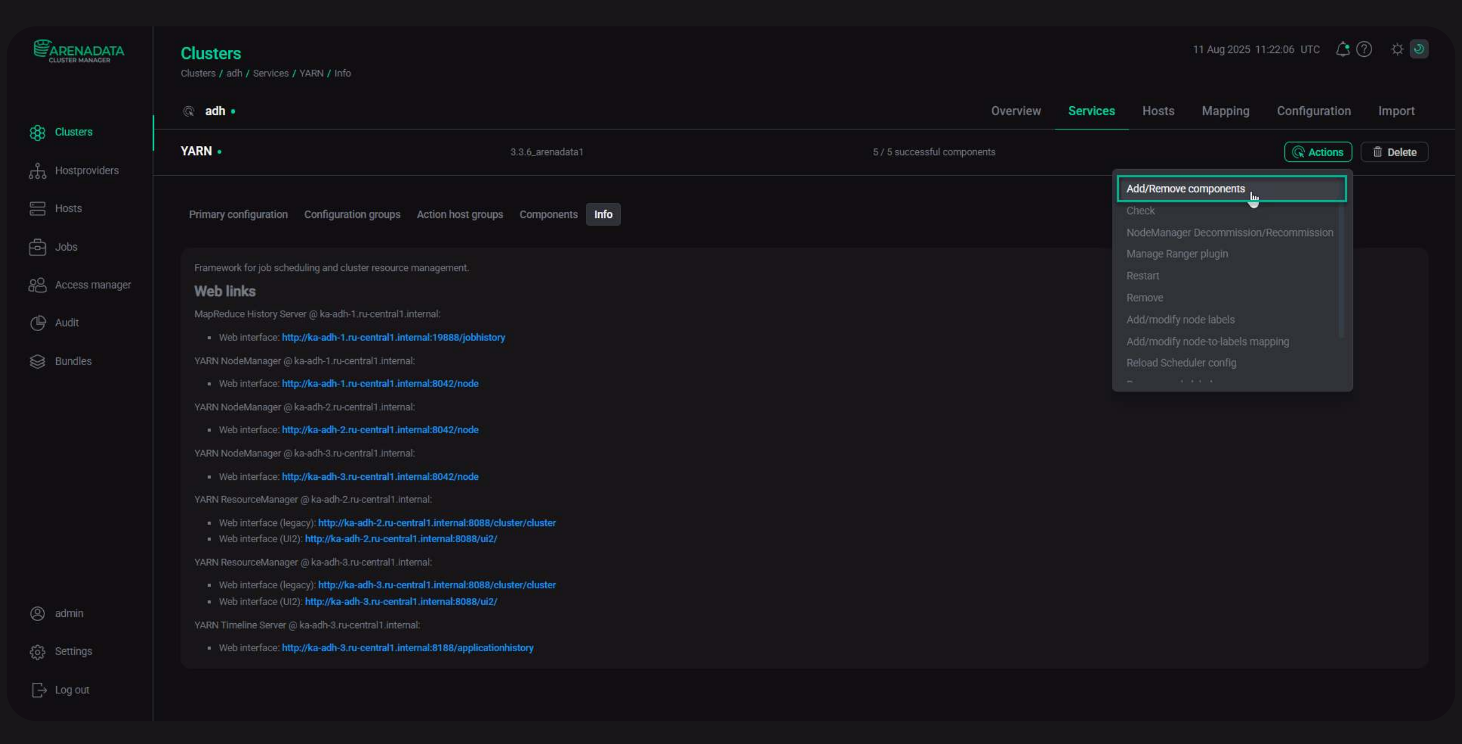
To Table of Contents

[Architecture]
[Automatic failover]
[Leader Election algorithm]
  [Election algorithm]
[Client, ApplicationMaster, and NodeManager on RM failover]

ResourceManager (RM) is a YARN component responsible for allocating resources like CPU/RAM in a Hyperwave cluster and scheduling applications such as MapReduce jobs.

Two or more RMs can work in the high availability (HA) mode to provide uninterrupted operation even when a cluster node with one RM goes down. When an RM becomes unreachable, one of the standby RM instances automatically becomes active based on the election algorithm.

Starting ADH 2.1.10.b1 no explicit actions are required to enable HA for ResourceManagers. The HA mode activates automatically when you add two or more YARN ResourceManager components using the ADCM UI as shown below.

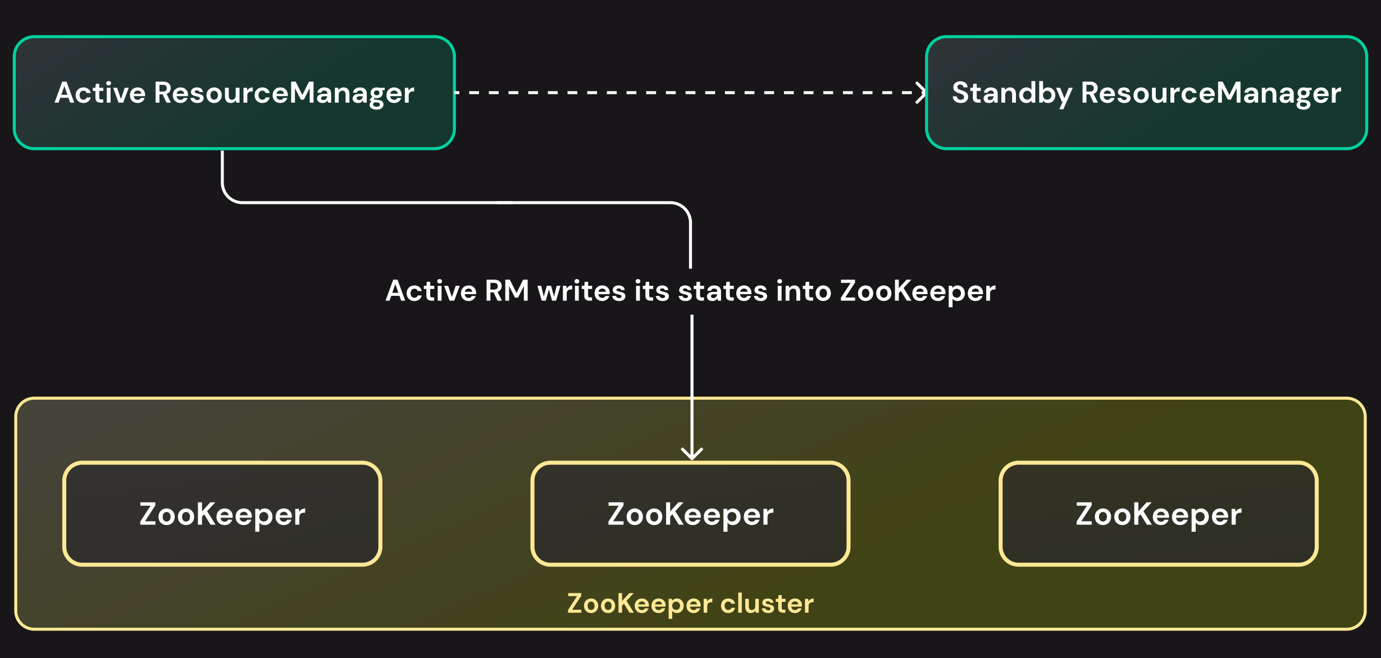

ARENADATA CLUSTER MANAGER
Clusters
Clusters / adh / Services / YARN / Info
11 Aug 2025 11:22:06 UTC
adh
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
YARN
3.3.6_arenadata1
5 / 5 successful components
Actions
Delete
Add/Remove components
Check
NodeManager Decommission/Recommission
Manage Ranger plugin
Restart
Remove
Add/modify node labels
Add/modify node-to-labels mapping
Reload Scheduler config
Primary configuration
Configuration groups
Action host groups
Components
Info
Framework for job scheduling and cluster resource management.
Web links
admin
Settings
Log out

Add ResourceManager

ResourceManager HA requires ZooKeeper and HDFS services to be running.

Architecture

The ResourceManager HA mode follows the active-standby approach, that is, at any given time one RM is active and one or more RMs are in the standby state, waiting to replace the active RM, if needed. Switching from standby to active occurs automatically through the integrated failover controller.


Active ResourceManager
Standby ResourceManager
Active RM writes its states into ZooKeeper
ZooKeeper
ZooKeeper
ZooKeeper
ZooKeeper cluster

Failover components

Automatic failover

By default, the system automatically switches from the failed to a standby RM in case of failures. During the failover, there is a need to ensure that no **split-brain** scenario occurs.

A split-brain scenario is a situation where multiple RMs assume they have the active role. This is undesirable because multiple RMs could proceed to manage some cluster nodes in parallel. This would cause inconsistency in resource management and job scheduling, potentially leading the entire system to a failure.

Thus, the goal is to ensure that at any given time there is one and only one active RM in the system. For this, ZooKeeper can be used to coordinate the failover process using the **Leader Election** algorithm described in the next section.

Leader Election algorithm

For better understanding of the election process, it might be useful to revise key ZooKeeper concepts.

A ZooKeeper client (a user, a service, or an application) can create Znodes of two types:

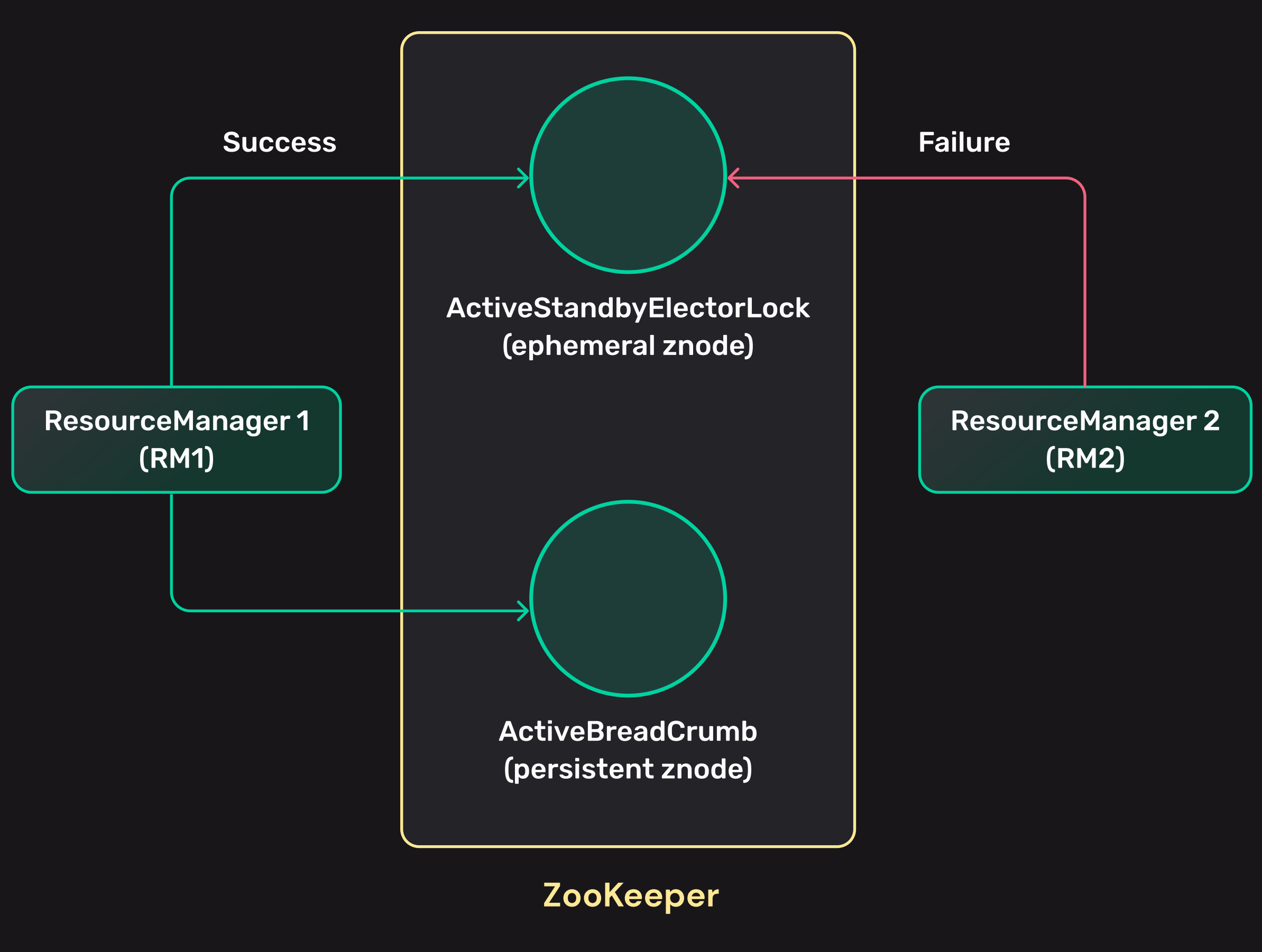
- **Ephemeral**. An ephemeral Znode gets deleted if the session in which the Znode was created has disconnected. For example, if RM1 has created an ephemeral Znode `eph_znode`, this Znode gets deleted as soon as the RM1's session with ZooKeeper is closed.
- **Persistent**. A persistent Znode exists within ZooKeeper until it is deleted. Unlike ephemeral Znodes, a persistent Znode is not affected by the existence of the process that has created it. For example, if RM1 creates a persistent Znode, then even if the RM1's session with ZooKeeper gets disconnected, the persistent Znode created by RM1 still exists and is not deleted upon RM1 going down. This Znode can be removed only when it is deleted explicitly.

For more details about ZooKeeper architecture and data model, see ZooKeeper architecture.

Election algorithm

Suppose there is a cluster with two RMs (RM1 and RM2) competing to become active. This race is resolved as follows: the RM that first succeeds in creating an ephemeral Znode named `ActiveStandbyElectorLock` gets the active role. It is important that only one RM is able to create the Znode named `ActiveStandbyElectorLock`.

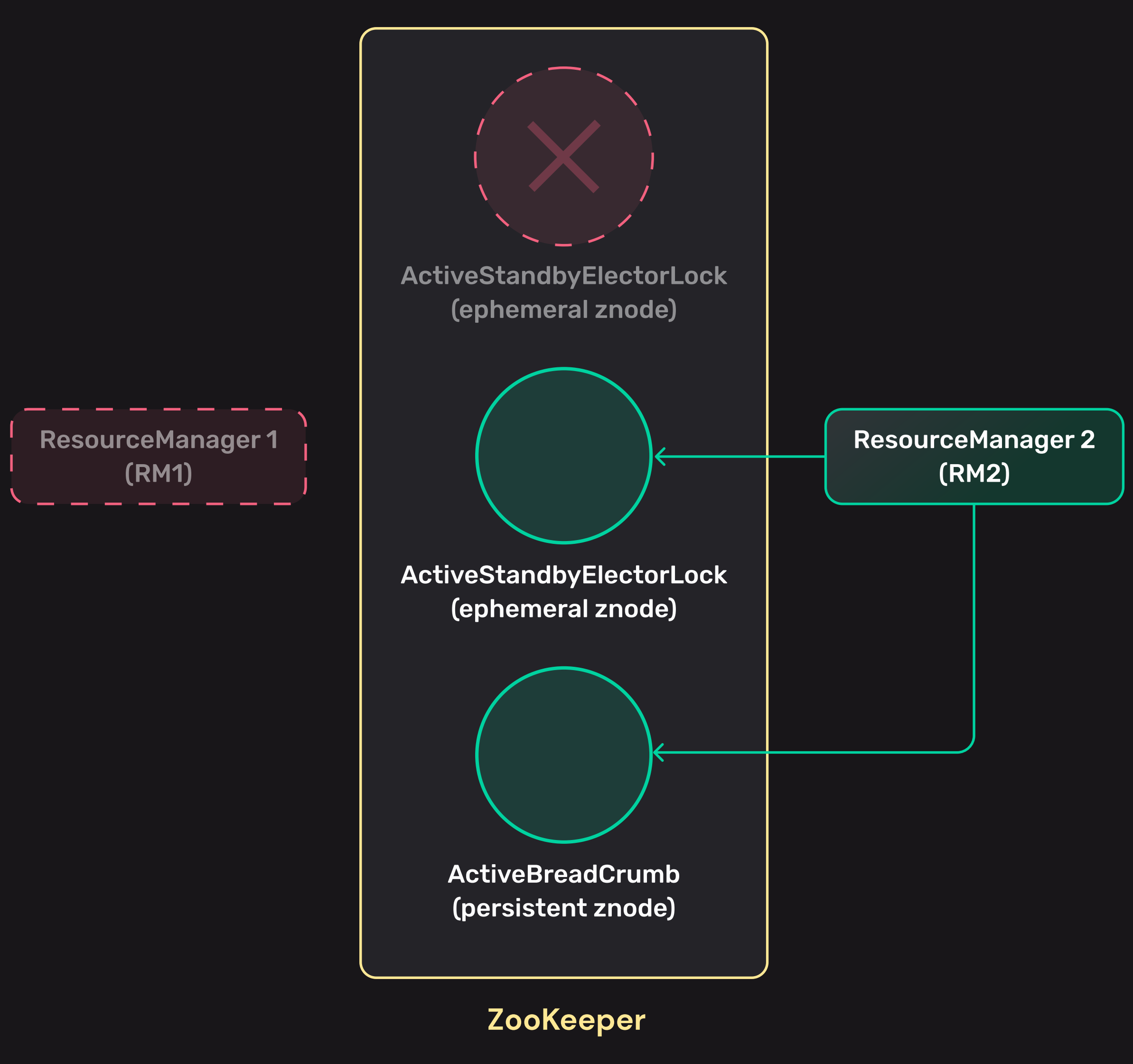
If RM1 was the first to create the `ActiveStandbyElectorLock` Znode, RM2 will no longer be able to create the same Znode. Thus, RM1 wins the election and becomes active, while RM2 gets the standby role. Before going to the standby mode, RM2 creates a watch for the `ActiveStandbyElectorLock` Znode to get notifications about the Znode's update/remove events.


Success
Failure
ActiveStandbyElectorLock
(ephemeral znode)
ResourceManager 1
(RM1)
ResourceManager 2
(RM2)
ActiveBreadCrumb
(persistent znode)
ZooKeeper

System state before failure

If RM1 goes down, the `ActiveStandbyElectorLock` Znode gets deleted automatically due to being ephemeral. Then, the standby RM2 becomes active.

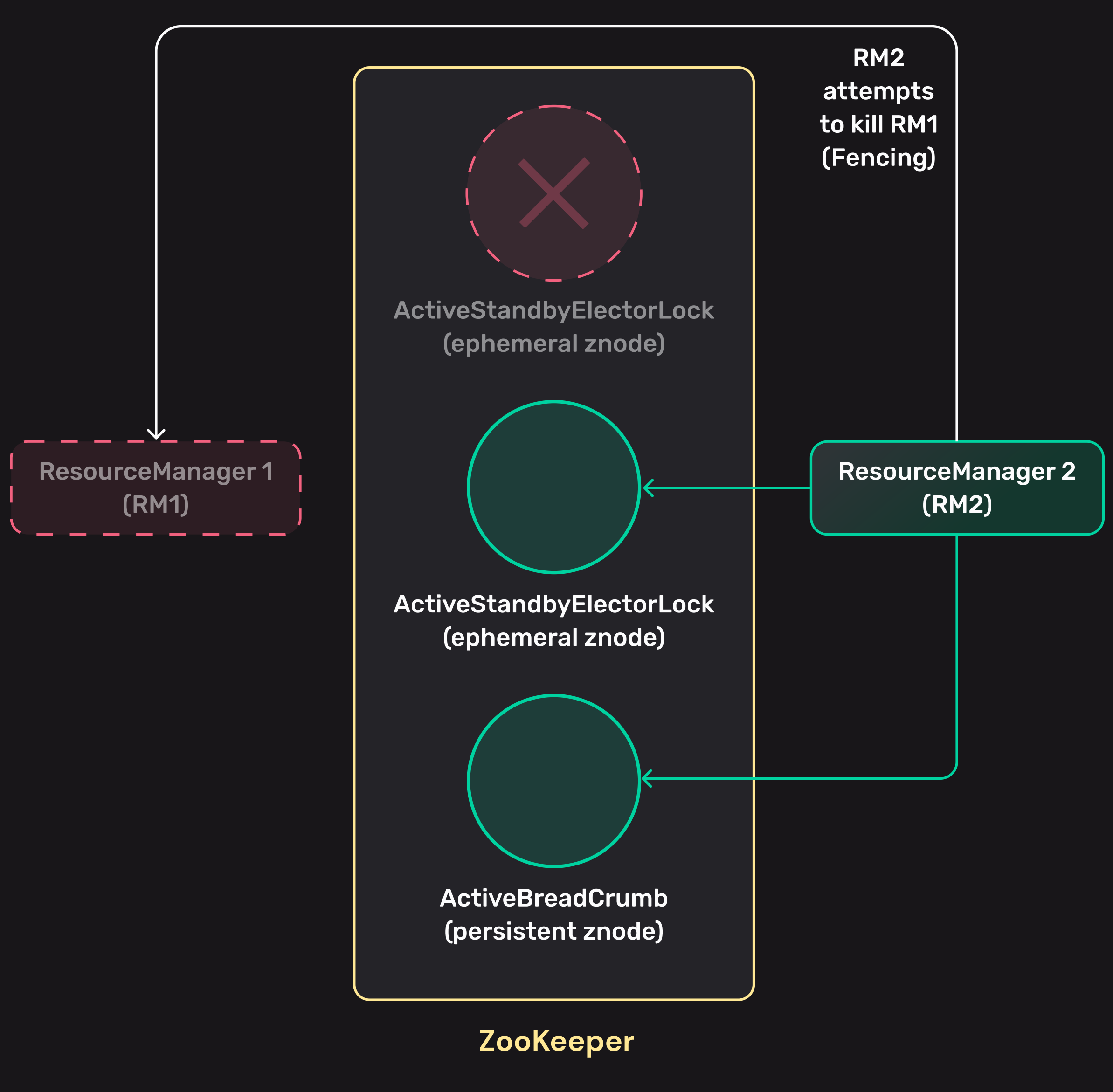
RM2 gets notified about the removal of the Znode that it has been watching, so it tries to create its own `ActiveStandbyElectorLock` ephemeral Znode, succeeds, and finally gets the active role.


ActiveStandbyElectorLock
(ephemeral znode)
ResourceManager 1
(RM1)
ResourceManager 2
(RM2)
ActiveStandbyElectorLock
(ephemeral znode)
ActiveBreadCrumb
(persistent znode)
ZooKeeper

System state after failover

Sometimes RM gets disconnected only temporarily and reconnects back to ZooKeeper. However, ZooKeeper cannot determine the cause of RM disconnect, and hence it deletes the ephemeral Znodes created by that RM. This can lead to a split-brain scenario, where both the RMs get the active role.

To avoid this, RM2 reads the information about the active RM from the persistent `ActiveBreadCrumb` Znode, where RM1 has already written its information (hostname and TCP port). Once RM2 knows that RM1 is alive, it tries to make RM1 fenced. On success, RM2 overwrites the data in the persistent `ActiveBreadCrumb` Znode with its own configuration (RM2's hostname and TCP port), since RM2 is now the active RM.


RM2
attempts
to kill RM1
(Fencing)
ActiveStandbyElectorLock
(ephemeral znode)
ResourceManager 1
(RM1)
ResourceManager 2
(RM2)
ActiveStandbyElectorLock
(ephemeral znode)
ActiveBreadCrumb
(persistent znode)
ZooKeeper

Fencing a competing resource manager

Client, ApplicationMaster, and NodeManager on RM failover

When there are multiple RMs, the configuration in the *yarn-site.xml* file used by clients and nodes must list all these RMs.

Clients, ApplicationMasters, and NodeManagers try connecting to the RMs in a round-robin order until they hit the active RM. If the active RM goes down, they resume the round-robin polling until they hit the "new" active RM. The failover policy for them is defined by one of the following properties:

- When HA is used, the retry logic is specified by the `yarn.client.failover-proxy-provider` property that defaults to `org.apache.hadoop.yarn.client.ConfiguredRMFailoverProxyProvider`. You can override the logic by providing a custom class that implements `org.apache.hadoop.yarn.client.RMFailoverProxyProvider` and setting the property value to this class name.
- When running in the non-HA mode, the retry logic is specified by the `yarn.client.failover-no-ha-proxy-provider` property.

Examples of configuring YARN HA

Mikhail Serov

Contents

To Table of Contents

Configuration properties
Sample configuration
Administrative commands

Configuration properties

You can tune most of the failover functionality using various configuration properties. Following is a list of the required and important ones.

The *yarn-default.xml* file carries a full list of knobs. See this file for more information including default values. See ResourceManager Restart for instructions on setting up the state-store.

Configuration parameter	Description
hadoop.zk.address	A list of host:port pairs of the ZooKeeper servers. Used for both the HDFS state-store and embedded leader-election
yarn.resourcemanager.ha.enabled	Enable RM HA
yarn.resourcemanager.ha.rm-ids	A list of logical IDs for the RMs. e.g., `rm1,rm2`
yarn.resourcemanager.hostname.rm-id	For each `rm-id`, specifies the name of the host where the RM is installed. Alternately, you can set each of the RM's service addresses
yarn.resourcemanager.address.rm-id	For each `rm-id`, specifies the resource manager host:port for clients to submit jobs. If set, overrides the hostname set in `yarn.resourcemanager.hostname.rm-id`
yarn.resourcemanager.scheduler.address.rm-id	For each `rm-id`, specifies the scheduler host:port for ApplicationMasters to obtain resources. If set, overrides the hostname set in `yarn.resourcemanager.hostname.rm-id`
yarn.resourcemanager.resource-tracker.address.rm-id	For each `rm-id`, specifies the host:port for NodeManagers to connect. If set, overrides the hostname set in `yarn.resourcemanager.hostname.rm-id`
yarn.resourcemanager.admin.address.rm-id	For each `rm-id`, specifies the host:port for administrative commands. If set, overrides the hostname set in `yarn.resourcemanager.hostname.rm-id`
yarn.resourcemanager.webapp.address.rm-id	For each `rm-id`, specifies the host:port of the RM web application. You do not need this if you set `yarn.http.policy` to HTTPS_ONLY. If set, overrides the hostname set in `yarn.resourcemanager.hostname.rm-id`
yarn.resourcemanager.webapp.https.address.rm-id	For each `rm-id`, specifies the host:port of the RM HTTPS web application. You do not need this if you set `yarn.http.policy` to HTTP_ONLY. If set, overrides the hostname set in `yarn.resourcemanager.hostname.rm-id`
yarn.resourcemanager.ha.id	Identifies the RM in the ensemble. This is optional; however, if set, admins have to ensure that all the RMs have their own IDs in the config
yarn.resourcemanager.ha.automatic-failover.enabled	Enable automatic failover. By default, it is enabled, but the service is active only when HA is enabled
yarn.resourcemanager.ha.automatic-failover.embedded	Use embedded leader-elector to pick the active RM when automatic failover is enabled. By default, it is enabled, but the service is active only when HA is enabled
yarn.resourcemanager.cluster-id	Identifies the cluster. Used by the elector to ensure an RM doesn't take over the active role in another cluster
yarn.client.failover-proxy-provider	The class to be used by clients, AMs, and NMs to failover to the active RM
yarn.client.failover-no-ha-proxy-provider	The class to be used by clients, AMs and NMs to failover to the active RM, when not running in HA mode
yarn.client.failover-max-attempts	The maximum number of times FailoverProxyProvider should attempt failover
yarn.client.failover-sleep-base-ms	The sleep base (in milliseconds) to be used for calculating the exponential delay between failover processes
yarn.client.failover-sleep-max-ms	The maximum sleep time (in milliseconds) between failover processes
yarn.client.failover-retries	The number of retries per attempt to connect to an RM
yarn.client.failover-retries-on-socket-timeouts	The number of retries per attempt to connect to an RM on socket timeouts

Sample configuration

Here is a sample of minimal setup for RM failover:

```
<property>
  <name>yarn.resourcemanager.ha.enabled</name>
  <value>true</value>
</property>
<property>
  <name>yarn.resourcemanager.cluster-id</name>
  <value>cluster1</value>
</property>
<property>
  <name>yarn.resourcemanager.ha.rm-ids</name>
  <value>rm1,rm2</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname.rm1</name>
  <value>master1</value>
</property>
<property>
  <name>yarn.resourcemanager.hostname.rm2</name>
  <value>master2</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm1</name>
  <value>master1:8088</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm2</name>
  <value>master2:8088</value>
</property>
<property>
  <name>hadoop.zk.address</name>
  <value>zk1:2181,zk2:2181,zk3:2181</value>
</property>
```

Administrative commands

The `yarn rmadmin` utility has a few HA-specific command options to check the health and state of an RM and its transition to the active or standby mode. When HA is enabled, the commands take the service ID of an RM set by `yarn.resourcemanager.ha.rm-ids` as an argument.

Requesting the RM1 state:

```
$ yarn rmadmin -getServiceState rm1
```

The output:

```
active
```

Similar request for the RM2 state:

```
$ yarn rmadmin -getServiceState rm2
standby
```

If automatic failover is enabled, you cannot use manual transition command.

> **CAUTION**
>
> Although you can override this by the `-forcemanual` flag, use it with caution.

A transition attempt:

```
$ yarn rmadmin -transitionToStandby rm1
```

The request is denied:

```
Automatic failover is enabled for org.apache.hadoop.yarn.client.RMHAServiceTarget@1d8299fd
Refusing to manually manage HA state, since it may cause
a split-brain scenario or other incorrect state.
If you are very sure you know what you are doing, please
specify the forcemanual flag.
```

See additionally YARN CLI for frequently used YARN commands.

Logging in YARN

Elena Kostyuchenko

Contents

To Table of Contents

Overview
Grep logs
Logging levels
Container logs
Log aggregation

Overview

YARN records command output and processes events in text logs. This information can be useful when diagnosing technical issues.

Each YARN component stores logs locally, on the host it occupies. The logs have the *.log* extension and are located in the *var/log/hadoop-yarn/* directory. In the same directory, you can find *.out* files that contain component restart information.

All logs of all YARN components are also available in YARN UIs.

The logs naming convention is `hadoop-yarn-<component>-<host>.log`.

Where:

- `<component>` — a component name, for example, ResourceManager;
- `<host>` — the FQDN of the component host.

The YARN logs configuration shares the same log4j properties file with HDFS.

Grep logs

You can search through the logs for a specific information, like error messages. To do this, connect to the host with the component whose logs you want to inspect, and use a `grep` command.

For example:

```
$ cat /var/log/hadoop-yarn/hadoop-yarn-nodemanager-elenas-adh3.ru-central1.internal.log | grep -i -A3 -B1 error
```

This command searches for messages containing the word `error` in the CLI log, located on the `elenas-adh3.ru-central1.internal` host. The `-i` option allows you to ignore case distinctions. The `-A3 -B1` options expand the output to one line before and three lines after the line containing the error.

The example output:

```
2024-04-08 13:26:24,910 INFO org.apache.hadoop.yarn.server.nodemanager.containermanager.ContainerManagerImpl: ContainerManager
bound to 0.0.0.0/0.0.0.0:8041
2024-04-08 13:26:24,912 ERROR org.apache.hadoop.yarn.server.nodemanager.containermanager.scheduler.ContainerScheduler: Unknown
event arrived at ContainerScheduler: EventType: RECOVERY_COMPLETED
2024-04-08 13:26:24,915 INFO org.apache.hadoop.yarn.server.nodemanager.webapp.WebServer: Instantiating NMWebApp at 0.0.0.0:8042
2024-04-08 13:26:24,935 INFO org.eclipse.jetty.util.log: Logging initialized @56117ms
2024-04-08 13:26:25,004 INFO org.apache.hadoop.security.authentication.server.AuthenticationFilter: Unable to initialize
FileSignerSecretProvider, falling back to use random secrets.
```

Logging levels

YARN uses the Log4j2 library for logging which supports the following log levels (from least to most informative):

1. `FATAL` — indicates that an operation can't continue execution and will terminate.
2. `ERROR` — notifies that a program is not working correctly or has stopped.
3. `WARN` — warns about potential problems. This doesn't mean that a program is not working, but raises a concern.
4. `INFO` — informs regarding the program lifecycle or state.
5. `DEBUG` — prints debugging information about internal states of the program.
6. `TRACE` — prints messages tracing the execution flow of a program.

The Log4j4 loggers also accept logging level values: `OFF` — for switching off the logging, and `ALL` — for allowing all types of messages.

Enabling one level of logging will enable this level and all levels above it. For example, if you set the logging level to `WARN`, then only warnings, errors, and fatal messages would get into the log files, but not `INFO`, `DEBUG`, and `TRACE`.

Container logs

Applications submitted to YARN generate their own logs. To read application logs, use the logs command.

Log aggregation

Log aggregation is a YARN feature that, when enabled, copies application log files into a single file in HDFS after an application has finished.

The default directory for aggregated application logs is *hdfs://<NameNode address>:8020/logs/yarn/logs/*, where `<NameNode address>` is the IP address or FQDN of the active NameNode.

You can configure log aggregation by changing its properties in the *yarn-site.xml* configuration file.

YARN log aggregation parameters

Parameter	Description	Default value
yarn.log.server.url	A URL for displaying the application logs. If set to the JobHistory server, the logs will be available in the Resource Manager UI	—
yarn.log-aggregation-enable	Enables log aggregation	true
yarn.log-aggregation.retain-seconds	The amount of time (in seconds) before the logs will be deleted. Setting this to `-1` or lower will disable the deletion of logs. Entering a value too small will overload the NameNode	172800
yarn.nodemanager.remote-app-log-dir	The application logs directory in HDFS	/logs

To change logging properties via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at YARN.
3. Find the ***yarn-site.xml*** field.
4. Select the required parameter and make the necessary changes.
5. Confirm changes by clicking ***Save***.
6. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

Configure and use GPU on YARN

Elena Kostyuchenko

Contents

To Table of Contents

Overview

GPU configuration

Use GPU on YARN

Overview

For computationally intensive tasks, Hadoop can be configured to run YARN applications on GPU instead of CPU.

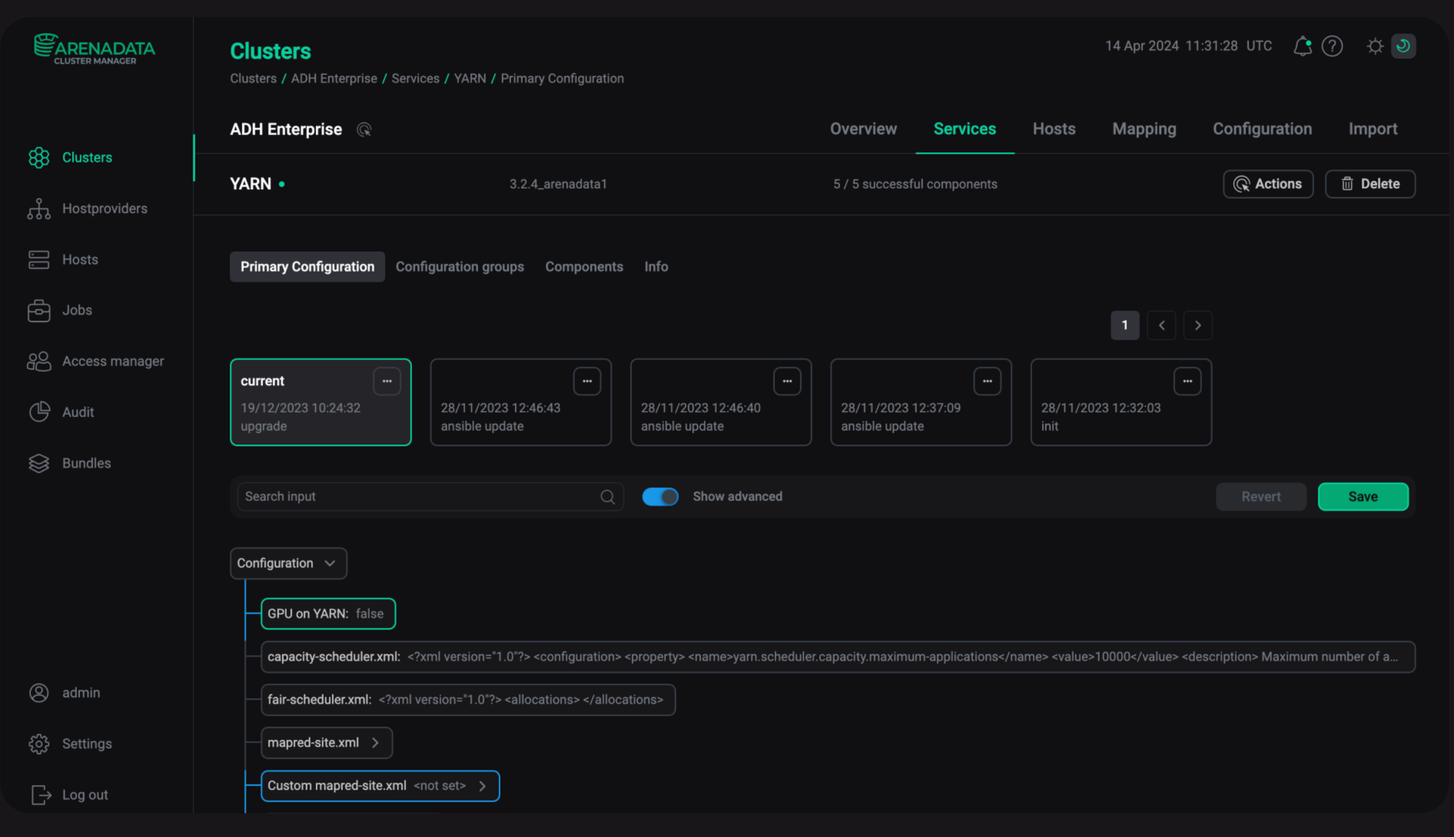
Prerequisites for using GPU on YARN:

- An Nvidia GPU.
- Nvidia drivers.
- If Docker is used, nvidia-docker 1.0 has to be installed.
- YARN uses Capacity scheduler.

GPU configuration

To configure the GPU support on YARN via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Services*** and click at YARN.
3. Toggle the ***Show advanced*** option, find the ***GPU on YARN*** setting and set it to `true`.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH Enterprise / Services / YARN / Primary Configuration
14 Apr 2024 11:31:28 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADH Enterprise
Overview
Services
Hosts
Mapping
Configuration
Import
YARN
3.2.4_arenadata1
5 / 5 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
current 19/12/2023 10:24:32 upgrade
28/11/2023 12:46:43 ansible update
28/11/2023 12:46:40 ansible update
28/11/2023 12:37:09 ansible update
28/11/2023 12:32:03 init
Search input
Show advanced
Revert
Save
Configuration

GPU on YARN: false

capacity-scheduler.xml: <?xml version="1.0"?> <configuration> <property> <name>yarn.scheduler.capacity.maximum-applications</name> <value>10000</value> <description> Maximum number of a...

fair-scheduler.xml: <?xml version="1.0"?> <allocations> </allocations>

mapred-site.xml

Custom mapred-site.xml <not set>

GPU on YARN setting in ADCM

4. Find and click at the ***capacity-scheduler.xml*** property.
5. In the opened window, change the `yarn.scheduler.capacity.resource-calculator` property value to `org.apache.hadoop.yarn.util.resource.DominantResourceCalculator`:

```
<configuration>
  <property>
    <name>yarn.scheduler.capacity.resource-calculator</name>
    <value>org.apache.hadoop.yarn.util.resource.DominantResourceCalculator</value>
  </property>
</configuration>
```

6. In the ***yarn-site.xml*** parameter group, edit the properties to create the desired configuration for your cluster that would account for the GPU usage. For example:

```
<configuration>
  <property>
    <name>yarn.nodemanager.resource-plugins</name>
    <value>yarn.io/gpu</value>
  </property>
  <property>
     <name>yarn.resource-types</name>
     <value>yarn.io/gpu</value>
  </property>
  <property>
     <name>yarn.nodemanager.resource-plugins.gpu.allowed-gpu-devices</name>
     <value>auto</value>
  </property>
  <property>
     <name>yarn.nodemanager.resource-plugins.gpu.path-to-discovery-executables</name>
     <value>/usr/bin</value>
  </property>
  <property>
     <name>yarn.nodemanager.linux-container-executor.cgroups.mount</name>
     <value>true</value>
  </property>
  <property>
     <name>yarn.nodemanager.linux-container-executor.cgroups.mount-path</name>
     <value>/sys/fs/cgroup</value>
  </property>
  <property>
     <name>yarn.nodemanager.linux-container-executor.cgroups.hierarchy</name>
     <value>yarn</value>
  </property>
  <property>
    <name>yarn.nodemanager.container-executor.class</name>
    <value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</value>
  </property>
  <property>
    <name>yarn.nodemanager.linux-container-executor.group</name>
    <value>yarn</value>
  </property>
</configuration>
```

7. Find the ***container-executor.cfg template*** parameter and add the following values to the template:

```
yarn.nodemanager.linux-container-executor.group=yarn

#--Original container-exectuor.cfg Content--

[gpu]
module.enabled=true
[cgroups]
root=/sys/fs/cgroup
yarn-hierarchy=yarn
```

If you need to run GPU applications under Docker environment, add these additional properties to ***container-executor.cfg template***:

```
[docker]
docker.allowed.devices=/dev/nvidiactl,/dev/nvidia-uvm,/dev/nvidia-uvm-tools,/dev/nvidia1,/dev/nvidia0 ①
docker.allowed.volume-drivers ②
docker.allowed.ro-mounts=nvidia_driver_375.66 ③
docker.allowed.runtimes=nvidia ④
```

① Adds GPU devices to the Docker section. Enter the list of values separated by comma. You can find the list of devices using the `ls /dev/nvidia*` command.

② Adds `nvidia-docker` to volume-driver whitelist.

③ Adds `nvidia_driver_<version>` to read-only mounts whitelist.

④ To use *nvidia-docker-v2* as the GPU Docker plugin, add `nvidia` to runtimes whitelist.

8. Click ***Apply*** and confirm changes to YARN configuration by clicking ***Save***.
9. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true` and click ***Run***. This will automatically update all the necessary configuration files on all hosts.

> **NOTE**
>
> In some cases, it might be required to share the NodeManager's local directory to all users to avoid permission-related errors.

Depending on your cluster configuration, some of the parameters can be optional for using GPUs and can be set to default values.

GPU on YARN: optional parameters

Parameter	Description	Default value
yarn.nodemanager.resource-plugins	Set to `yarn.io/gpu` to enable GPU isolation module on the NodeManager side if necessary. By default, YARN automatically detects and configures GPUs	—
yarn.nodemanager.resource-plugins.gpu.allowed-gpu-devices	A list of comma-separated GPU devices in the following format: `GPU1_index:GPU1_minor_number, GPU2_index:GPU2_minor_number`. For example: `0:0,1:1,2:2`. If the default value is set, YARN will automatically discover GPU resources in the system. If it fails, you can use this property to manually specify the GPUs	auto
yarn.nodemanager.resource-plugins.gpu.path-to-discovery-executables	The path to the GPU discovery binary used by NodeManagers for gathering the information about GPUs if `yarn.nodemanager.resource.gpu.allowed-gpu-devices` is set to `auto`. For example: `/usr/local/bin/nvidia-smi`. If this value is empty, a NodeManager will try to locate the binary on its own	—
yarn.nodemanager.resource-plugins.gpu.docker-plugin	The Docker command plugin for GPU. By default, the Nvidia Docker v1.0 is used	nvidia-docker-v1
yarn.nodemanager.resource-plugins.gpu.docker-plugin.nvidia-docker-v1.endpoint	The Nvidia Docker plugin endpoint	http://localhost:3476/v1.0 /docker/cli
yarn.nodemanager.linux-container-executor.cgroups.mount	Use the CGroup devices controller to do per-GPU device isolation	true

Use GPU on YARN

Here are the examples of running YARN applications using the yarn jar command in a YARN or Docker container.

YARN container

To run an application in a YARN container, run:

```
$ yarn jar <path/to/hadoop-yarn-applications-distributedshell.jar> \
  -jar <path/to/hadoop-yarn-applications-distributedshell.jar> \
  -shell_command /usr/local/nvidia/bin/nvidia-smi \
  -container_resources memory-mb=3072,vcores=1,yarn.io/gpu=2 \
  -num_containers 2
```

This command requests 2 containers, 3GB each, 1 vCore, and 2 GPUs.

Docker container

To run an application in a Docker container, run:

```
$ yarn jar <path/to/hadoop-yarn-applications-distributedshell.jar> \
       -jar <path/to/hadoop-yarn-applications-distributedshell.jar> \
       -shell_env YARN_CONTAINER_RUNTIME_TYPE=docker \
       -shell_env YARN_CONTAINER_RUNTIME_DOCKER_IMAGE=<docker-image-name> \
       -shell_command nvidia-smi \
       -container_resources memory-mb=3072,vcores=1,yarn.io/gpu=2 \
       -num_containers 2
```

This command requests 2 containers, 3GB each, 1 vCore, and 2 GPUs.

YARN service management via ADCM

Elena Kostyuchenko

To Table of Contents

The ADCM UI provides actions to manage the YARN service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the YARN service are listed in the table below.

For more information about YARN components, see YARN architecture.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute YARN components
NodeManager Decommission/Recommission	Allows you to decommission/recommission NodeManager components. The decommission action can be used to temporarily take a NodeManager out of service or to safely delete it. When running this action, specify the hosts with NodeManagers to decommission in the ***Hosts in decommissioned mode*** list. For each item in this list, specify the FQDN of the host (***Enter field name***) and the timeout in seconds to start the decommission process for that host (***Enter field value***). By default, the decommission starts immediately. After all hosts are specified, review the configuration and run the action. To decommission a NodeManager gracefully, use the `refreshNodes` command with the `-g` option
Manage Ranger plugin	Enables or disables Ranger plugin for YARN
Add/modify node labels	Adds or changes node labels. You can use node labels to create groups of hosts, for example, to designate nodes to specific tasks or to manage nodes' resources between queues. When using this action, enter labels to be used in a cluster and their exclusivity (`True` or `False`). The dafault exclusivity is `True`
Add/modify node-to-labels mapping	Maps the node labels added using the ***Add/modify node labels*** action to the hosts. When using this action, enter the FQDNs of the hosts with NodeManager components and the labels you want to assign to them
Reload Scheduler config	Reloads access control lists, states, and scheduler specific properties. Use this action to apply changes made to Capacity Scheduler's configuration file. Fair Scheduler doesn't require reloading after its configuration has been altered
Remove node label	Removes node labels from the cluster. When using this action, enter the list of labels you want to remove separated by commas
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM. The service supports the ***Rolling restart*** option that allows you to restart its NodeManager components one by one (or in batches) rather than all at once. With high availability enabled, this feature allows for a seamless restart of the YARN service, making it possible to preserve running YARN containers. The ***Rolling restart*** option has the following parameters: ▪ ***batch_size*** — the number of NodeManager components to be restarted in one iteration. ▪ ***batch_delay*** — the delay in seconds between restarting batches of components. ▪ ***health_checks*** — indicates whether to perform health checks on the restarted components. ▪ ***max_failed_batches_number*** — the maximum number of NodeManager component batches allowed to fail during restart. Upon reaching this value, the ***Restart*** action fails.

YARN components actions

Action	Description
Check	Verifies whether all the component instances in the cluster work correctly
Restart	Restarts all the component instances in the cluster
Start	Starts all the component instances in the cluster
Stop	Stops all the component instances in the cluster

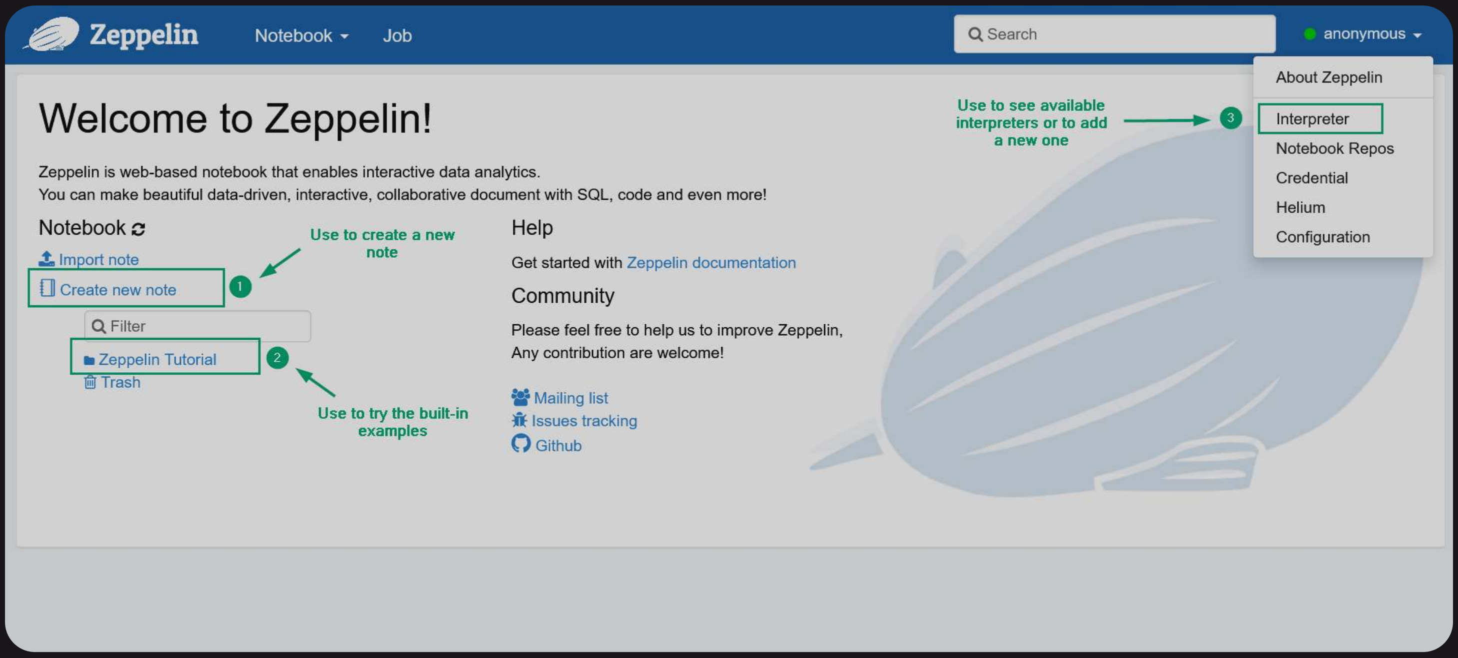
Zeppelin UI overview

Daria Barysheva

To Table of Contents

Zeppelin provides a web-based user interface that allows to work with interpreters and notes. The interface is available at the following URL, where *<IP>* indicates the IP address of your Zeppelin server: *http://<IP>:8180/*.

The home page of Zeppelin web UI is shown below:


Zeppelin
Notebook
Job
Search
anonymous
About Zeppelin
Interpreter
Notebook Repos
Credential
Helium
Configuration
Notebook
Import note
Create new note
Filter
Zeppelin Tutorial
Trash
Help
Community
Mailing list
Issues tracking
Github
1
2
3

Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Get started with Zeppelin documentation

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Use to create a new note

Use to try the built-in examples

Use to see available interpreters or to add a new one

Zeppelin home page

To start working with Zeppelin, use one of these ways:

1. **Create a new note**. To create a new note, click the ***Create new note*** list item at the left side of the home page. This function is also available after expanding the ***Notebook*** main menu item. The examples of creating notes, based on different interpreters, are available in the Examples of creating notes in Zeppelin section.

2. **View the available tutorials**. Clicking the ***Zeppelin Tutorial*** item at the left side of the home page opens the list of built-in examples. You can go through each of them and try to execute the listed code by yourself.

3. **View the existing interpreters or add a new one**. To start working with interpreters, you should select the ***Interpreter*** item in the account menu located in the top right corner of the home page. The example of adding a custom interpreter is described in the Add a custom interpreter to a group section.

NOTE

The detailed UI description is available in the Zeppelin documentation. It is highly recommended that you read it before starting the work.

Add a custom interpreter to a group

Daria Barysheva

Contents

To Table of Contents

Preparation

Method 1. Using prefixes

Method 2. Adding an interpreter from scratch

The list of built-in Zeppelin interpreters is available in the web UI after selecting the ***Interpreter*** item in the account menu. You can also add your own interpreters within the existing groups. This section shows two methods of creating a new interpreter, using the example of adding Phoenix interpreter to the `jdbc` group.

> **NOTE**
>
> You can find more information about interpreters in the Zeppelin documentation. Zeppelin also allows to create your own interpreters, which are not based on the existing groups, and to install third-party interpreters.

Preparation

As described in the Connect to Phoenix section, you can access Phoenix databases via JDBC. In the examples throughout this section, we use the Phoenix Thin Driver for this purpose. Before using any of two methods, you should perform preparation steps:

1. Get the corresponding version of the JAR file *phoenix-<phoenix_version>-HBase-<hbase_version>-thin-client.jar*. For example, for HBase 2.0 and Phoenix 5.0.0, this file is called *phoenix-5.0.0-HBase-2.0-thin-client.jar*. You can copy it from any host of your HBase cluster (from the */usr/lib/phoenix* directory). The other way is to download it from Phoenix repository.
2. Copy the prepared JAR file from the source folder to the interpreter directory */usr/lib/zeppelin/interpreter/jdbc/*:

```
$ sudo cp /usr/lib/phoenix/phoenix-5.0.0-HBase-2.0-thin-client.jar /usr/lib/zeppelin/interpreter/jdbc/
```

3. Verify that the file is copied successfully using the command:

```
$ ls -la /usr/lib/zeppelin/interpreter/jdbc/
```

The output looks as follows:

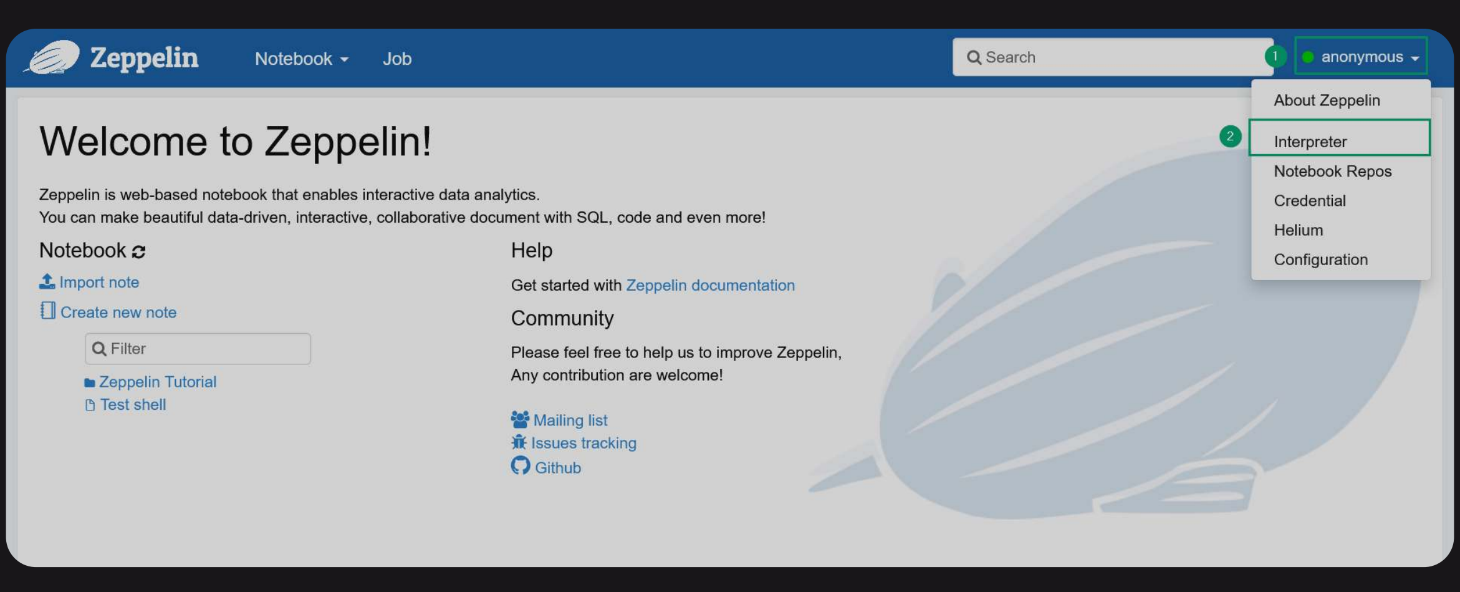
```
total 35412
drwxr-xr-x.  3 root root     4096 Dec  7 17:46 .
drwxr-xr-x. 28 root root     4096 Dec  7 08:36 ..
-rw-r--r--.  1 root root     2314 Apr 15  2021 ansi.sql.keywords
-rw-r--r--.  1 root root   167962 Apr 15  2021 commons-dbcp2-2.0.1.jar
-rw-r--r--.  1 root root    60686 Apr 15  2021 commons-logging-1.1.1.jar
-rw-r--r--.  1 root root   109568 Apr 15  2021 commons-pool2-2.3.jar
-rw-r--r--.  1 root root     4800 Apr 15  2021 interpreter-setting.json
-rw-r--r--.  1 root root   489884 Apr 15  2021 log4j-1.2.17.jar
drwxr-xr-x.  2 root root       55 Dec  6 09:10 META-INF
-rw-r--r--.  1 root root 34651233 Dec  7 17:46 phoenix-5.0.0-HBase-2.0-thin-client.jar
-rw-r--r--.  1 root root   648487 Apr 15  2021 postgresql-9.4-1201-jdbc41.jar
-rw-r--r--.  1 root root     6573 Apr 15  2021 postgresql-native-driver-sql.keywords
-rw-r--r--.  1 root root    32119 Apr 15  2021 slf4j-api-1.7.10.jar
-rw-r--r--.  1 root root     8866 Apr 15  2021 slf4j-log4j12-1.7.10.jar
-rw-r--r--.  1 root root    46253 Apr 15  2021 zeppelin-jdbc-0.8.1.jar
```

> **TIP**
>
> Starting with Zeppelin 0.9.0, you can skip the preparation steps and use the Apache Phoenix guide to create a new Phoenix interpreter exclusively via the web UI. In older versions, errors may occur while accessing the Maven repository for resolving dependencies. That is why we recommend placing the necessary jars manually.

Method 1. Using prefixes

This method is based on adding new properties with a specific prefix to the existing interpreter. Thus, when working with notes, you can use the source interpreter not only with the default settings, but with the settings that have the special prefix. To call such an interpreter from the note, you should append the prefix to the main interpreter name, for example: `%jdbc(phoenix)`.

1. Select the ***Interpreter*** item in the account menu placed in the top right corner of the Zeppelin home page.



Zeppelin
Notebook
Job
Search
1
anonymous
About Zeppelin
2
Interpreter
Notebook Repos
Credential
Helium
Configuration

Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Notebook
Import note
Create new note
Filter
Zeppelin Tutorial
Test shell
Help

Get started with Zeppelin documentation

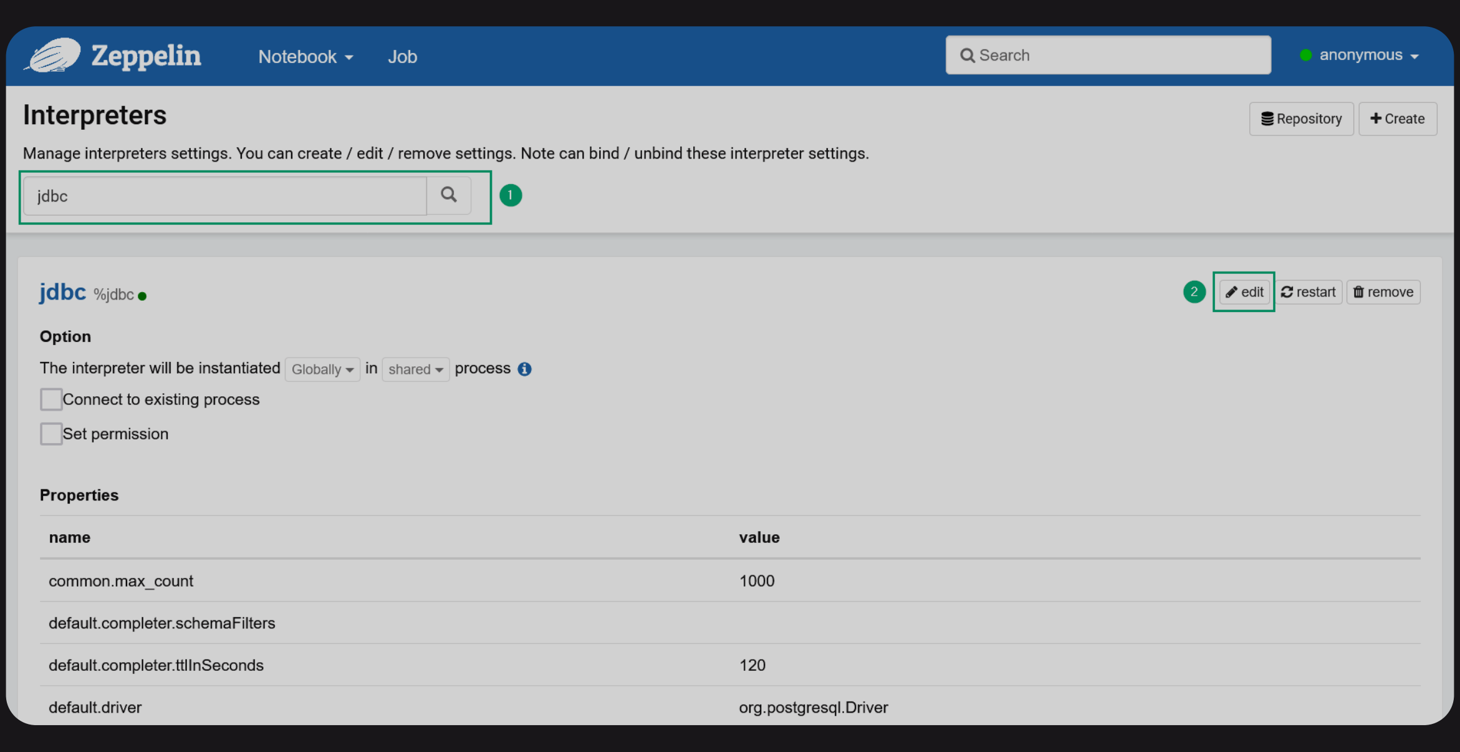
Community

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Mailing list
Issues tracking
Github

Switching to the list of interpreters

2. Using the search bar, find the *jdbc* interpreter, and click ***edit***.



Zeppelin
Notebook
Job
Search
anonymous
Interpreters
Repository
Create
1
jdbc %jdbc
2
edit
restart
remove
Option

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

jdbc

The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

☐ Set permission

Properties

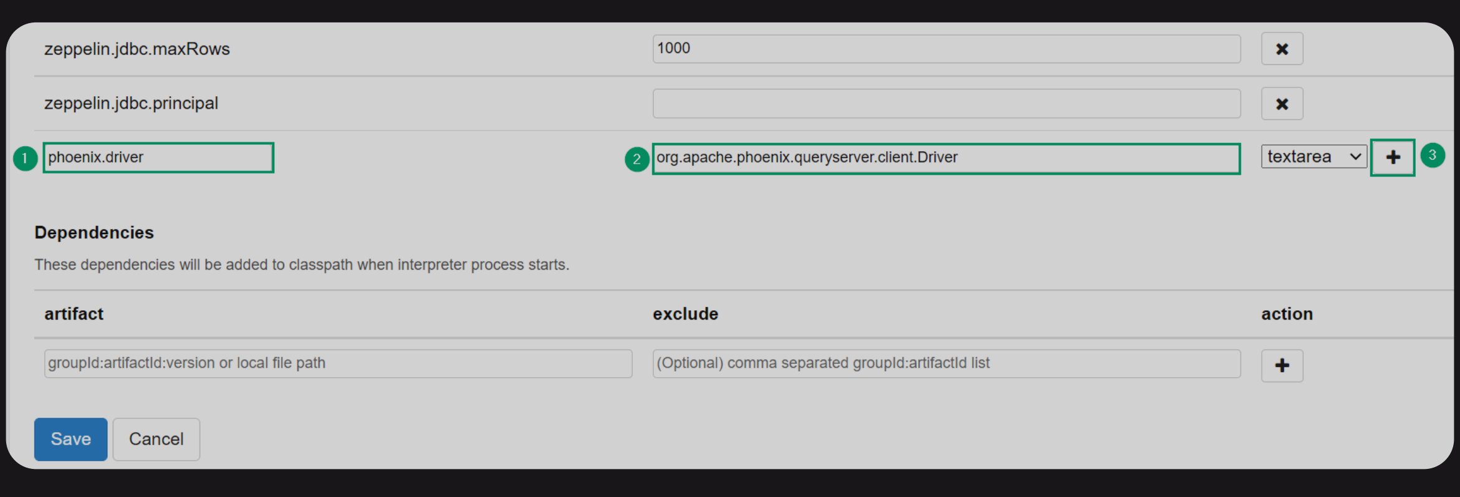
name	value
common.max_count	1000
default.completer.schemaFilters	
default.completer.ttlInSeconds	120
default.driver	org.postgresql.Driver

Editing the JDBC interpreter

3. Add new properties to the interpreter settings list with the `phoenix` prefix.

Name	Value	Description
phoenix.driver	org.apache.phoenix.queryserver.client.Driver	The driver used to connect to Phoenix
phoenix.url	jdbc:phoenix:thin:url=http://<IP>:8765;serialization=PROTOBUF	The url used for connecting to Phoenix. `<IP>` means the IP-address of your Phoenix Query Server
phoenix.user	—	The login used for connecting to Phoenix
phoenix.password	—	The password used for connecting to Phoenix

To add a property, you should enter its name, value, and then click **+** to add the next field.



1
2
textarea
+
3
Dependencies
artifact
exclude
action
Save
Cancel

zeppelin.jdbc.maxRows 1000

zeppelin.jdbc.principal

phoenix.driver org.apache.phoenix.queryserver.client.Driver

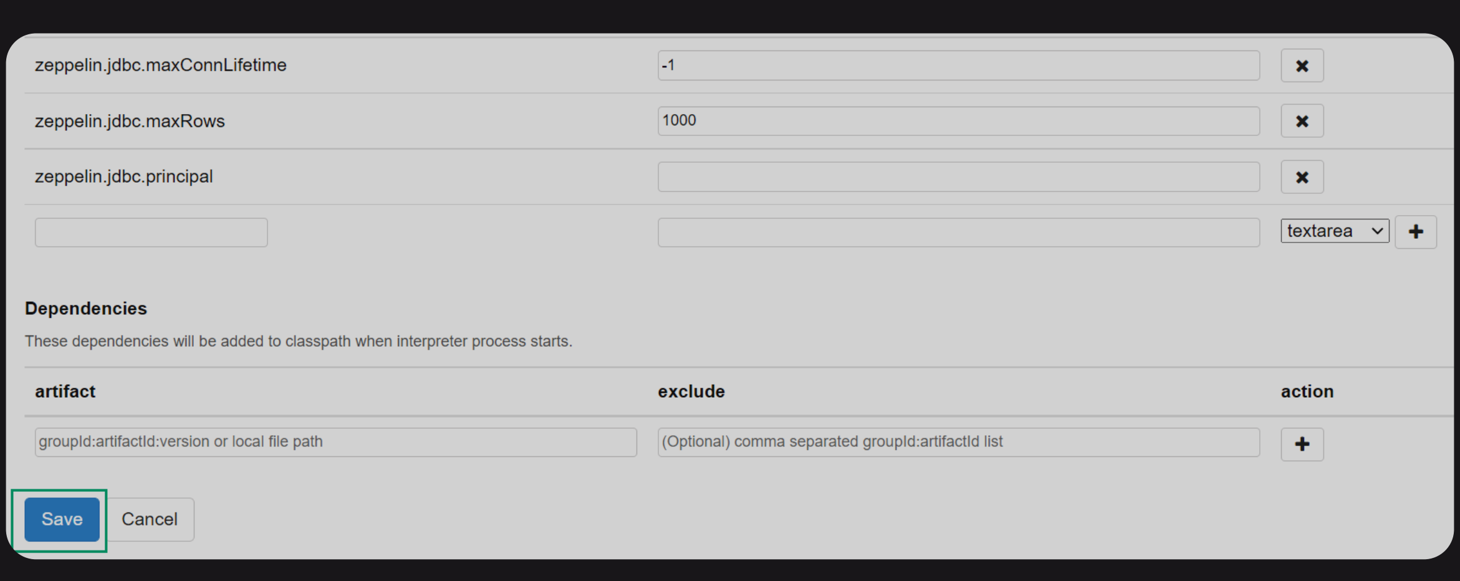
These dependencies will be added to classpath when interpreter process starts.

groupId:artifactId:version or local file path

(Optional) comma separated groupId:artifactId list

Adding new fields

4. After adding all the necessary fields, click ***Save***.



textarea
+
Dependencies
artifact
exclude
action
Save
Cancel

zeppelin.jdbc.maxConnLifetime -1

zeppelin.jdbc.maxRows 1000

zeppelin.jdbc.principal

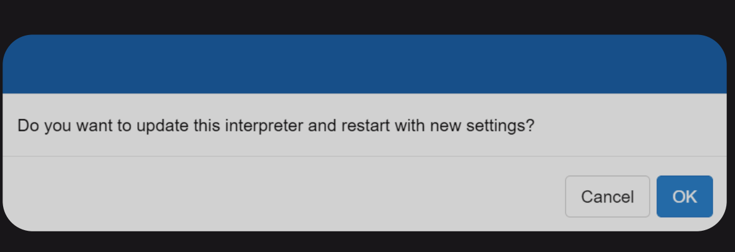
These dependencies will be added to classpath when interpreter process starts.

groupId:artifactId:version or local file path

(Optional) comma separated groupId:artifactId list

Saving the JDBC interpreter

5. Confirm the operation in the opened dialog window. Your interpreter is ready.



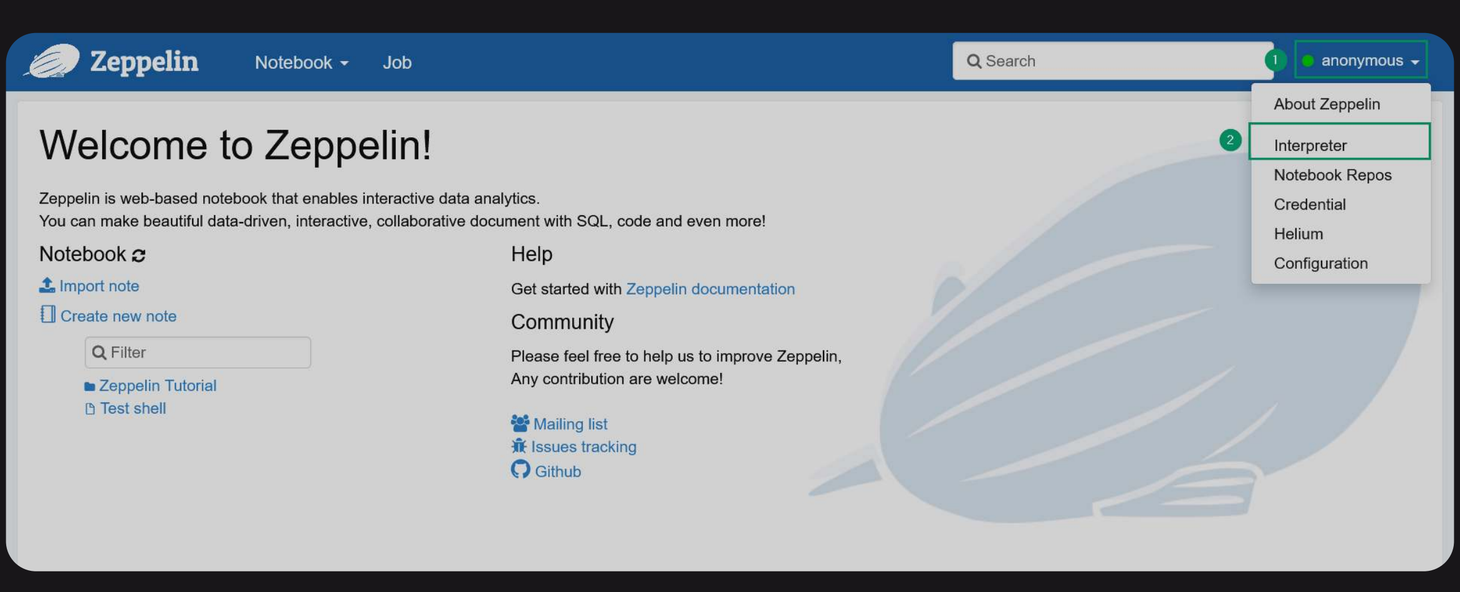
Do you want to update this interpreter and restart with new settings?

Confirming saving the interpreter

Method 2. Adding an interpreter from scratch

This method assumes adding a new interpreter based on the existing one. For this, select the current interpreter as a group (in this example — *jdbc*) and change its default properties to your own. All the libraries available for the source interpreter are also available for the newly created one. To call such an interpreter from a note, you should use the new interpreter name: `%phoenix`.

1. Select the ***Interpreter*** item in the account menu placed in the top right corner of the Zeppelin home page.



Zeppelin
Notebook
Job
Search
1
anonymous
About Zeppelin
2
Interpreter
Notebook Repos
Credential
Helium
Configuration

Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Notebook
Import note
Create new note
Filter
Zeppelin Tutorial
Test shell
Help

Get started with Zeppelin documentation

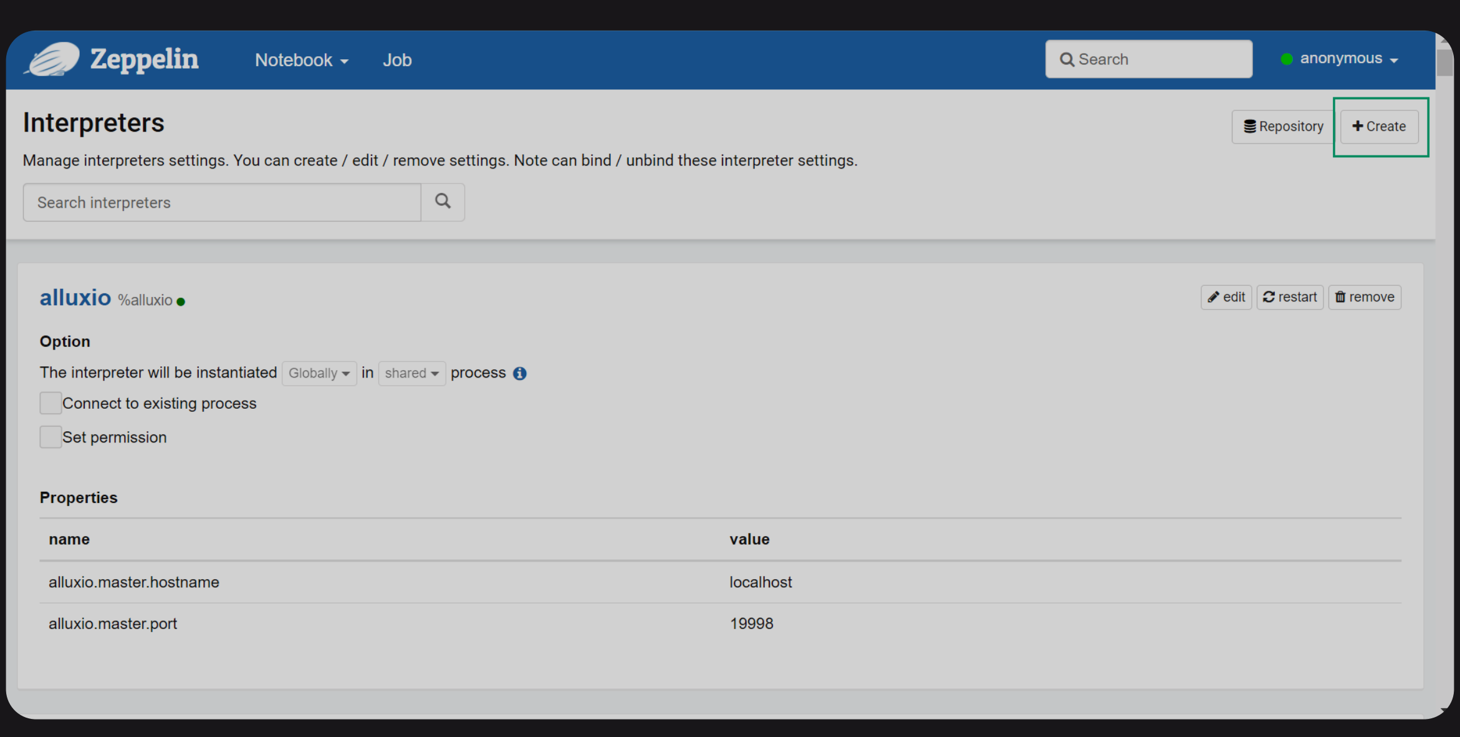
Community

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Mailing list
Issues tracking
Github

Switching to the list of interpreters

2. Click ***Create***.



Zeppelin
Notebook
Job
Search
anonymous
Interpreters
Repository
Create
Search interpreters
alluxio %alluxio
edit
restart
remove
Option

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

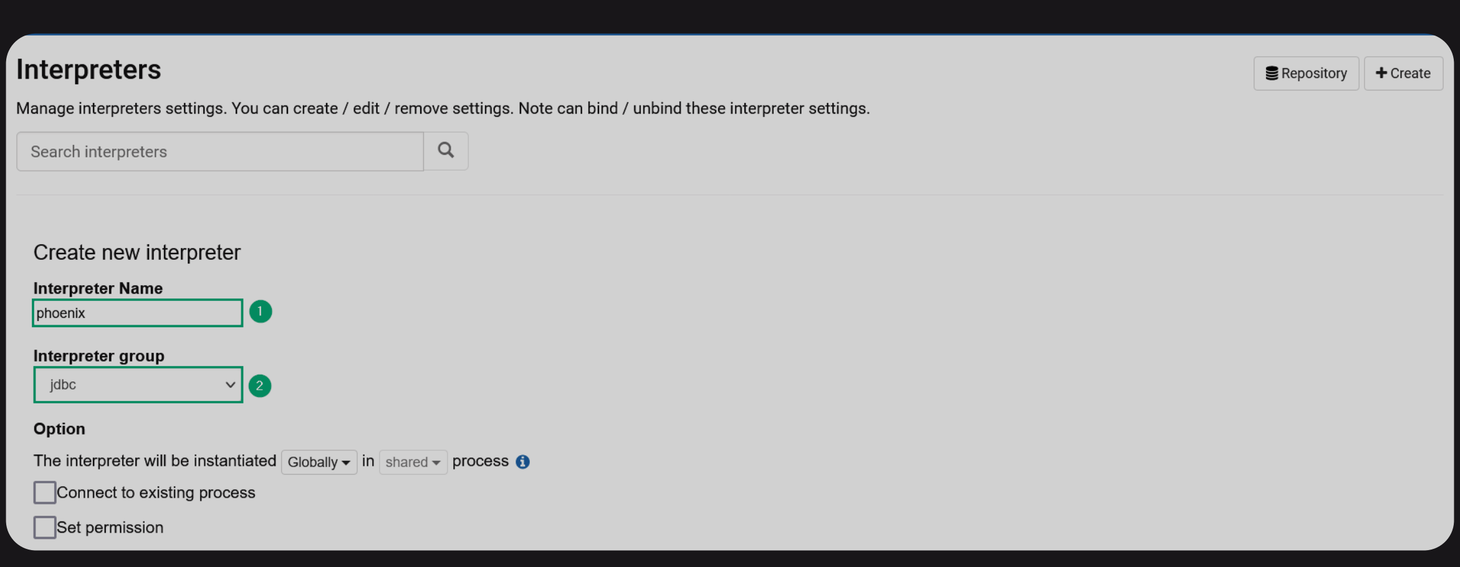
☐ Set permission

Properties

name	value
alluxio.master.hostname	localhost
alluxio.master.port	19998

Creating a new interpreter

3. Enter `phoenix` in the ***Interpreter Name*** field and select `jdbc` in the ***Interpreter group***.



Interpreters
Repository
Create
Search interpreters
1
2
Option

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

Create new interpreter

Interpreter Name: phoenix

Interpreter group: jdbc

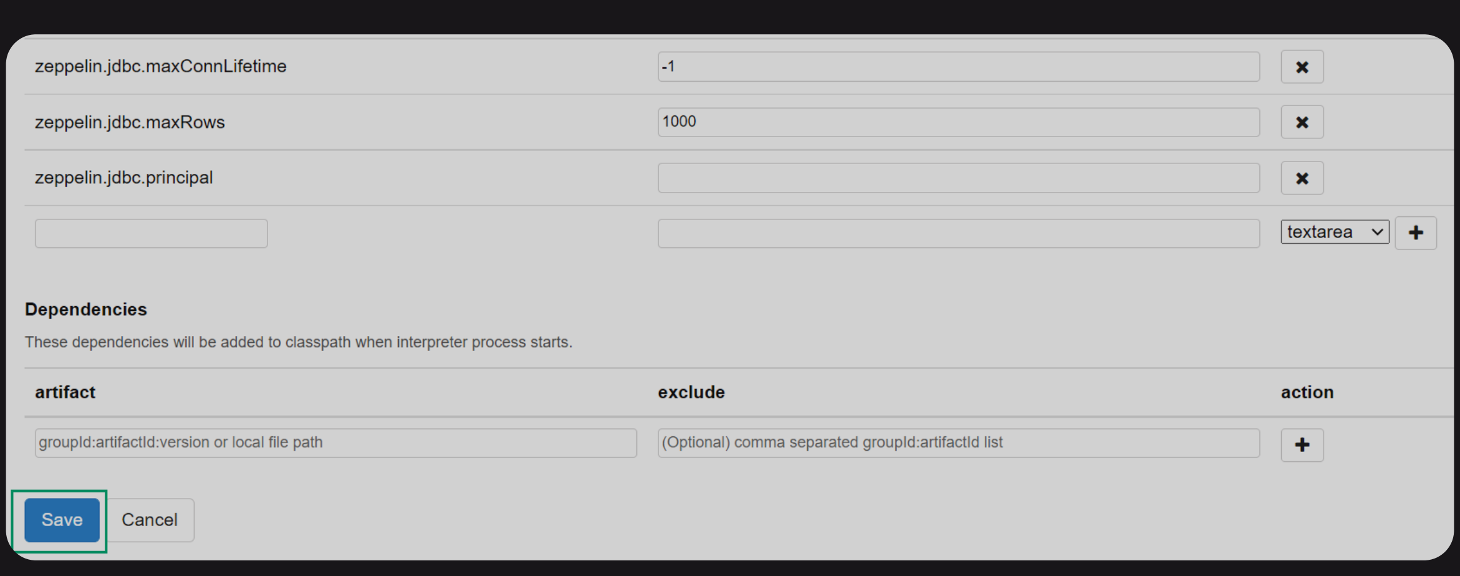
The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

☐ Set permission

Filling a name and a group

4. Clear the values of the following fields: `default.driver`, `default.url`, `default.user`, `default.password`. Fill them with the new values according to the table from the previous section.
5. Click ***Save***. The new interpreter is ready.



textarea
+
Dependencies
artifact
exclude
action
Save
Cancel

zeppelin.jdbc.maxConnLifetime -1

zeppelin.jdbc.maxRows 1000

zeppelin.jdbc.principal

These dependencies will be added to classpath when interpreter process starts.

groupId:artifactId:version or local file path

(Optional) comma separated groupId:artifactId list

Saving the Phoenix interpreter

> **NOTE**
>
> The example of creating a note using the described interpreters can be found in the article Create a note using Phoenix.

Install a third-party interpreter

Daria Barysheva

Contents

To Table of Contents

Step 1. Load interpreter libraries

Step 2. Add an interpreter using web UI

Zeppelin allows not only to work with community-managed interpreters, but also to install other third-party interpreters from Maven repositories. This article describes the example of installing an interpreter for Apache Solr. Make sure that this service is added to your ADH cluster before making the following steps.

NOTE

You can read more information about installing interpreters in the Zeppelin documentation.

Step 1. Load interpreter libraries

1. Connect to the Zeppelin Server (via SSH, etc.).
2. Run the following command to load libraries for the new `solr` interpreter:

```
$ sudo ZEPPELIN_INTERPRETER_DEP_MVNREPO=https://repo1.maven.org/maven2 /usr/lib/zeppelin/bin/install-interpreter.sh --name solr --
artifact com.lucidworks.zeppelin:zeppelin-solr:0.1.6
```

The result should be like this:

```
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512m; support was removed in 8.0
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/zeppelin/lib/interpreter/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.
class]
SLF4J: Found binding in [jar:file:/usr/lib/zeppelin/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Can't find interpreter list /etc/zeppelin/conf/interpreter-list
Install solr(com.lucidworks.zeppelin:zeppelin-solr:0.1.6) to /usr/lib/zeppelin/interpreter/solr ...
Interpreter solr installed under /usr/lib/zeppelin/interpreter/solr.

1. Restart Zeppelin
2. Create interpreter setting in 'Interpreter' menu on Zeppelin GUI
3. Then you can bind the interpreter on your note
```

TIP

The environment variable `ZEPPELIN_INTERPRETER_DEP_MVNREPO` is used to avoid the errors with dependencies resolution that can occur in Zeppelin versions older than 0.9.0. You can also assign the value `https://repo1.maven.org/maven2` to the environment variable `ZEPPELIN_INTERPRETER_DEP_MVNREPO` in the configuration file */etc/zeppelin/conf/zeppelin-env.sh* and restart Zeppelin Server.

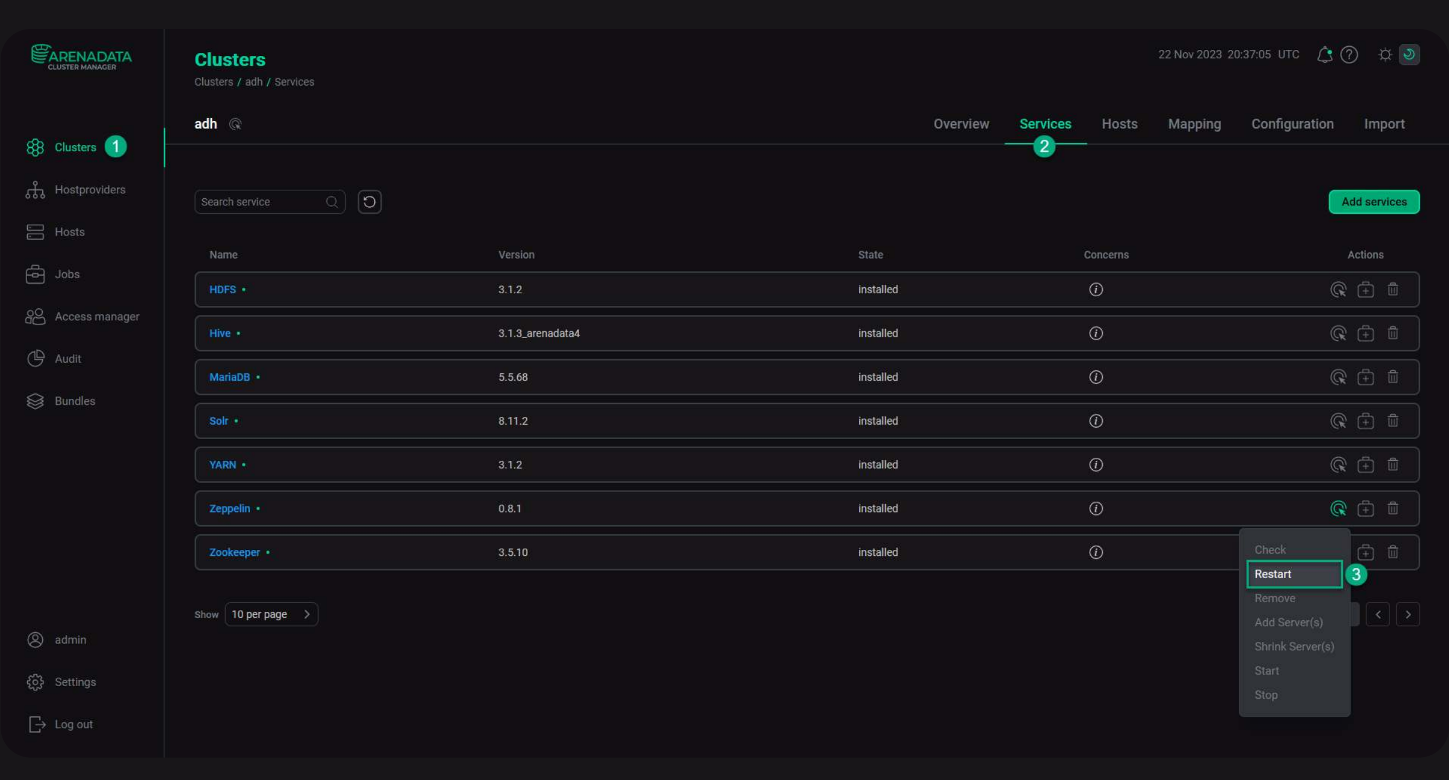
3. Verify that the folder that corresponds to the new `solr` interpreter is added:

```
$ ls -la /usr/lib/zeppelin/interpreter/solr
```

Sample output

```
total 39440
drwxr-xr-x.  2 root root     4096 Dec 10 09:12 .
drwxr-xr-x. 31 root root     4096 Dec 10 09:12 ..
-rw-r--r--.  1 root root    89616 Dec  6 15:58 aether-api-1.12.jar
-rw-r--r--.  1 root root    24449 Dec  6 15:58 aether-connector-file-1.12.jar
-rw-r--r--.  1 root root    29646 Dec  6 15:58 aether-connector-wagon-1.12.jar
-rw-r--r--.  1 root root   129254 Dec  6 15:58 aether-impl-1.12.jar
-rw-r--r--.  1 root root    15430 Dec  6 15:58 aether-spi-1.12.jar
-rw-r--r--.  1 root root   126629 Dec  6 15:58 aether-util-1.12.jar
-rw-r--r--.  1 root root   622849 Dec  6 15:58 bcpkix-jdk15on-1.52.jar
-rw-r--r--.  1 root root  2902942 Dec  6 15:58 bcprov-jdk15on-1.52.jar
-rw-r--r--.  1 root root    30085 Dec  6 15:58 commons-codec-1.2.jar
-rw-r--r--.  1 root root   358048 Dec  6 15:58 commons-configuration-1.9.jar
-rw-r--r--.  1 root root    54423 Dec  6 15:58 commons-exec-1.3.jar
-rw-r--r--.  1 root root   305001 Dec  6 15:58 commons-httpclient-3.1.jar
-rw-r--r--.  1 root root   208700 Dec  6 15:58 commons-io-2.5.jar
-rw-r--r--.  1 root root   279193 Dec  6 15:58 commons-lang-2.5.jar
-rw-r--r--.  1 root root    60686 Dec  6 15:58 commons-logging-1.1.1.jar
-rw-r--r--.  1 root root  2213560 Dec  6 15:58 commons-math3-3.6.1.jar
-rw-r--r--.  1 root root   109568 Dec  6 15:58 commons-pool2-2.3.jar
-rw-r--r--.  1 root root   258075 Dec  6 15:58 gson-2.8.9.jar
-rw-r--r--.  1 root root    26742 Dec  6 15:58 gson-extras-0.2.1.jar
-rw-r--r--.  1 root root  2172168 Dec  6 15:58 guava-15.0.jar
-rw-r--r--.  1 root root   747794 Dec  6 15:58 httpclient-4.5.3.jar
-rw-r--r--.  1 root root   323824 Dec  6 15:58 httpcore-4.4.6.jar
-rw-r--r--.  1 root root    41029 Dec  6 15:58 httpmime-4.5.3.jar
-rw-r--r--.  1 root root    16516 Dec  6 15:58 jcl-over-slf4j-1.7.24.jar
-rw-r--r--.  1 root root   213911 Dec  6 15:58 jline-2.12.1.jar
-rw-r--r--.  1 root root   281579 Dec  6 15:58 jsoup-1.6.1.jar
-rw-r--r--.  1 root root   227712 Dec  6 15:58 libthrift-0.9.2.jar
-rw-r--r--.  1 root root   489884 Dec  6 15:58 log4j-1.2.17.jar
-rw-r--r--.  1 root root    55639 Dec  6 15:58 maven-aether-provider-3.0.3.jar
-rw-r--r--.  1 root root    51908 Dec  6 15:58 maven-artifact-3.0.jar
-rw-r--r--.  1 root root   163439 Dec  6 15:58 maven-model-3.0.3.jar
-rw-r--r--.  1 root root   149093 Dec  6 15:58 maven-model-builder-3.0.3.jar
-rw-r--r--.  1 root root    48920 Dec  6 15:58 maven-plugin-api-3.0.jar
-rw-r--r--.  1 root root    30088 Dec  6 15:58 maven-repository-metadata-3.0.3.jar
-rw-r--r--.  1 root root    27948 Dec  6 15:58 noggit-0.8.jar
-rw-r--r--.  1 root root    47058 Dec  6 15:58 plexus-classworlds-2.4.jar
-rw-r--r--.  1 root root     4211 Dec  6 15:58 plexus-component-annotations-1.5.5.jar
-rw-r--r--.  1 root root    61072 Dec  6 15:58 plexus-interpolation-1.14.jar
-rw-r--r--.  1 root root   223943 Dec  6 15:58 plexus-utils-2.0.7.jar
-rw-r--r--.  1 root root 15612191 Dec  6 15:58 scala-compiler-2.11.12.jar
-rw-r--r--.  1 root root  5749423 Dec  6 15:58 scala-library-2.11.12.jar
-rw-r--r--.  1 root root   423753 Dec  6 15:58 scala-parser-combinators_2.11-1.0.4.jar
-rw-r--r--.  1 root root   671138 Dec  6 15:58 scala-xml_2.11-1.0.5.jar
-rw-r--r--.  1 root root   489453 Dec  6 15:58 sisu-guice-3.0.2-no_aop.jar
-rw-r--r--.  1 root root   247361 Dec  6 15:58 sisu-inject-bean-2.2.2.jar
-rw-r--r--.  1 root root   203536 Dec  6 15:58 sisu-inject-plexus-2.2.2.jar
-rw-r--r--.  1 root root    32119 Dec  6 15:58 slf4j-api-1.7.10.jar
-rw-r--r--.  1 root root     8866 Dec  6 15:58 slf4j-log4j12-1.7.10.jar
-rw-r--r--.  1 root root  1870892 Dec  6 15:58 solr-solrj-7.5.0.jar
-rw-r--r--.  1 root root   161867 Dec  6 15:58 stax2-api-3.1.4.jar
-rw-r--r--.  1 root root    10494 Dec  6 15:58 wagon-http-1.0.jar
-rw-r--r--.  1 root root    14945 Dec  6 15:58 wagon-http-lightweight-1.0.jar
-rw-r--r--.  1 root root    24599 Dec  6 15:58 wagon-http-shared-1.0.jar
-rw-r--r--.  1 root root    53186 Dec  6 15:58 wagon-provider-api-1.0.jar
-rw-r--r--.  1 root root   486013 Dec  6 15:58 woodstox-core-asl-4.4.1.jar
-rw-r--r--.  1 root root   100679 Dec  6 15:58 zeppelin-solr-0.1.6.jar
-rw-r--r--.  1 root root   887738 Dec  6 15:58 zookeeper-3.4.11.jar
```

4. Restart Zeppelin Server. You can do it using ADCM:
   - Open the ***Clusters*** page.
   - Select your cluster in the common list.
   - Go to ***Services***.
   - Click the ***Restart*** item in the drop-down menu of the ***Zeppelin*** service.



Clusters

Clusters / adh / Services

22 Nov 2023 20:37:05 UTC

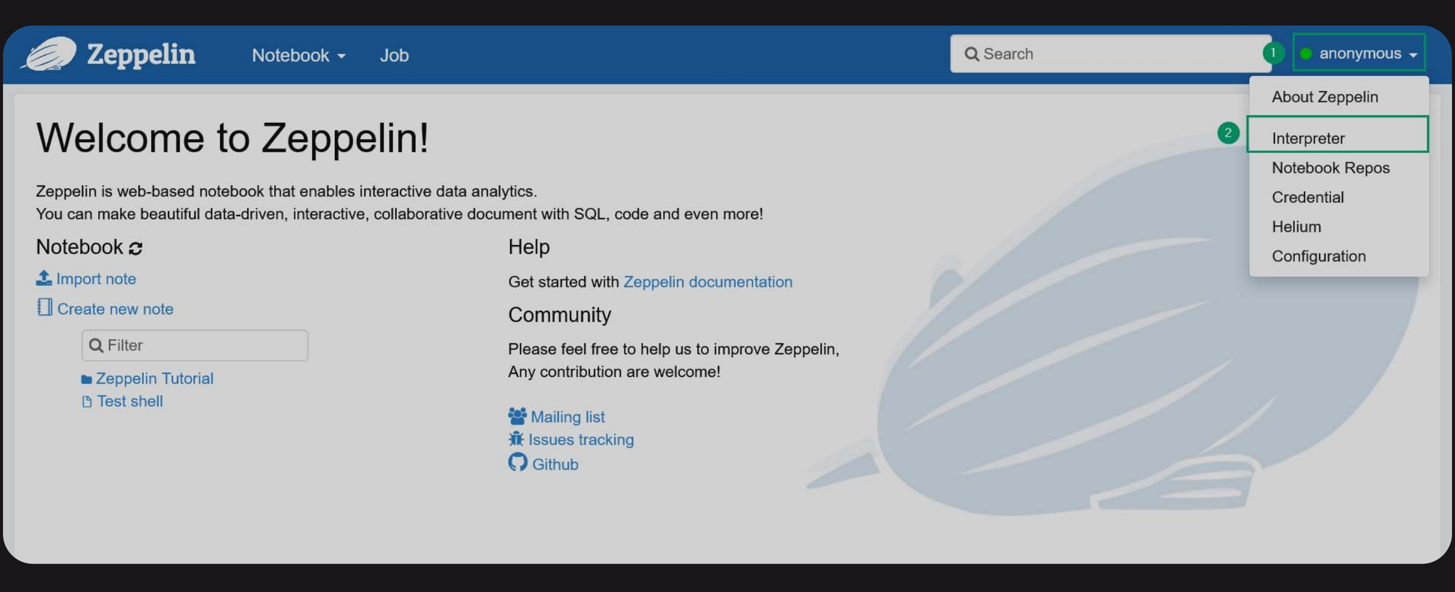
adh

Name	Version	State	Concerns	Actions
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
Solr	8.11.2	installed		
YARN	3.1.2	installed		
Zeppelin	0.8.1	installed		
Zookeeper	3.5.10	installed		

Restarting Zeppelin service via ADCM

Step 2. Add an interpreter using web UI

1. Select the ***Interpreter*** item in the account menu placed in the top right corner of the Zeppelin home page.



Welcome to Zeppelin!

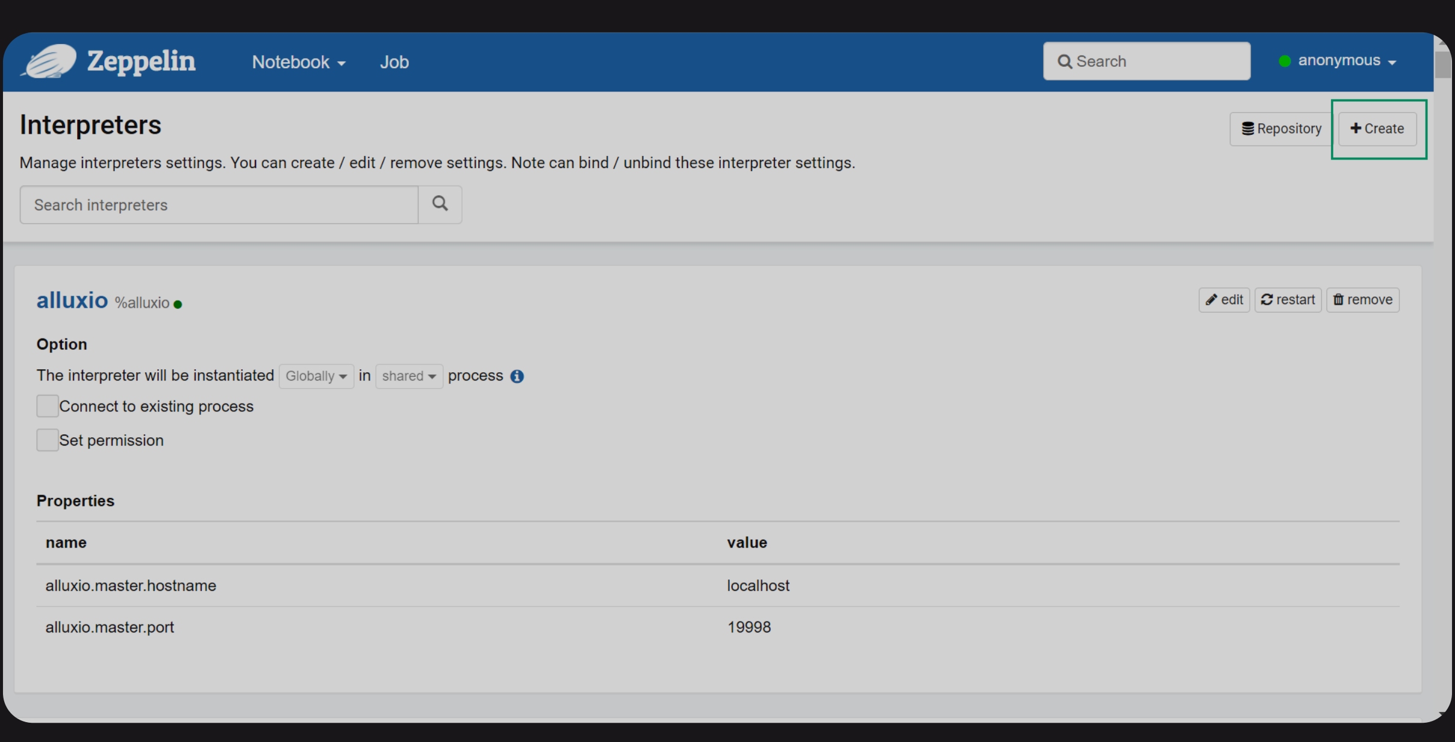
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Get started with Zeppelin documentation

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Switching to the list of interpreters

2. Click ***Create***.



Interpreters

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

alluxio %alluxio

Option

The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

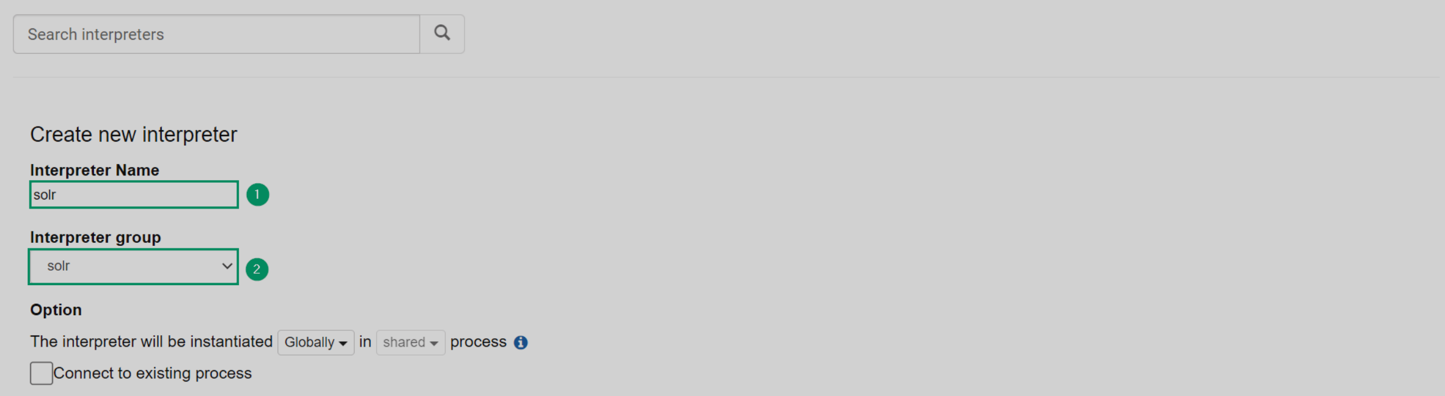
☐ Set permission

Properties

name	value
alluxio.master.hostname	localhost
alluxio.master.port	19998

Creating a new interpreter

3. In the field ***Interpreter Name***, enter `solr` and select `solr` in the ***Interpreter group*** field.



Interpreters

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

Create new interpreter

Interpreter Name: solr

Interpreter group: solr

Option

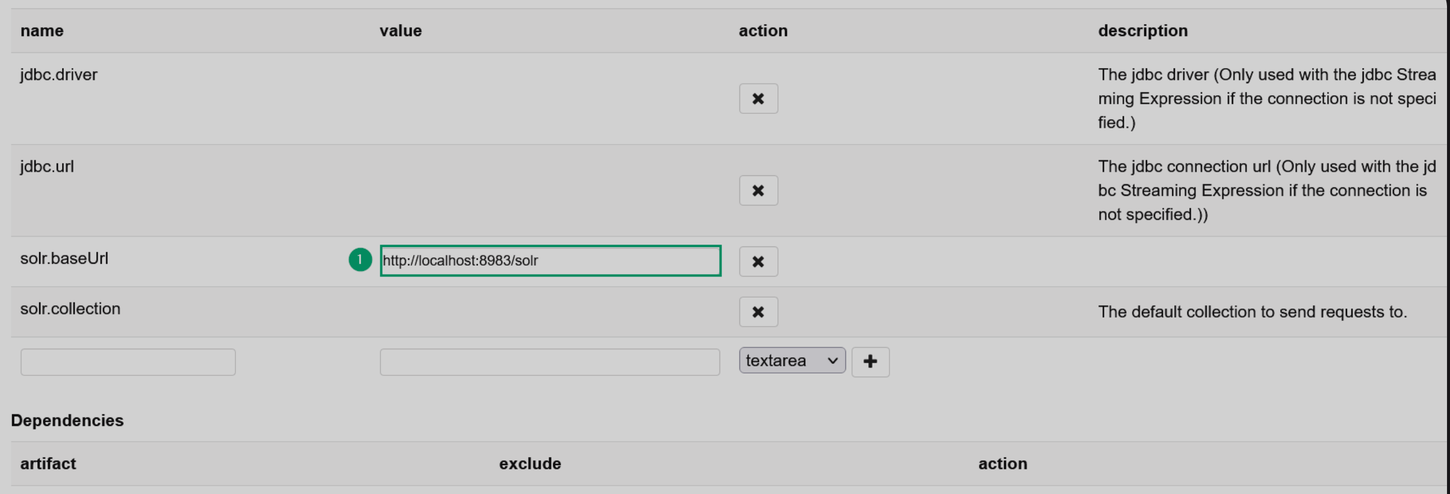
The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

☐ Set permission

Filling a name and a group

4. Set the value of the ***solr.baseUrl*** field to `http://<IP>:8983/solr`, where `<IP>` is the IP-address of your Solr service. If the value can not be edited, delete this field and create the new one with the same name.
5. Click ***Save***. Your interpreter is ready.



Properties

name	value	action	description
jdbc.driver		✖	The jdbc driver (Only used with the jdbc Streaming Expression if the connection is not specified.)
jdbc.url		✖	The jdbc connection url (Only used with the jdbc Streaming Expression if the connection is not specified.))
solr.baseUrl	http://localhost:8983/solr	✖	
solr.collection		✖	The default collection to send requests to.

Dependencies

artifact	exclude	action
groupId:artifactId:version or local file path	(Optional) comma separated groupId:artifactId list	+

Filling the property and saving the interpreter

NOTE

- The example of creating a note using the interpreter described in this section can be found in the Create a note using Solr article.
- You can also use the basic JDBC interpreter for connecting to Solr. If you decide to do this, follow the instructions described in the Add a custom interpreter to a group section and in the Solr documentation.

User-managed interpreters

Konstantin Alpashkin

Contents

To Table of Contents

- Overview
- Usage examples
  - Case: protect a system interpreter from changes during Zeppelin restarts
  - Case: update interpreter config after toggling Kerberos/SSL/HA
  - Case: submit interpreter configuration to ADCM as a file
  - Case: reset all custom interpreters to default

Overview

This section describes the specifics of working with custom Zeppelin interpreters tagged with the `user-managed` property and provides common scenarios of using this option.

By default, user-defined interpreter properties get lost when ADCM is used to restart Zeppelin. To preserve custom properties between the restarts, the following should be observed:

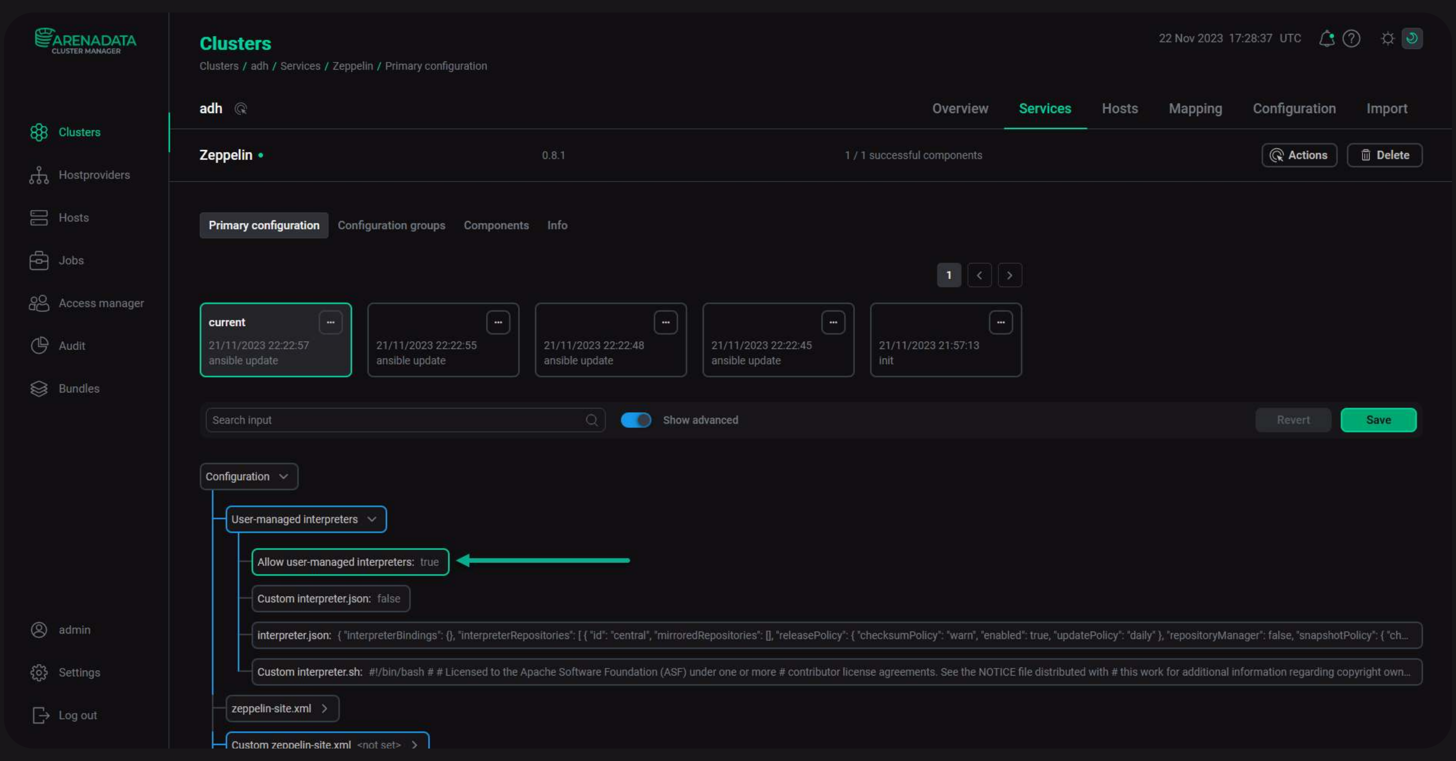
- The `Allow user-managed interpreters` option is enabled on the Zeppelin configuration page in ADCM (enabled by default).
- For those interpreters which properties you want to preserve, the `user-managed=true` property must be set in Zeppelin UI.

More details on using these options are described below.

Zeppelin interpreters can be of two types:

- **System**. These are the interpreters distributed with ADH bundles and available in the Zeppelin web UI by default.
- **Custom**. The interpreters that were manually added by a user.

Using Zeppelin UI, one can add custom key/value properties for an interpreter to tune its behavior. However, when you restart Zeppelin via ADCM, such custom properties get lost. To preserve custom properties between the restarts, use the ***Allow user-managed interpreters*** option in ADCM UI as shown below.



ARENADATA
CLUSTER MANAGER
Clusters
Clusters / adh / Services / Zeppelin / Primary configuration
22 Nov 2023 17:28:37 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
adh
Overview
Services
Hosts
Mapping
Configuration
Import
Zeppelin
0.8.1
1 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Components
Info
current
21/11/2023 22:22:57
ansible update
21/11/2023 22:22:55
ansible update
21/11/2023 22:22:48
ansible update
21/11/2023 22:22:45
ansible update
21/11/2023 21:57:13
init
Search input
Show advanced
Revert
Save

Configuration

User-managed interpreters

Allow user-managed interpreters: true

Custom interpreter.json: false

interpreter.json: { "interpreterBindings": {}, "interpreterRepositories": [{ "id": "central", "mirroredRepositories": [], "releasePolicy": { "checksumPolicy": "warn", "enabled": true, "updatePolicy": "daily" }, "repositoryManager": false, "snapshotPolicy": { "ch...

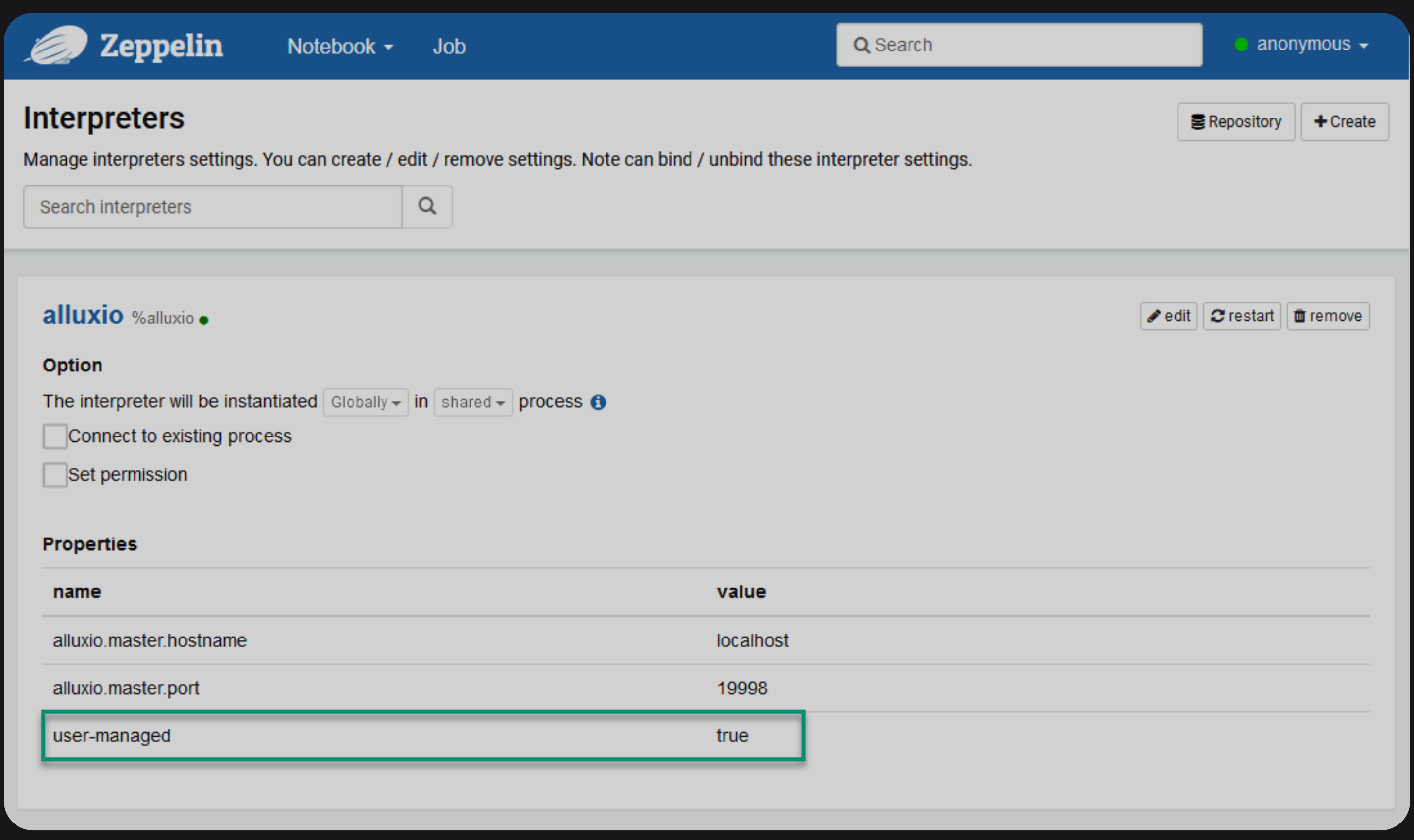
Custom interpreter.sh: #!/bin/bash # # Licensed to the Apache Software Foundation (ASF) under one or more # contributor license agreements. See the NOTICE file distributed with # this work for additional information regarding copyright own...

zeppelin-site.xml

Custom zeppelin-site.xml <not set>

Allow user-managed interpreters

If the ***Allow user-managed interpreters*** option is enabled, ADCM picks up the interpreter configuration from the Zeppelin host for all the interpreters that have `user-managed=true`.



Zeppelin
Notebook
Job
Search
anonymous
Repository
Create
Search interpreters
edit
restart
remove

Interpreters

Manage interpreters settings. You can create / edit / remove settings. Note can bind / unbind these interpreter settings.

alluxio %alluxio

Option

The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

☐ Set permission

Properties

name	value
alluxio.master.hostname	localhost
alluxio.master.port	19998
user-managed	true

User-managed option in Zeppelin UI

If ***Allow user-managed interpreters*** is disabled, ADCM will create a default list of system interpreters when restarting Zeppelin.

Usage examples

Below are examples of common tasks that demonstrate the use of the user-managed interpreters.

Case: protect a system interpreter from changes during Zeppelin restarts

If you do not want an interpreter's settings to change when Zeppelin is restarted via ADCM, set the `user-managed=true` property for this interpreter in Zeppelin UI. If ***Allow user-managed interpreters*** is enabled in ADCM, custom interpreter settings will not be changed after the restart.

Case: update interpreter config after toggling Kerberos/SSL/HA

Assume you have just enabled or disabled Kerberos/SSL/HA for an ADH service. Typically, such operations lead to a change in the JDBC connection string. Thus, if your interpreter has a custom property that stores a connection string, this string must be updated. A quick way to update the interpreter is as follows:

1. Disable the ***Allow user-managed interpreters*** option in ADCM or set `user-managed=false` in Zeppelin web UI for those interpreters which parameters should be updated.
2. Restart Zeppelin.

After Zeppelin restarts, ADCM will re-create the default interpreters from a template, using the correct JDBC string value.

Case: submit interpreter configuration to ADCM as a file

You can feed custom definitions of *interpreter.json*/*interpreter.sh* to ADCM. For this, provide your custom files contents on the Zeppelin configuration page and save the configuration. When Zeppelin restarts, it will initialize interpreters based on the provided *interpreter.json*/*interpreter.sh*.

Case: reset all custom interpreters to default

To reset all interpreters to the default state, do the following:

1. Disable the ***Allow user-managed interpreters*** option (enabled by default) on the Zeppelin configuration page in ADCM.
2. Disable the ***Custom interpreter.json***/***interpreter.sh*** options to restore the default configuration.
3. Restart the Zeppelin service.

Once you do this, after Zeppelin restarts, all interpreters will be re-created with default configs.

Write a new Zeppelin interpreter

Konstantin Alpashkin

Contents

To Table of Contents

Flow overview

Step 1. Clone the Zeppelin project

Step 2. Create a new interpreter module

Step 3. Build the interpreter JAR

Step 4. Add the interpreter JAR to Zeppelin

This article describes the process of creating a custom Zeppelin interpreter from scratch. For information on how to add a new Zeppelin interpreter to an existing interpreter group, see the Add a custom interpreter to a group article.

Flow overview

In Zeppelin, to create a new interpreter, one has to extend the Interpreter abstract class and put the compiled JAR on the ADH host with the Zeppelin Server component. Developing a full-fledged interpreter capable of processing user code from notes requires writing complex logic and thorough testing, which is beyond the scope of this article. Instead, the scenario below describes a mock interpreter implementation with stub methods, which is still enough to demonstrate the interpreter's activity. The steps demonstrated in the scenario work for any kind of interpreter and show one of the possible approaches to creating a custom interpreter.

At a high level, the scenario comprises the following steps:

1. Clone the Zeppelin project.
2. Create a new interpreter from an existing one.
3. Implement the `org.apache.zeppelin.interpreter.Interpreter` class and compile the module.
4. Put the interpreter JAR on an ADH host.

Step 1. Clone the Zeppelin project

The scenario assumes using the open-source Zeppelin project as a base to create a new interpreter. This approach facilitates dependency and file hierarchy management.

1. Clone the Zeppelin repository.

```
$ git clone https://github.com/apache/zeppelin/tree/master
```

2. Checkout the branch that corresponds to your Zeppelin Service version. For example:

```
$ cd zeppelin
$ git checkout 0.11.2
```

Step 2. Create a new interpreter module

In this scenario, the new interpreter is created by copying an existing interpreter module and modifying its content.

1. Copy an existing interpreter module, for example `file`.

```
$ cp -r file ./mycustominterpreter
```

2. In the copied interpreter module, delete all classes and create a new Java class `CustomInterpreter` that extends `org.apache.zeppelin.interpreter.Interpreter`. In this scenario, the `CustomInterpreter` class is a mock implementation that does not process user code from notes, but creates log entries when Zeppelin invokes its methods.

Click to view sample CustomInterpreter.java

```
package io.arenadata.zeppelin.mycustominterpreter;

import org.apache.zeppelin.interpreter.Interpreter;
import org.apache.zeppelin.interpreter.InterpreterContext;
import org.apache.zeppelin.interpreter.InterpreterException;
import org.apache.zeppelin.interpreter.InterpreterResult;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.Properties;

public class CustomInterpreter extends Interpreter {
    public CustomInterpreter(Properties properties) {
        super(properties);
        Logger logger = LoggerFactory.getLogger(CustomInterpreter.class);
    }

    @Override
    public void open() throws InterpreterException {
        logger.info("Stub method called: open()");
    }

    @Override
    public void close() throws InterpreterException {
        logger.info("Stub method called: close()");
    }

    @Override
    public InterpreterResult interpret(String st, InterpreterContext context) throws InterpreterException {
        logger.info("Stub method called: interpret()");
        return null;
    }

    @Override
    public void cancel(InterpreterContext context) throws InterpreterException {
        logger.info("Stub method called: cancel()");
    }

    @Override
    public FormType getFormType() throws InterpreterException {
        logger.info("Stub method called: getFormType()");
        return null;
    }

    @Override
    public int getProgress(InterpreterContext context) throws InterpreterException {
        logger.info("Stub method called: getProgress()");
        return 0;
    }
}
```

TIP

To explore interpreter implementations provided with Zeppelin out-of-the box, see the Zeppelin GitHub repository.

Your interpreter module directory must have the following structure:

```
zeppelin/
+---alluxio
+---angular
+---bigquery
+---bin
+---build-tools
+---cassandra
+---conf
+---custominterpreter
|   |   pom.xml
|   |
|   +---src
|   |   +---main
|   |   |   +---java
|   |   |   |   \---io
|   |   |   |       \---arenadata
|   |   |   |           \---zeppelin
|   |   |   |               \---mycustominterpreter
|   |   |   |                       CustomInterpreter.java <-- Interpreter logic here
|   |   |   |
|   |   |   \---resources
|   |   |           interpreter-setting.json
|   |   |
|   |   \---test
|   |
|   \---target
+---flink
+---flink-cmd
+---groovy
...
// other modules
```

3. Since the interpreter module was copied, it needs a bit of refactoring, namely:
   - Add the new interpreter to the list of modules in the parent POM (*zeppelin/pom.xml*). For example:

```
<module>mycustominterpreter</module>
```

   - In the new module's POM (*zeppelin/mycustominterpreter/pom.xml*), update the `artifactId` and `interpreter.name` values to correspond to the name of the interpreter.
   - In *zeppelin/mycustominterpreter/src/main/resources/interpreter-setting.json*, update the `className` field to point to the new interpreter class (`CustomInterpreter`). Also, provide new `name` and `group` values. These will be used for invoking the interpreter in a note using the `%<interpreter.group>.<interpreter.name>` notation. Also, using this file, you can define a list of interpreter properties that will be displayed in the Zeppelin UI. For more information about options like `editOnDblClick` and `completionSupport`, see Zeppelin documentation.
   - Fix package names.
   - Drop test classes.

Step 3. Build the interpreter JAR

Build the Zeppelin project with Maven:

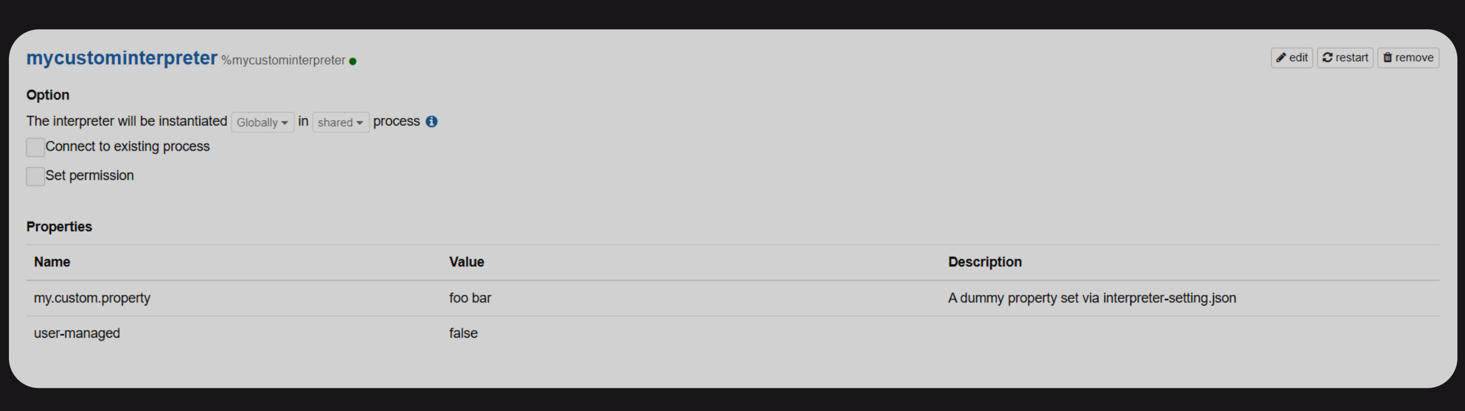
```
$ mvn clean package -DskipTests
```

As a result, the interpreter JAR is available in the module's *target* dir. This must be an Uber-JAR, including the dependencies, with the structure as shown below:

```
mycustominterpreter.jar
|   interpreter-setting.json
|
+---com
|   \---google
|       \--- ...
+---javassist
|   \--- ...
+---javax
|   \--- ...
+---jersey
|   \--- ...
+---META-INF
|   \--- ...
+---io
        +---arenadata
                \---zeppelin
                        \---mycustominterpreter
                                \---CustomInterpreter.class
    +---aopalliance
    +---apache
    +---slf4j
...
```

Step 4. Add the interpreter JAR to Zeppelin

1. Copy the new interpreter JAR to the */usr/lib/zeppelin/interpreter/* location on the ADH host with the Zeppelin Server component.
2. Restart the Zeppelin service. After restarting Zeppelin, the newly added interpreter becomes available in the list of Zeppelin interpreters.



Option

The interpreter will be instantiated Globally in shared process

☐ Connect to existing process

☐ Set permission

Properties

Name	Value	Description
my.custom.property	foo bar	A dummy property set via interpreter-setting.json
user-managed	false	

Newly added interpreter

3. Create a new note using the interpreter. When creating a note, select the new interpreter as the default one, or use `%mycustominterpreter` in each paragraph. In this scenario, submitting any data to the test interpreter returns no results but creates log entries. Check logs under */var/log/zeppelin/* to verify the interpreter's activity, for example:

```
$ grep -r "Stub method called:*" /var/log/zeppelin
```

Examples of creating notes in Zeppelin

Daria Barysheva

To Table of Contents

This section shows how to create notes in Zeppelin based on the following interpreters:

- Phoenix
- Solr
- Spark
- Python

TIP

The first example is the most detailed — we recommend to start with it. You can also find more useful examples in the Zeppelin Tutorial section of the Zeppelin Web UI.

Create a note using Phoenix

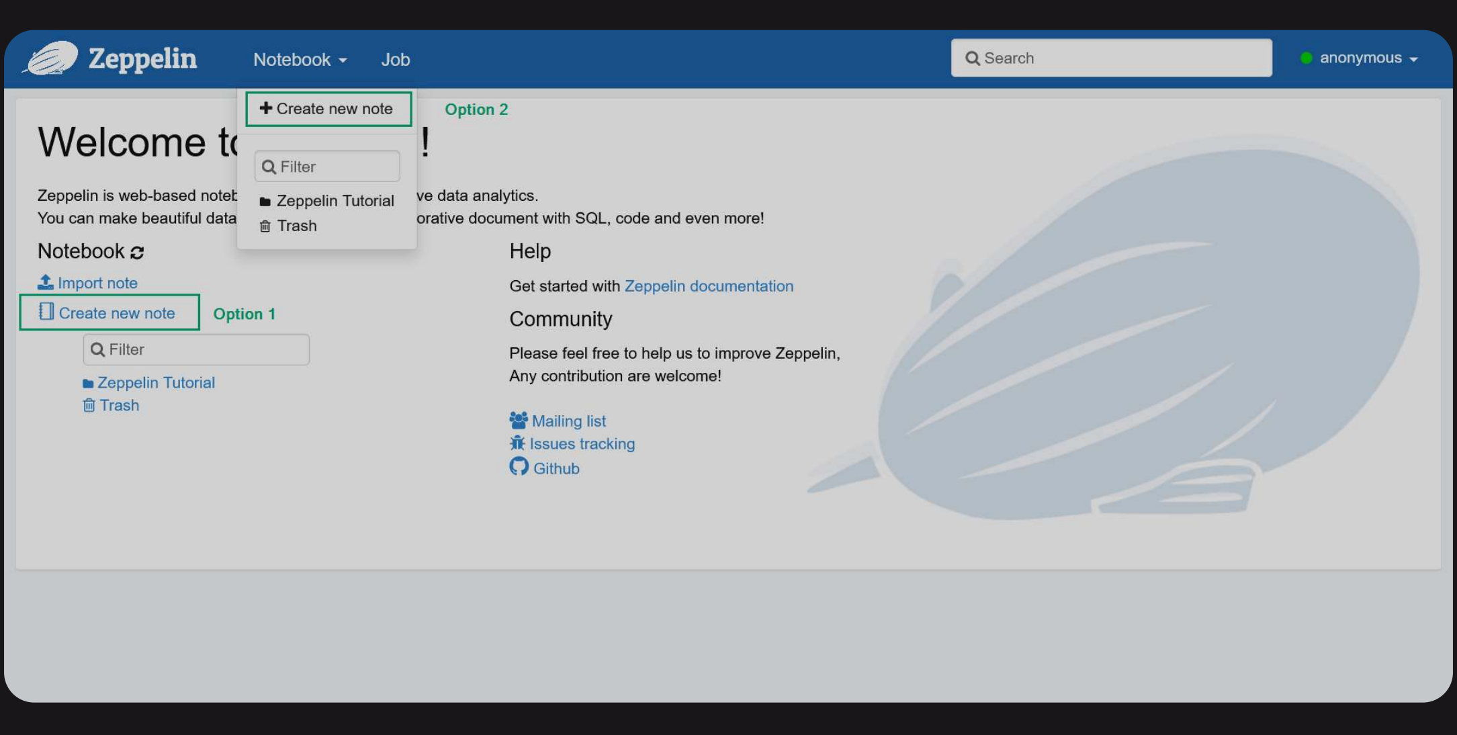
Daria Barysheva

To Table of Contents

NOTE

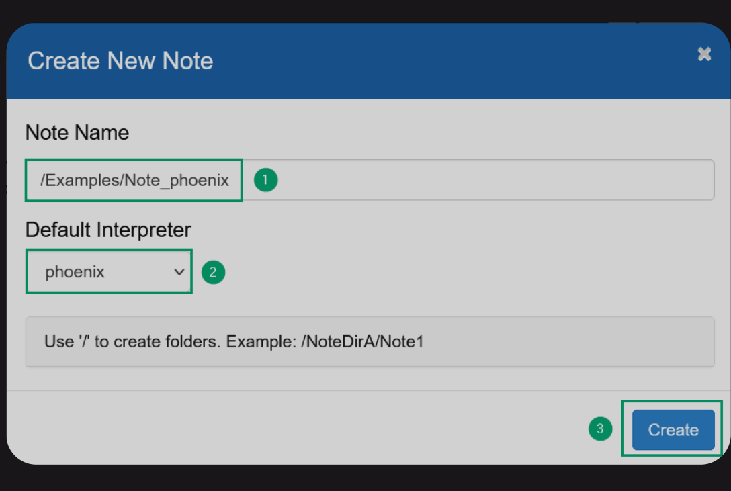
Before creating a note, make sure that your Phoenix interpreter is configured as described in the Add a custom interpreter to a group article.

1. In the Zeppelin web UI, click the ***Create new note*** list item or click the main menu item ***Notebook*** → ***Create new note***.


Zeppelin
Notebook
Job
Search
anonymous
Create new note
Option 2
Filter
Zeppelin Tutorial
Trash
Notebook
Import note
Create new note
Option 1
Help
Community
Mailing list
Issues tracking
Github

Adding a new note

2. In the opened dialog, enter the note name and select the default interpreter that will be used in cases when the interpreter type is not specified explicitly. Notice that you can create different folders for your notes. Then, click ***Create***.


Create New Note
1
2
3
Create

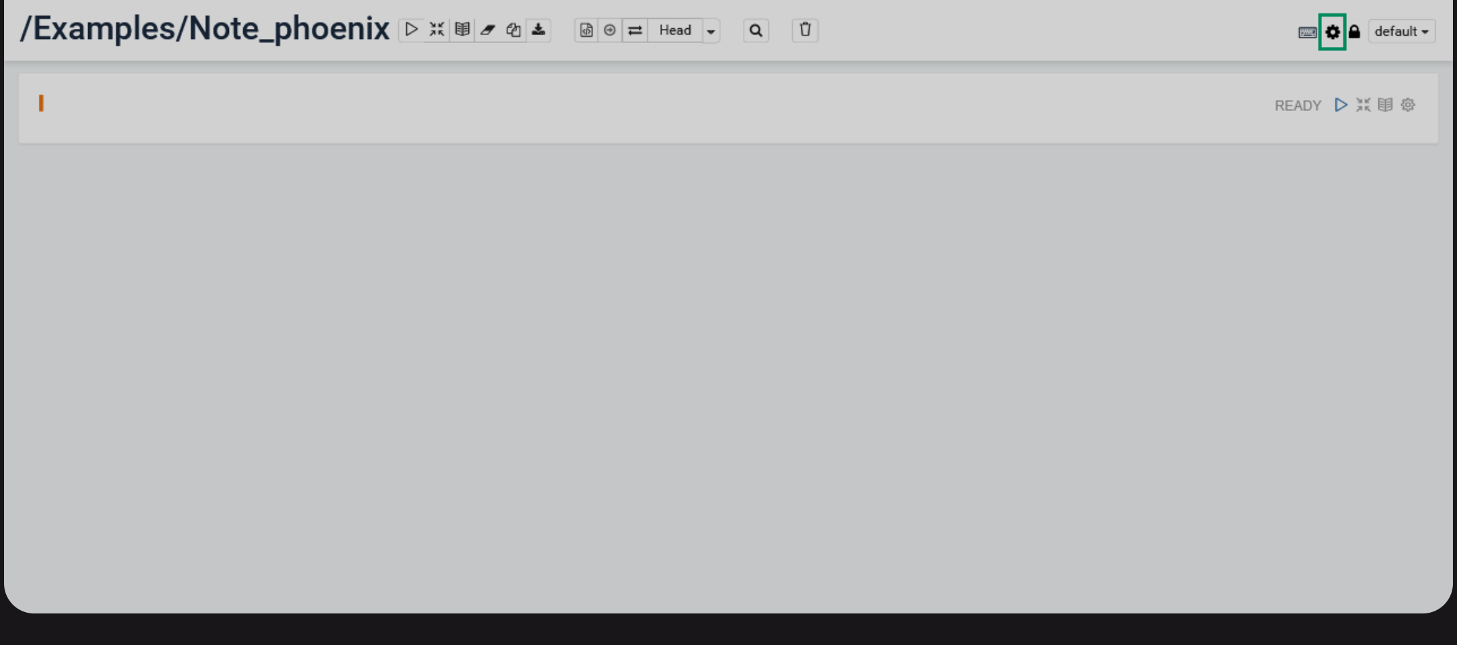
Note Name: /Examples/Note_phoenix

Default Interpreter: phoenix

Use '/' to create folders. Example: /NoteDirA/Note1

Describing a new note

3. In the opened window, click the ***Interpreter binding*** icon to view the available interpreters.

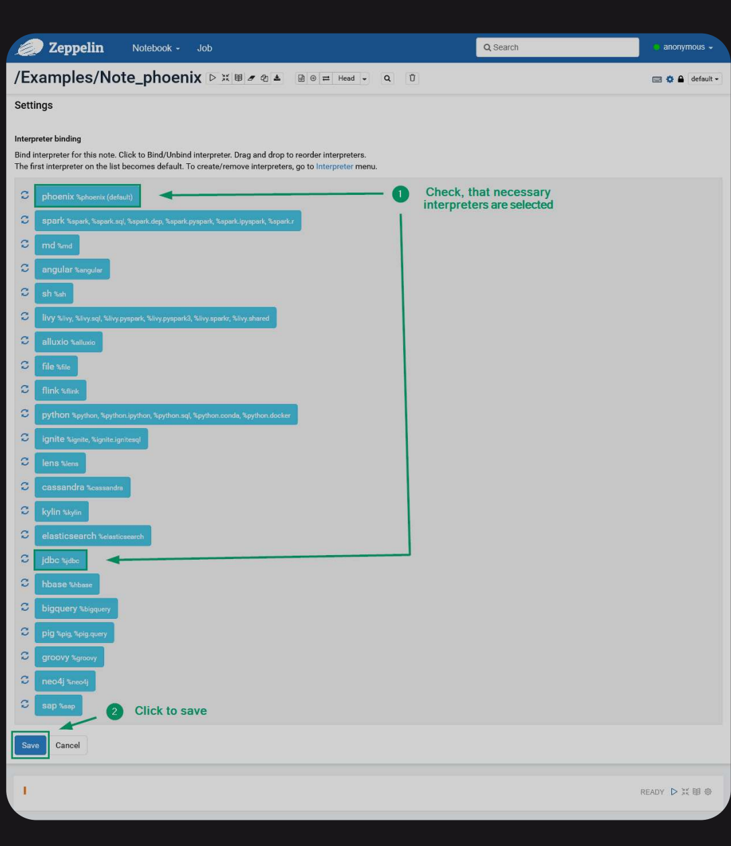

Zeppelin
Notebook
Job
Search
anonymous
/Examples/Note_phoenix
Head
default
READY

Switch to binding interpreters

4. Verify that the interpreters `%jdbc` and `%phoenix` are selected in the opened list, select them otherwise. Click ***Save***.

NOTE

You can use only one interpreter, however this section shows working with both of them.


Settings
Interpreter binding
1 Check, that necessary interpreters are selected
2 Click to save
Save
Cancel

Binding necessary interpreters

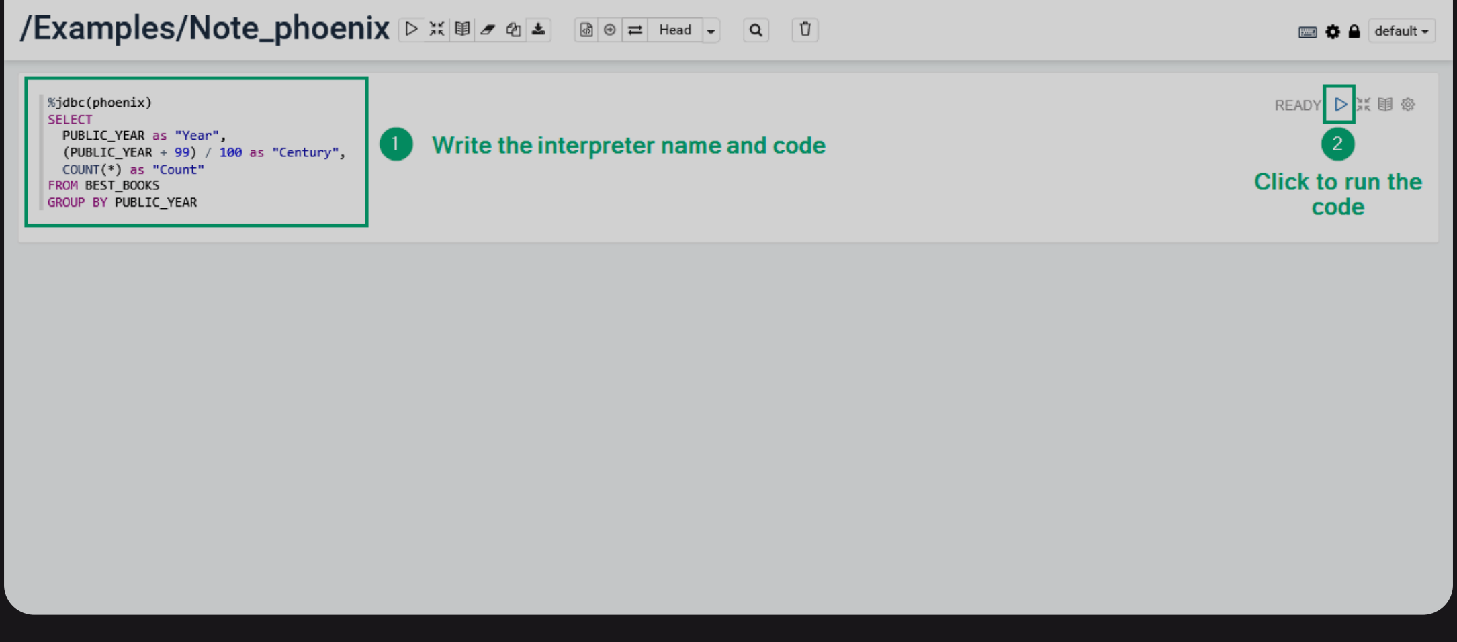
5. First, use the %jdbc interpreter with the settings marked with the `phoenix_` prefix. Enter the following code in the first paragraph of your note. This query returns the number of books grouped by their first publication year.

```
%jdbc(phoenix)
SELECT
  PUBLIC_YEAR as "Year",
  (PUBLIC_YEAR + 99) / 100 as "Century",
  COUNT(*) as "Count"
FROM BEST_BOOKS
GROUP BY PUBLIC_YEAR
```

NOTE

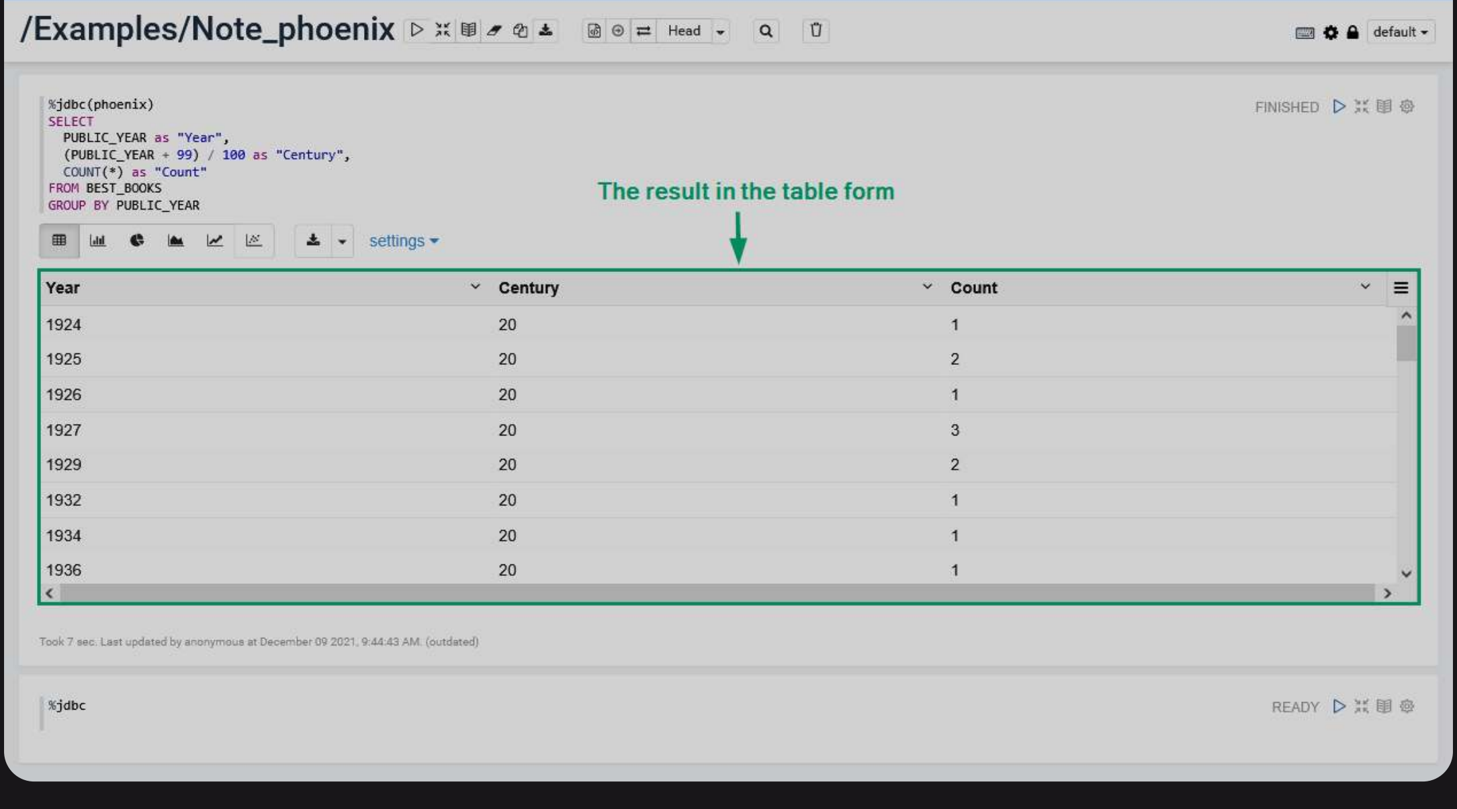
This query uses the `BEST_BOOKS` table. Its creation is described in Bulk loading in Phoenix. If you did not create this table, you can run any other query on your Phoenix tables.

6. To run the code, click ***Run this paragraph***.


Zeppelin
/Examples/Note_phoenix
1 Write the interpreter name and code
READY
2 Click to run the code

Running the code

7. View the query results in the table form.


The result in the table form
FINISHED
settings
READY

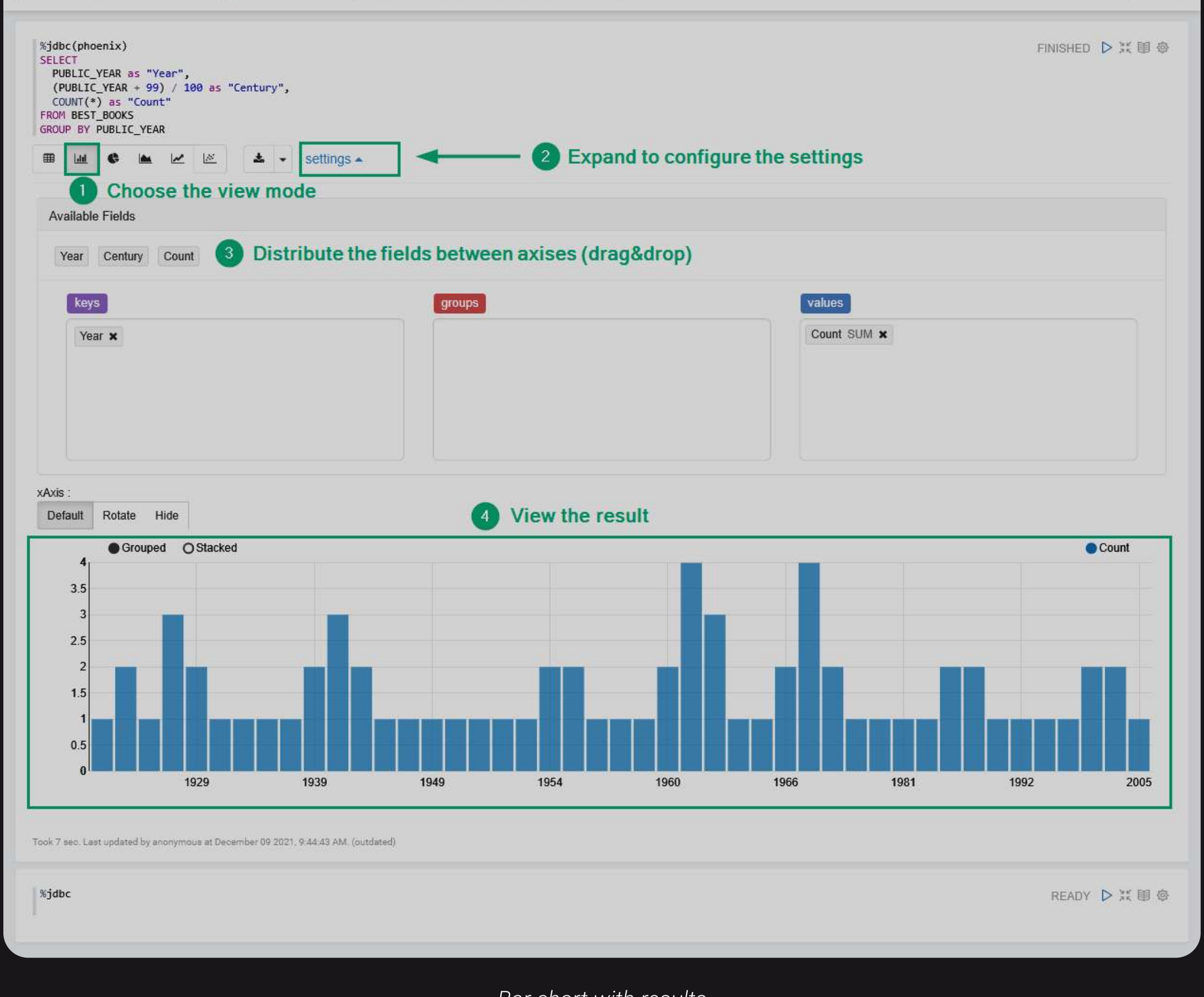
Year	Century	Count
1924	20	1
1925	20	2
1926	20	1
1927	20	3
1929	20	2
1932	20	1
1934	20	1
1936	20	1

Took 7 sec. Last updated by anonymous at December 09 2021, 9:44:43 AM. (outdated)

Results table

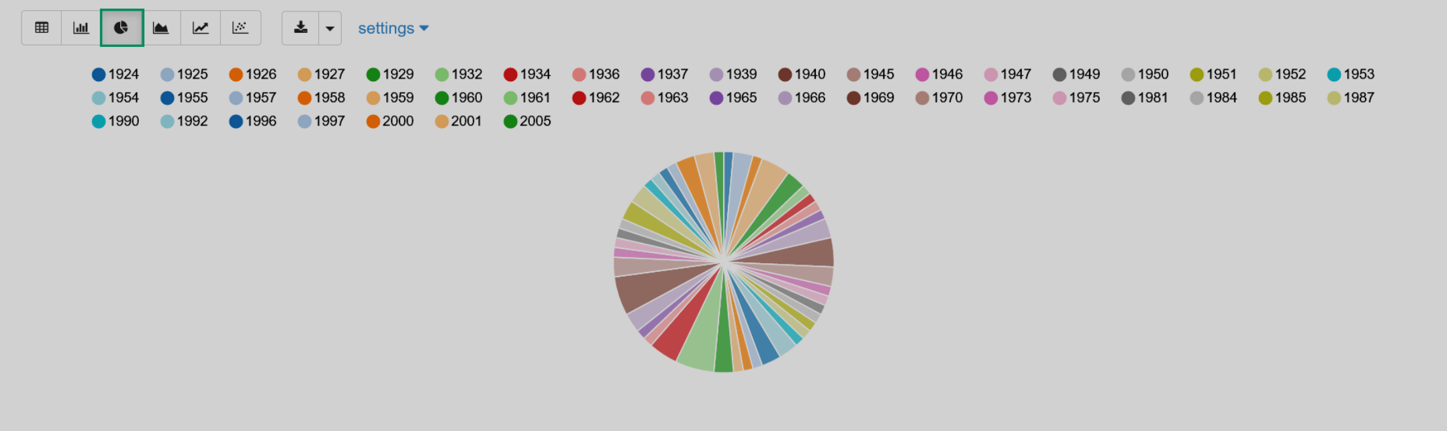
TIP

You can also use other result visualization types. To do this, click the icon that corresponds to the specified view mode, expand the ***settings*** pane, and distribute output fields of your query between available axes. Use drag-and-drop method for this purpose.

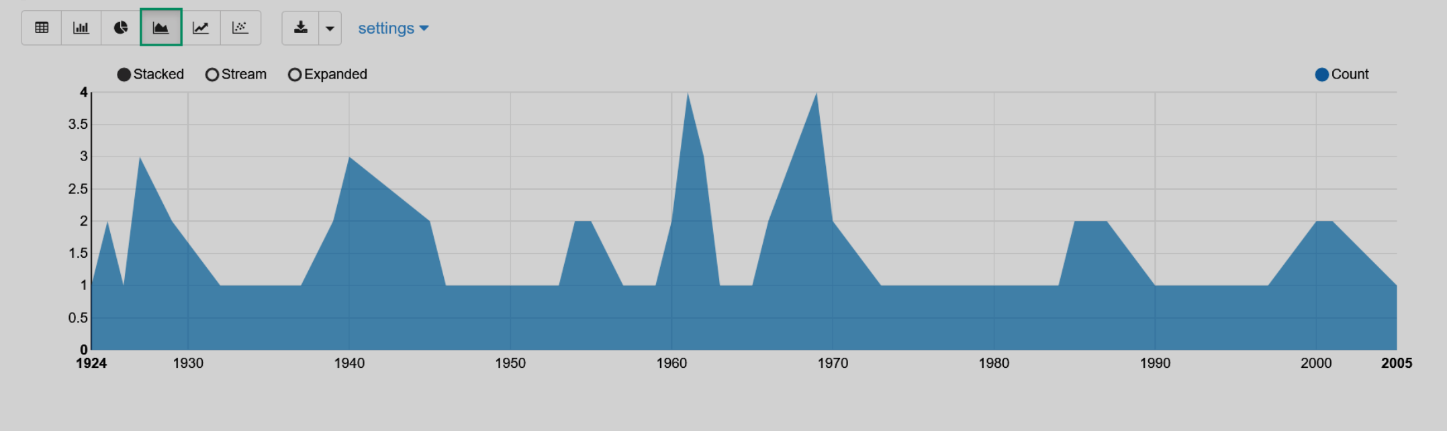

1 Choose the view mode
2 Expand to configure the settings
3 Distribute the fields between axises (drag&drop)
4 View the result
Available Fields
Year
Century
Count
keys
groups
values
Year
Count SUM
xAxis :
Default
Rotate
Hide
Grouped
Stacked
Count
4
3.5
3
2.5
2
1.5
1
0.5
0
1929
1939
1949
1954
1960
1966
1981
1992
2005

Bar chart with results

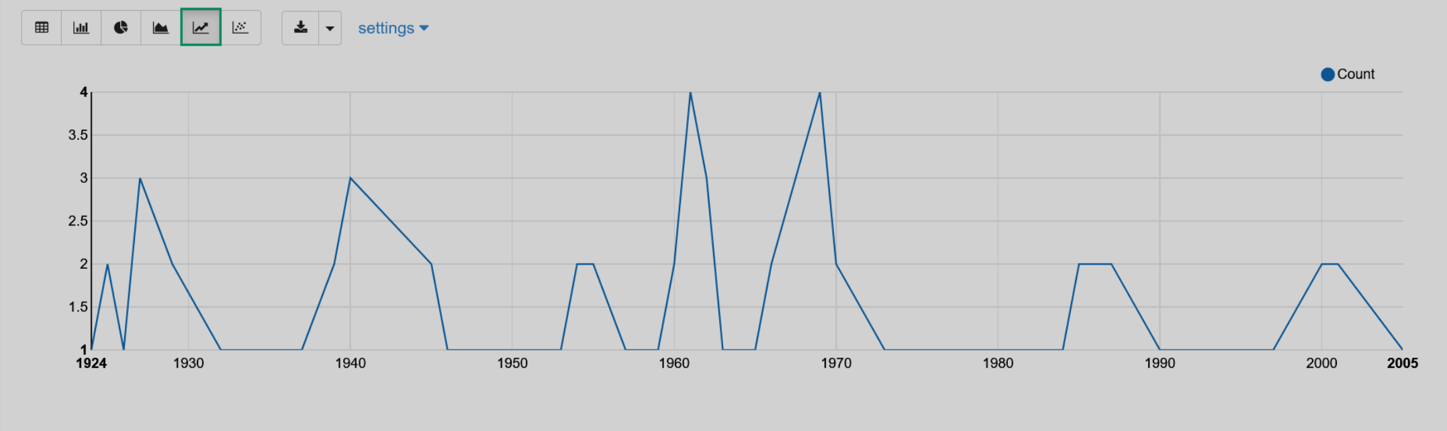
Available result visualization types


1924
1925
1926
1927
1929
1932
1934
1936
1937
1939
1940
1945
1946
1947
1949
1950
1951
1952
1953
1954
1955
1957
1958
1959
1960
1961
1962
1963
1965
1966
1969
1970
1973
1975
1981
1984
1985
1987
1990
1992
1996
1997
2000
2001
2005

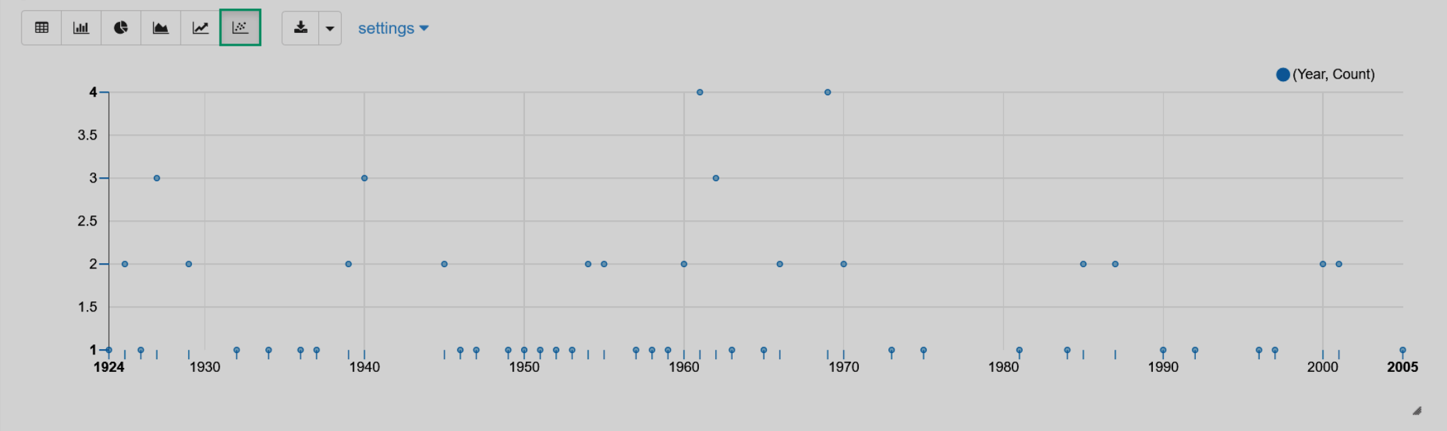
A pie chart


Stacked
Stream
Expanded
Count

An area chart

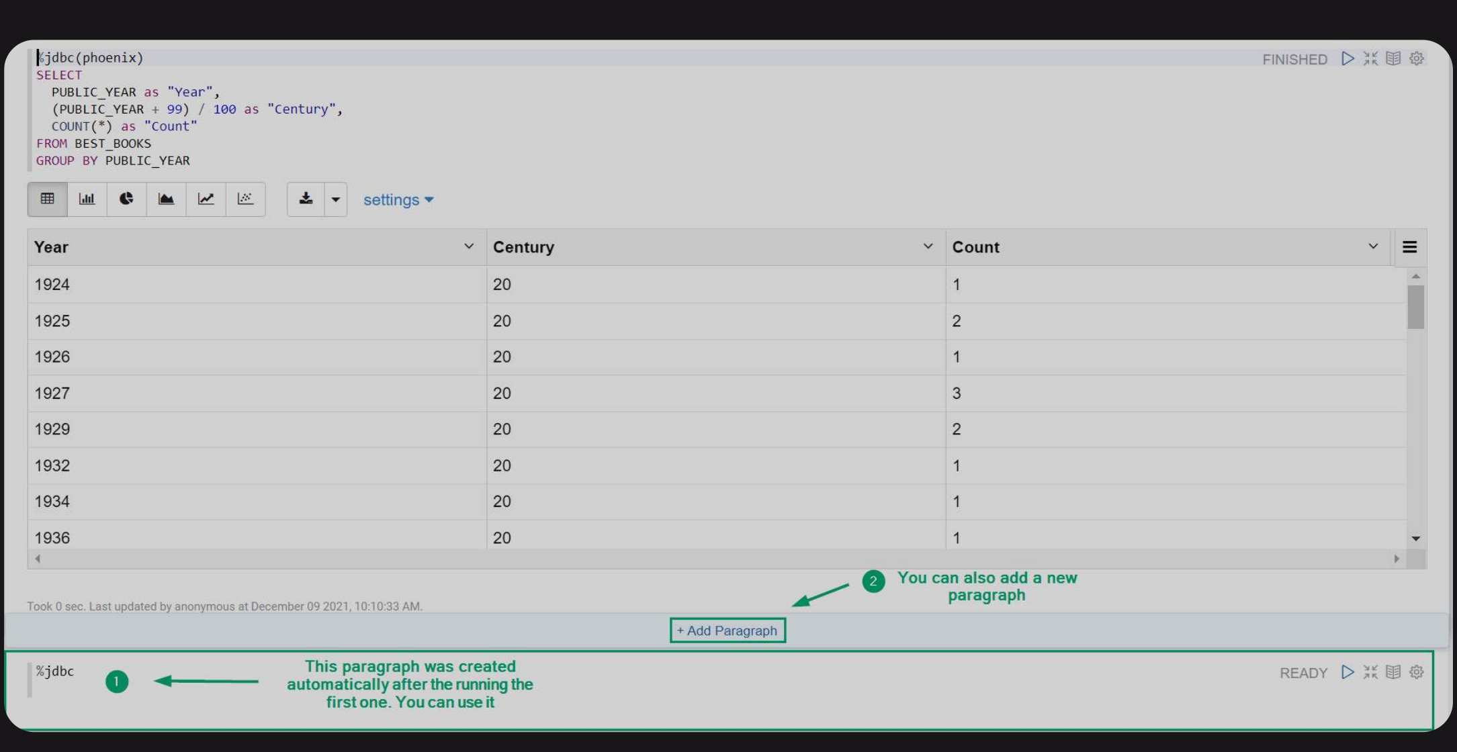


A line chart


(Year, Count)
4
3.5
3
2.5
2
1.5
1
1924
1930
1940
1950
1960
1970
1980
1990
2000
2005

A scatter chart

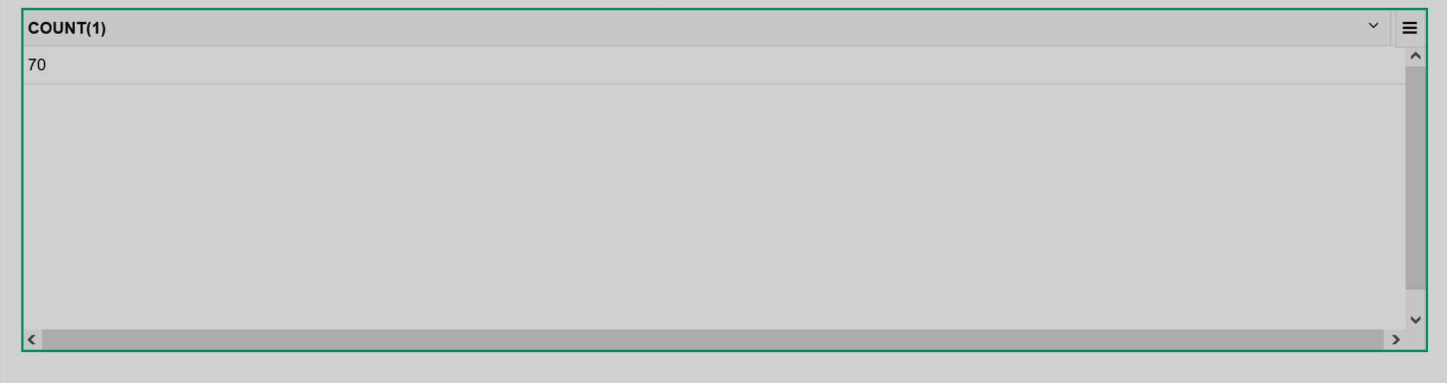
8. Create the second note paragraph to use the %phoenix interpreter. You can use the paragraph that is automatically created after successful running of the first paragraph. Or you can add a new paragraph by clicking the ***Add Paragraph*** button that becomes visible on hover.


2 You can also add a new paragraph
+ Add Paragraph
%jdbc
1 This paragraph was created automatically after the running the first one. You can use it
READY

Adding a new note paragraph

9. Enter the following code in the second paragraph and run it. This query returns the rows count from the `BEST_BOOKS` table.

```
%phoenix
SELECT COUNT(*)
FROM BEST_BOOKS
```


1 Write the interpreter name and code
2 Click to run the code
3 View the result
FINISHED

COUNT(1)
70

Took 3 sec. Last updated by anonymous at December 09 2021, 9:59:55 AM. (outdated)

Running the code

Create a note using Solr

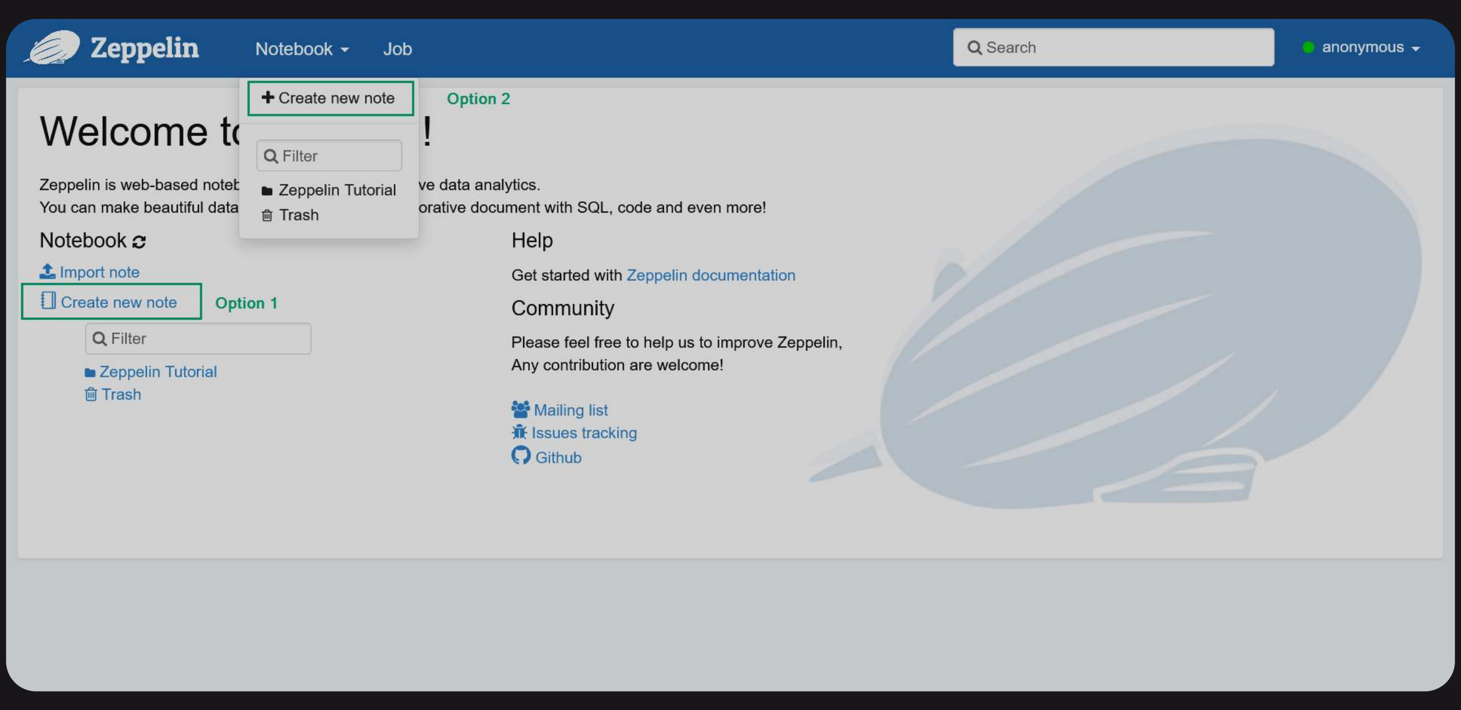
Daria Barysheva

To Table of Contents

NOTE

- Before creating a note, make sure that the Solr interpreter is configured according to the article Install a third-party interpreter.
- The example is based on the Solr collection `people_ages` that contains names and age of 1000 people. The sample dataset is available for download using the direct link.

1. In the Zeppelin web UI, click the ***Create new note*** list item or click the main menu item ***Notebook*** → ***Create new note***.


Zeppelin
Notebook
Job
Search
anonymous
Create new note
Option 2
Filter
Zeppelin Tutorial
Trash
Welcome to Zeppelin!
Notebook
Import note
Create new note
Option 1
Help
Community
Mailing list
Issues tracking
Github

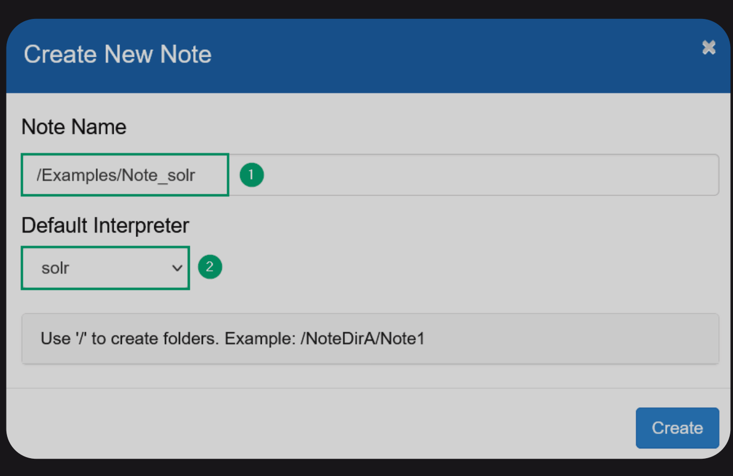
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Get started with Zeppelin documentation

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Adding a new note

2. In the opened dialog, enter the note name and select the default interpreter that will be used in cases when the interpreter type is not specified explicitly. Notice that you can create different folders for your notes. Then, click ***Create***.


Create New Note
1
2
Create

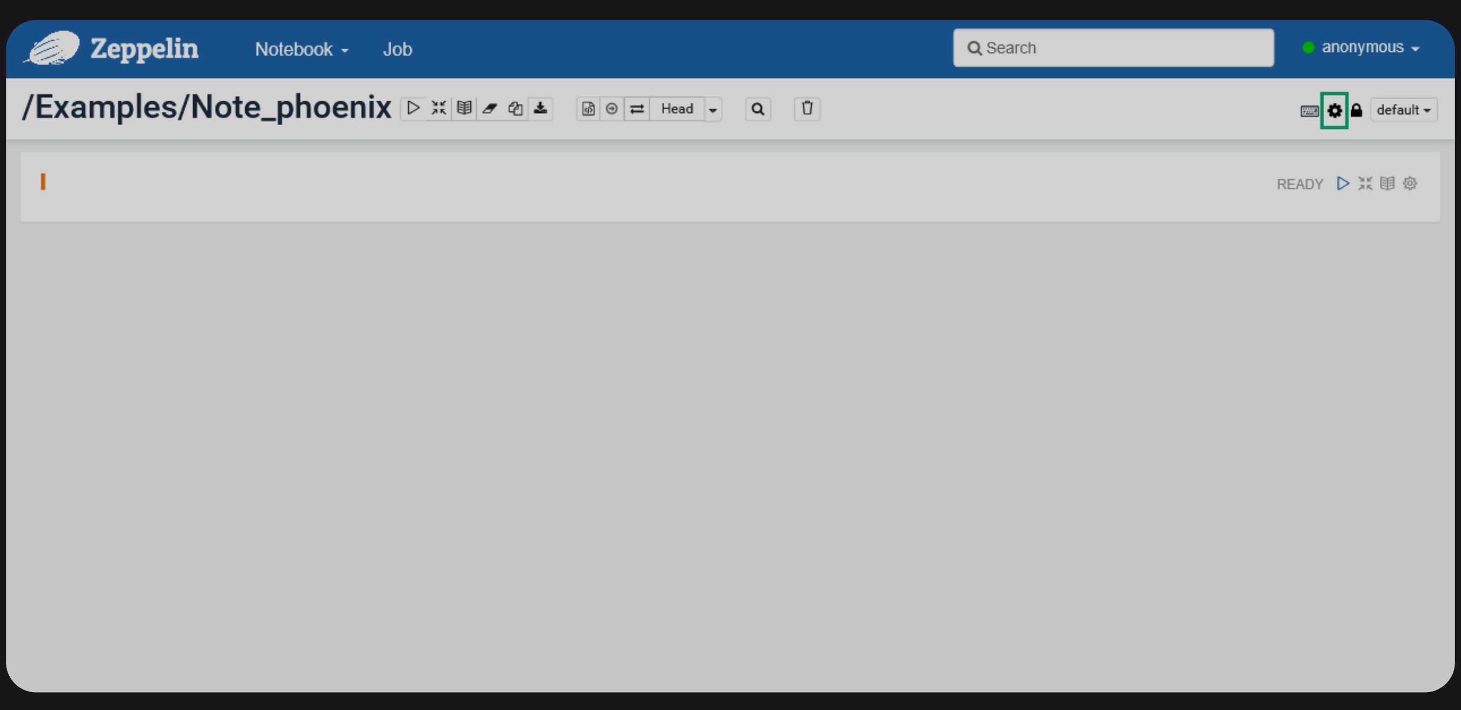
Note Name: /Examples/Note_solr

Default Interpreter: solr

Use '/' to create folders. Example: /NoteDirA/Note1

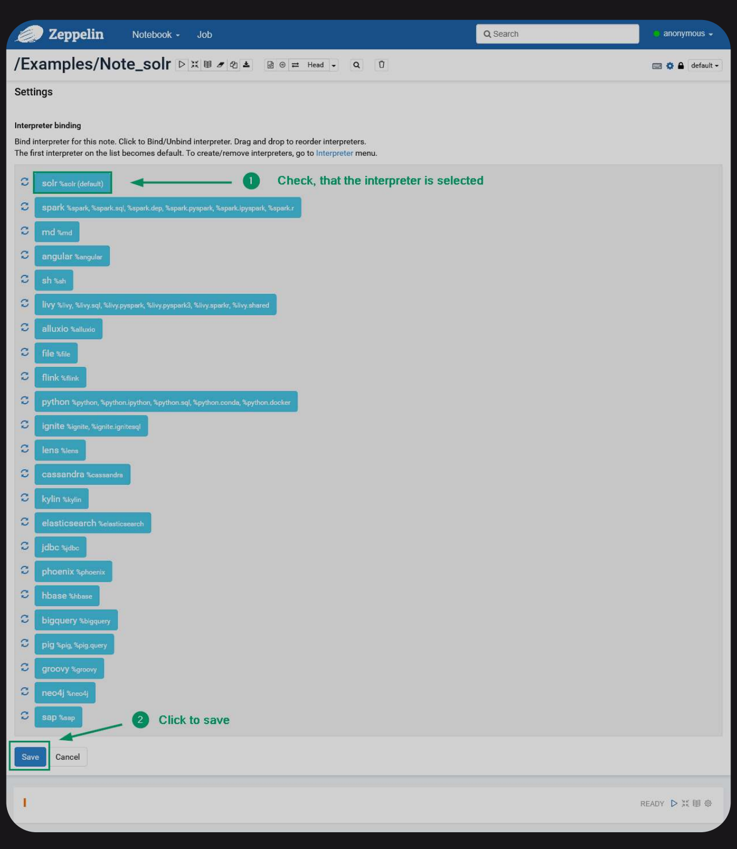
Describing a new note

1. In the opened window, click the ***Interpreter binding*** icon to view the available interpreters.


Zeppelin
Notebook
Job
Search
anonymous
/Examples/Note_phoenix
Head
default
READY

Switch to binding interpreters

2. Verify that the interpreter `%solr` is selected in the opened list; select it otherwise. Then, click ***Save***.


Zeppelin
Notebook
Job
Search
anonymous
/Examples/Note_solr
Head
default
Settings
1
Check, that the interpreter is selected
2
Click to save
Save
Cancel
READY

Interpreter binding

Bind interpreter for this note. Click to Bind/Unbind interpreter. Drag and drop to reorder interpreters.
The first interpreter on the list becomes default. To create/remove interpreters, go to Interpreter menu.

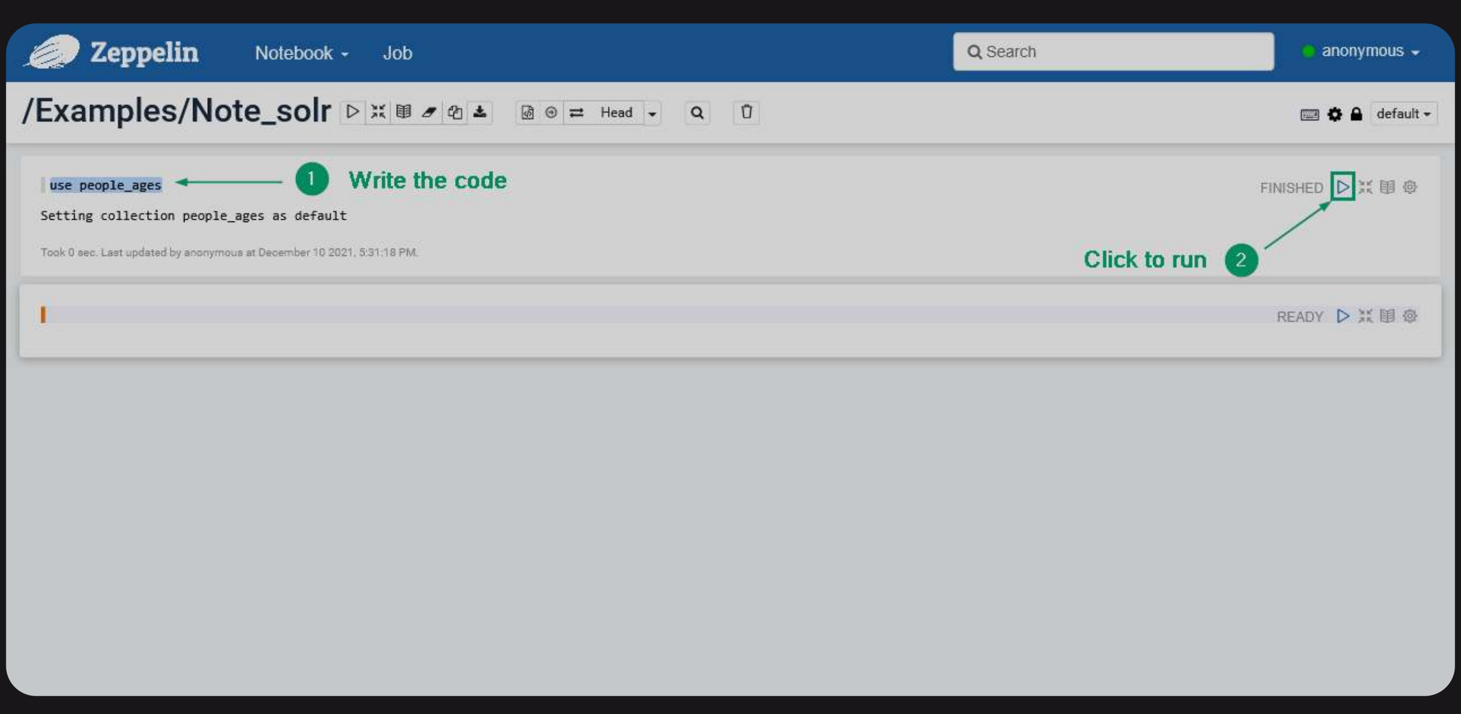
- solr %solr (default)
- spark %spark, %spark.sql, %spark.dep, %spark.pyspark, %spark.ipyspark, %spark.r
- md %md
- angular %angular
- sh %sh
- livy %livy, %livy.sql, %livy.pyspark, %livy.pyspark3, %livy.sparkr, %livy.shared
- alluxio %alluxio
- file %file
- flink %flink
- python %python, %python.ipython, %python.sql, %python.conda, %python.docker
- ignite %ignite, %ignite.ignitesql
- lens %lens
- cassandra %cassandra
- kylin %kylin
- elasticsearch %elasticsearch
- jdbc %jdbc
- phoenix %phoenix
- hbase %hbase
- bigquery %bigquery
- pig %pig, %pig.query
- groovy %groovy
- neo4j %neo4j
- sap %sap

Saving an interpreter

3. In the first paragraph of your note, enter the command that sets the default collection to be selected from Solr (if any other is not defined).

```
use people_ages
```

Then, click ***Run this paragraph*** to execute the code.


Zeppelin
Notebook
Job
Search
anonymous
/Examples/Note_solr
Head
default
1
Write the code
FINISHED
Click to run
2
READY

```
use people_ages
Setting collection people_ages as default
```

Took 0 sec. Last updated by anonymous at December 10 2021, 5:31:18 PM.

Setting the default Solr collection

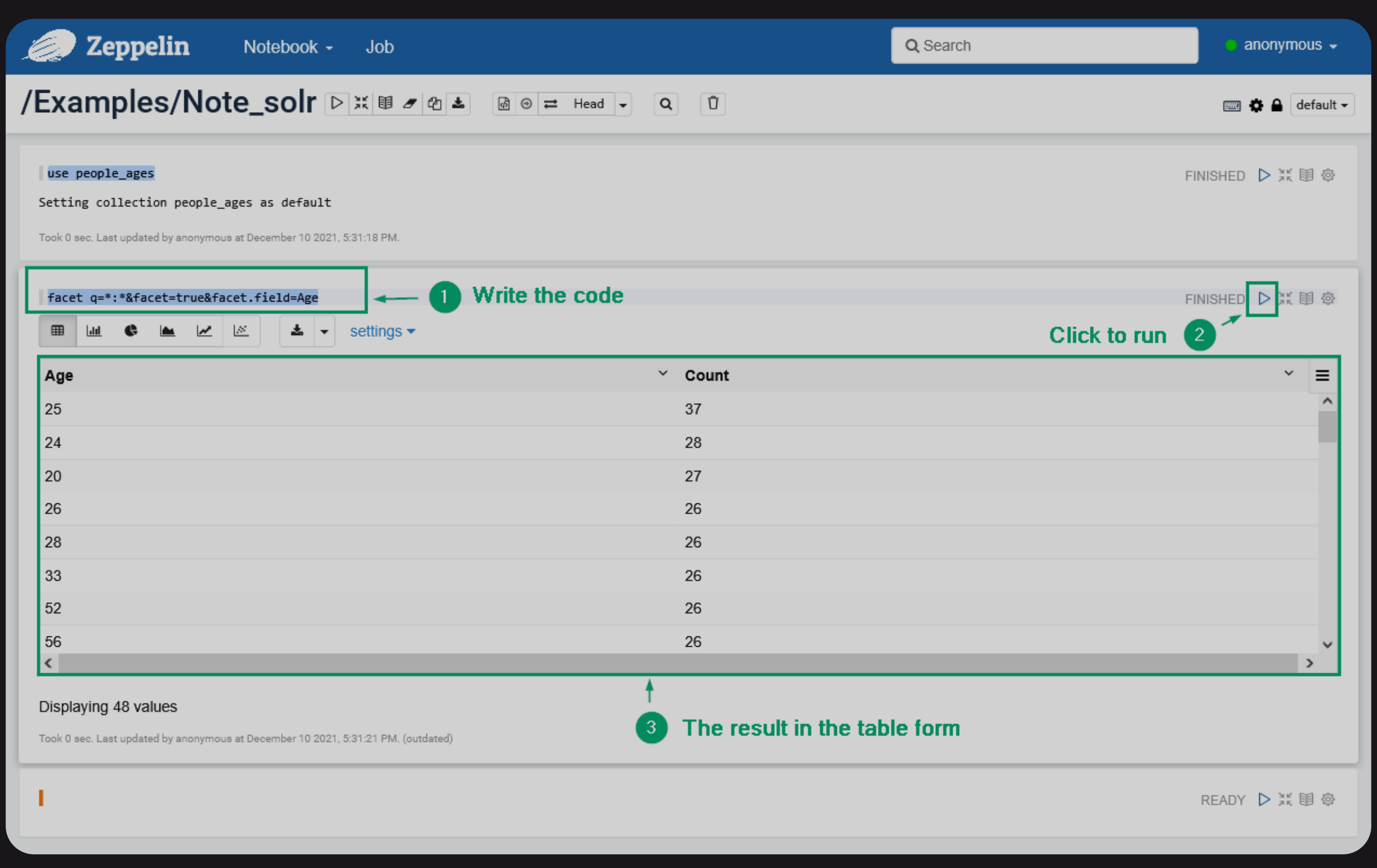
TIP

If you do not specify the interpreter name, the default one is used — `%solr`.

4. In the second note paragraph, enter the following query. The query returns the number of people grouped by age.

```
facet q=*:*&facet=true&facet.field=Age
```

The result is shown below.


Zeppelin
Notebook
Job
Search
anonymous
/Examples/Note_solr
Head
default
FINISHED
1
Write the code
settings
Click to run
2
3
The result in the table form
READY

```
use people_ages
Setting collection people_ages as default
```

Took 0 sec. Last updated by anonymous at December 10 2021, 5:31:18 PM.

```
facet q=*:*&facet=true&facet.field=Age
```

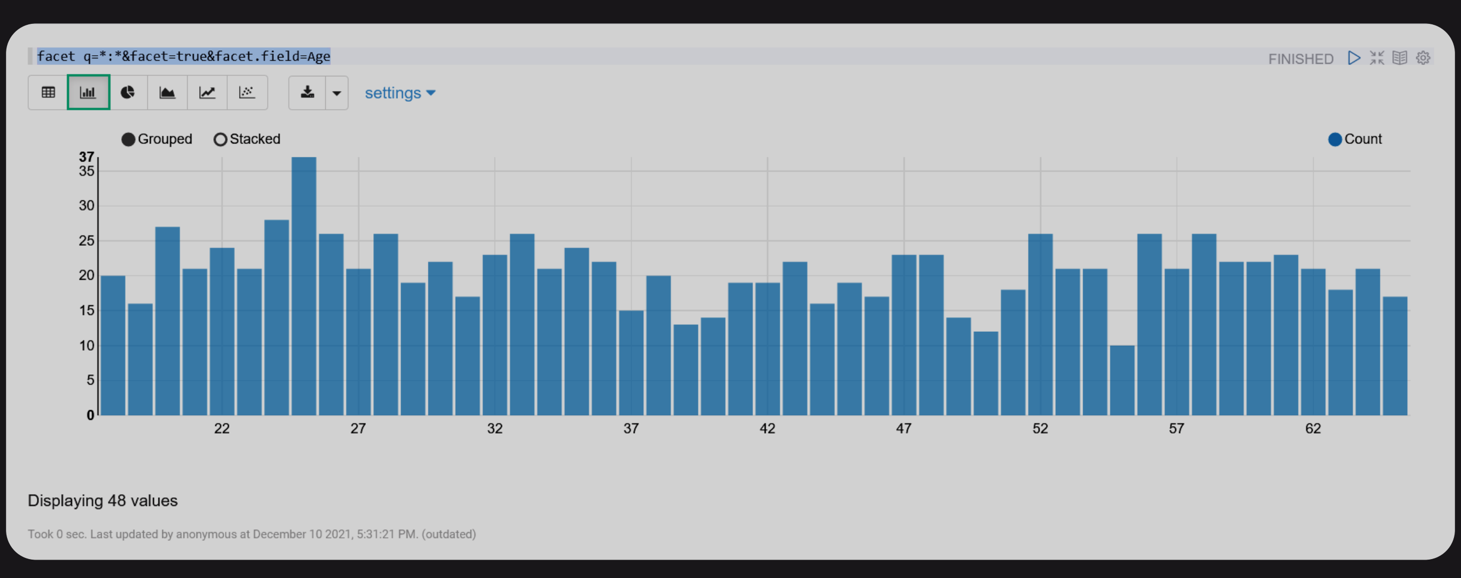
Age	Count
25	37
24	28
20	27
26	26
28	26
33	26
52	26
56	26

Displaying 48 values

Took 0 sec. Last updated by anonymous at December 10 2021, 5:31:21 PM. (outdated)

Result table

5. Click the ***Bar Chart*** icon to switch to a different view mode.



```
facet q=*:*&facet=true&facet.field=Age
```

Displaying 48 values

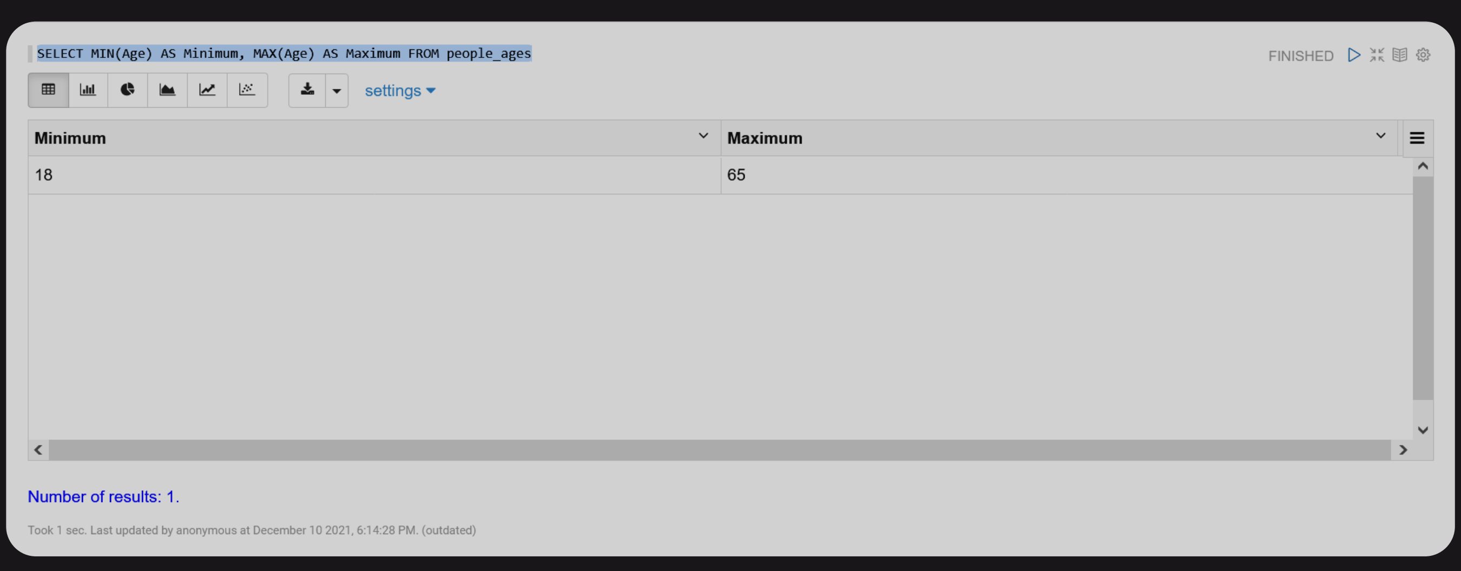
Took 0 sec. Last updated by anonymous at December 10 2021, 5:31:21 PM. (outdated)

A bar chart

6. You can also use SQL statements. Enter the following code in the third paragraph:

```
SELECT MIN(Age) AS Minimum, MAX(Age) AS Maximum FROM people_ages
```

It returns the minimum and the maximum people age values.


FINISHED
settings

```
SELECT MIN(Age) AS Minimum, MAX(Age) AS Maximum FROM people_ages
```

Minimum	Maximum
18	65

Number of results: 1.

Took 1 sec. Last updated by anonymous at December 10 2021, 6:14:28 PM. (outdated)

Running an SQL query

Create a note using Spark

Konstantin Alpashkin

Contents

To Table of Contents

Spark interpreters
Shared context objects
Test data
Create and run a Spark note
  - Scala
  - PySpark
  - SparkSQL

This article shows how to create and run Spark notes in Zeppelin. By default, the Zeppelin service uses the Spark3 service core as a backend for executing Spark notes. If Spark3 and Zeppelin are installed in ADH, Zeppelin runs Spark jobs on a YARN-managed ADH cluster without any configuration.

Spark interpreters

Zeppelin comes with a built-in `spark` interpreter group that includes a set of interpreters for working with specific languages like Scala, Python, and R. These interpreters are listed in the table below.

Interpreter	Class	Description
%spark	SparkInterpreter	Provides a Scala environment for running Spark programs
%spark.sql	SparkSQLInterpreter	Allows running ANSI SQL queries on Spark DataFrame/DataSet objects
%spark.pyspark	PySparkInterpreter	Provides a Python environment
%spark.ipyspark	IPySparkInterpreter	Provides an IPython environment
%spark.r	SparkRInterpreter	Provides a vanilla R environment with SparkR support
%spark.ir	SparkIRInterpreter	Provides an R environment with SparkR support based on Jupyter IRKernel
%spark.shiny	SparkShinyInterpreter	Used to create R shiny apps with SparkR support

Shared context objects

This article assumes that the `spark` interpreter is used in the default shared binding mode, i.e. the interpreter is instantiated globally in the shared mode. This means that Zeppelin creates a single JVM process with a single interpreter group to serve all Spark notes. As a result, variables declared in one note can be directly accessed in other notes, assuming the notes are of the same interpreter type (for example, both notes are `%spark.pyspark`).

For Scala, Python, and R environments, Zeppelin automatically creates `SparkContext`, `SQLContext`, `SparkSession` and `ZeppelinContext` objects and exposes them as `sc`, `sqlContext`, `spark`, and `z` variables respectively. Being shared variables, the same instance of these objects can be accessed by different Spark notes.

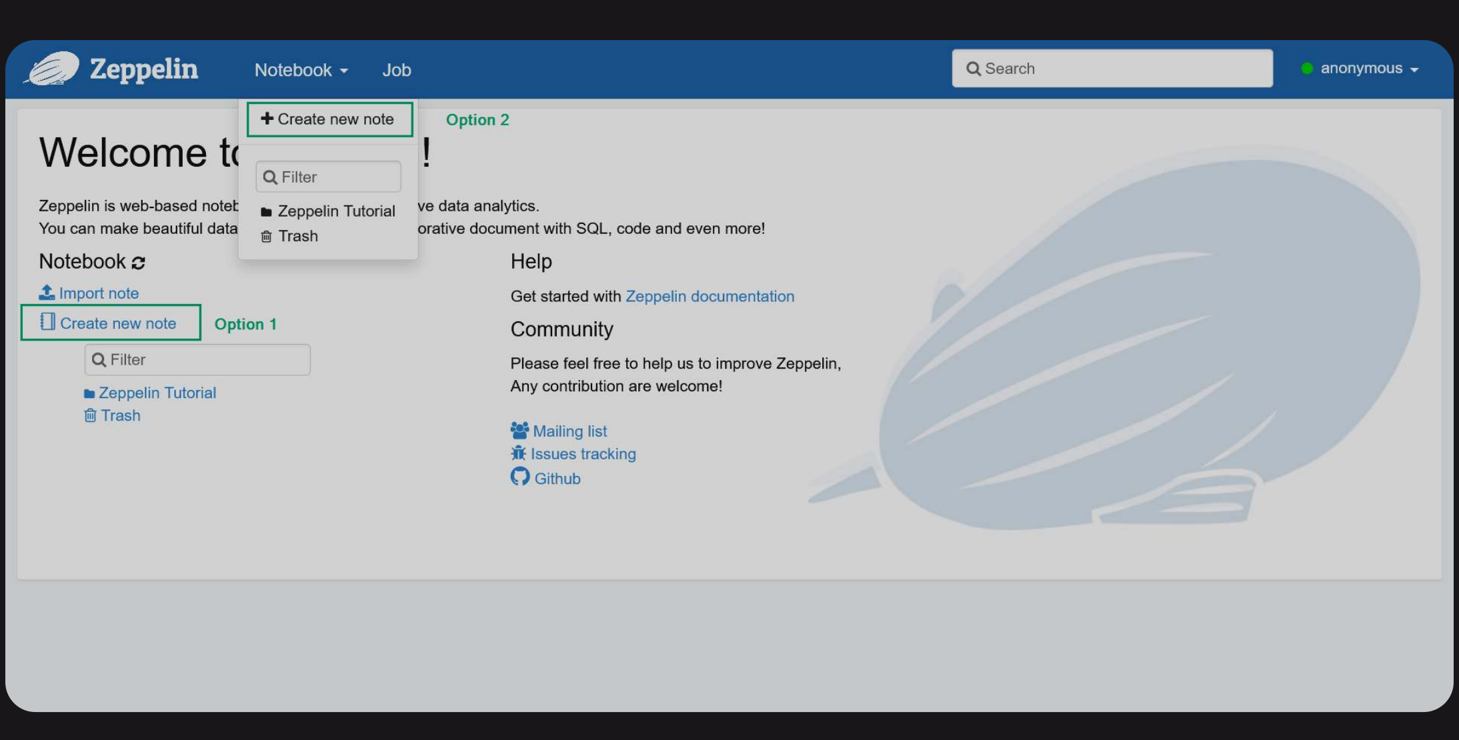
Test data

In this article, a dummy dataset people.csv is used for testing Spark examples. Before running the examples, the dataset should be loaded to HDFS. For this, use the following commands:

```
$ hdfs dfs -mkdir /user/zeppelin/test_data
$ hdfs dfs -chown zeppelin:hadoop /user/zeppelin/test_data
$ hdfs dfs -put people.csv /user/zeppelin/test_data/people.csv
```

Create and run a Spark note

1. In the Zeppelin web UI, click the ***Create new note*** list item or click the main menu item ***Notebook*** → ***Create new note***.

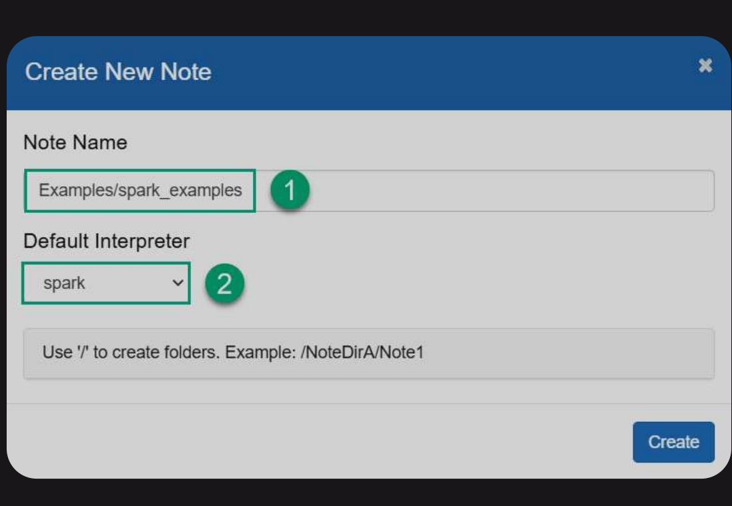


Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive and collaborative document with SQL, code and even more!

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

Adding a new note

2. In the opened dialog, enter the note name and select the default interpreter that will be used in cases when the interpreter type is not specified explicitly. Notice that you can create different folders for your notes. Then, click ***Create***.



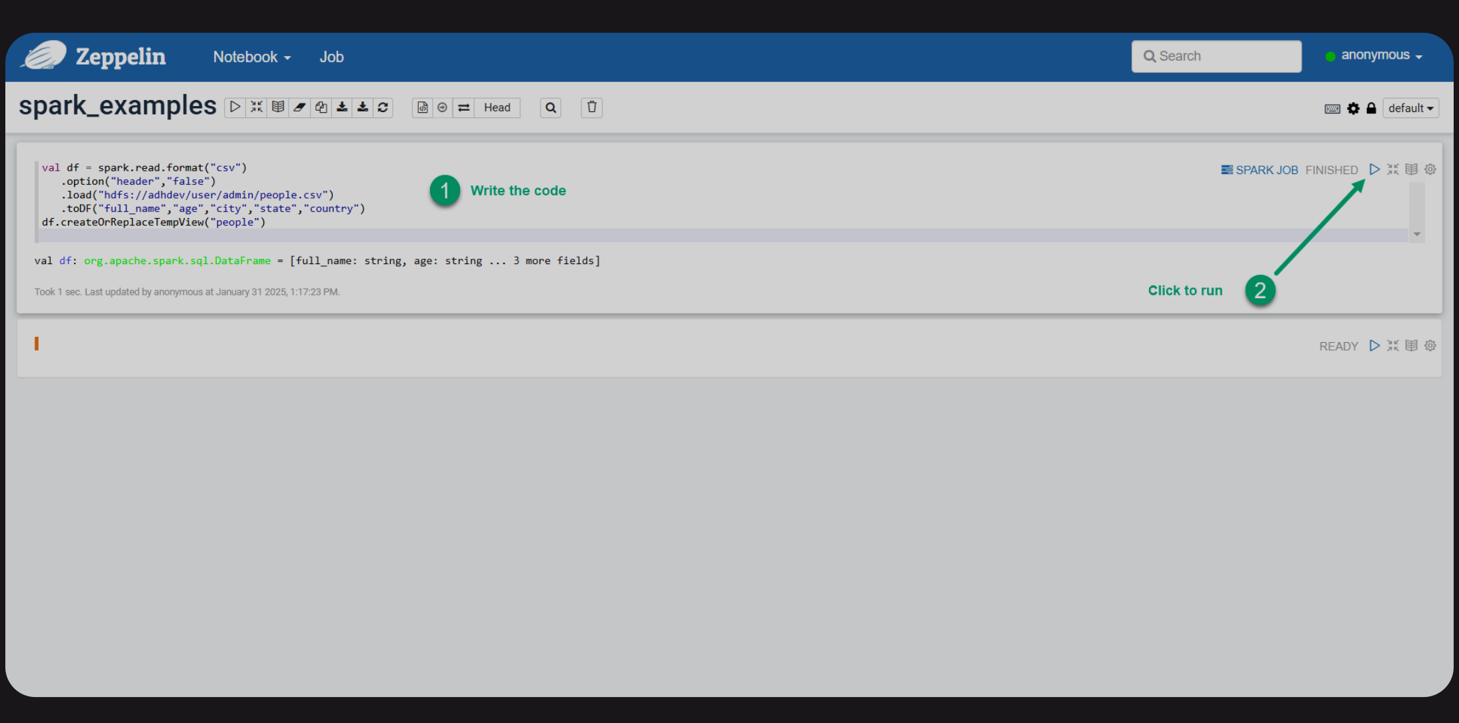
Note Name: Examples/spark_examples ①

Default Interpreter: spark ②

Use '/' to create folders. Example: /NoteDirA/Note1

Create a new note

3. Enter the Spark code in the paragraphs of your note and run the paragraph.



```
val df = spark.read.format("csv")
   .option("header","false")
   .load("hdfs://adhdev/user/admin/people.csv")
   .toDF("full_name","age","city","state","country")
df.createOrReplaceTempView("people")
```

```
val df: org.apache.spark.sql.DataFrame = [full_name: string, age: string ... 3 more fields]
```

Took 1 sec. Last updated by anonymous at January 31 2025, 1:17:23 PM.

Run Spark code in Zeppelin

The examples of Spark code in Scala, PySpark, and SparkSQL are provided below. For more Spark examples, explore the Spark tutorial notes available in Zeppelin (***Notebook*** → ***Spark Tutorial***).

Scala

To run a Spark note in Scala, specify the `%spark` interpreter on the first line of the paragraph. Although technically the `%spark` definition can be omitted (since it is the default interpreter in a group), it is still recommended to declare interpreters explicitly for each paragraph of a note.

```
%spark ①
val df_people = spark.read.format("csv") ②
   .option("header","false")
   .load("hdfs://adhdev/user/zeppelin/test_data/people.csv")
   .toDF("full_name","age","city","state","country")
df_people.printSchema()

case class SW_movie(id:Int, name:String, episode:Int, year:Int)

val df_sw_movies = spark.createDataFrame(Seq ③
    (
        SW_movie(1, "Revenge of the Sith", 3, 2005),
        SW_movie(2, "Attack of the Clones", 2, 2002),
        SW_movie(3, "The Phantom Menace", 1, 1999),
        SW_movie(4, "A New Hope", 4, 1977),
        SW_movie(5, "Return of the Jedi", 6, 1983),
        SW_movie(6, "The Empire Strikes Back", 5, 1980)
    )
)
df_sw_movies.printSchema()
val df_sw_movies_comments = df_sw_movies.withColumn("rank", lit(0.0)) ④
df_sw_movies_comments.printSchema()
df_sw_movies_comments.drop("rank") ⑤

val sorted_by_year = rdd.map(mv => ⑥
    (mv._1,mv._2,mv._3)
  ).sortBy(_._3).toDF("name", "episode", "year")
sorted_by_year.show()

sorted_by_year.write ⑦
  .mode("overwrite")
  .json("/user/zeppelin/test_data/star_wars_movies")
```

① Specifies an explicit interpreter for running Spark code in Scala.

② Creates a DataFrame by reading the test dataset file in HDFS.

③ Creates a DataFrame from a tuple of Case class objects.

④ Adds a static column to the DataFrame.

⑤ Creates a DataFrame with data sorted by a custom column.

⑥ Drops a DataFrame column.

⑦ Writes the DataFrame to HDFS as JSON.

PySpark

To run PySpark notes, declare the `%spark.pyspark` interpreter on the first line of the paragraph.

```
%spark.pyspark ①
df = spark.createDataFrame([ ②
    (1, "Ivan", 20, "Russia"),
    (2, "Yao", 23, "China"),
    (3, "Ann", 28, "Russia"),
    (4, "John", 20, "USA")
]).toDF("id", "name", "age", "country")
df.printSchema()
df.show()

from pyspark.sql.types import StructType
peopleSchema = StructType() \
    .add("full_name", "string") \
    .add("age", "integer") \
    .add("city", "string") \
    .add("state", "string") \
    .add("country", "string")
df_people = spark.read.csv("hdfs://adhdev/user/zeppelin/test_data/people.csv", schema=peopleSchema) ③
df_people.printSchema();

df_name_city = df_people.select("full_name", "city") ④
df_name_city.show(10)

import pyspark.sql.functions as f
df_name_uppered = df_people.withColumn("full_name_upper", f.upper(df_people["full_name"])) ⑤
df_name_uppered.printSchema()

df_people_stateless = df_people.drop("state"); ⑥
df_people_stateless.printSchema();

df = df_people.filter( ⑦
    (df_people["age"] >= 20) & (df_people["city"] == "Houston")
)
df.show()

from pyspark.sql.functions import udf
from pyspark.sql.types import BooleanType
@udf(returnType=BooleanType()) ⑧
def udf_older_than(e):
    if e > 60:
        return True;
    else:
        return False
df_people_older_than60 = df_people.filter(udf_older_than(df_people["age"]))
df_people_older_than60.show()
```

① Specifies an explicit interpreter for running PySpark code.

② Creates a DataFrame from a list of Python tuples.

③ Creates a DataFrame from the sample dataset in HDFS.

④ Creates a DataFrame by selecting a subset of columns.

⑤ Adds a new column to the DataFrame with string values upper-cased using `pyspark.sql.functions`.

⑥ Drops a column from the DataFrame.

⑦ Applies a combination of filter conditions.

⑧ Defines a UDF, which is later used for filtering.

SparkSQL

In Scala/PySpark environments, you can run SQL queries against a DataFrame by using `spark.sql()`. For example:

```
%spark.pyspark ①

from pyspark.sql.types import StructType

peopleSchema = StructType() \
    .add("full_name", "string") \
    .add("age", "integer") \
    .add("city", "string") \
    .add("state", "string") \
    .add("country", "string")
df_people = spark.read.csv("hdfs://adhdev/user/zeppelin/test_data/people.csv", schema=peopleSchema)
df_people.createOrReplaceTempView("people") ②

df_people30 = spark.sql("select full_name, age from people where age=30") ③
df_people30.show()
```

① Defines the interpreter for handling PySpark code.

② Registers a temporary view based on a DataFrame. This is a mandatory step for using SparkSQL queries.

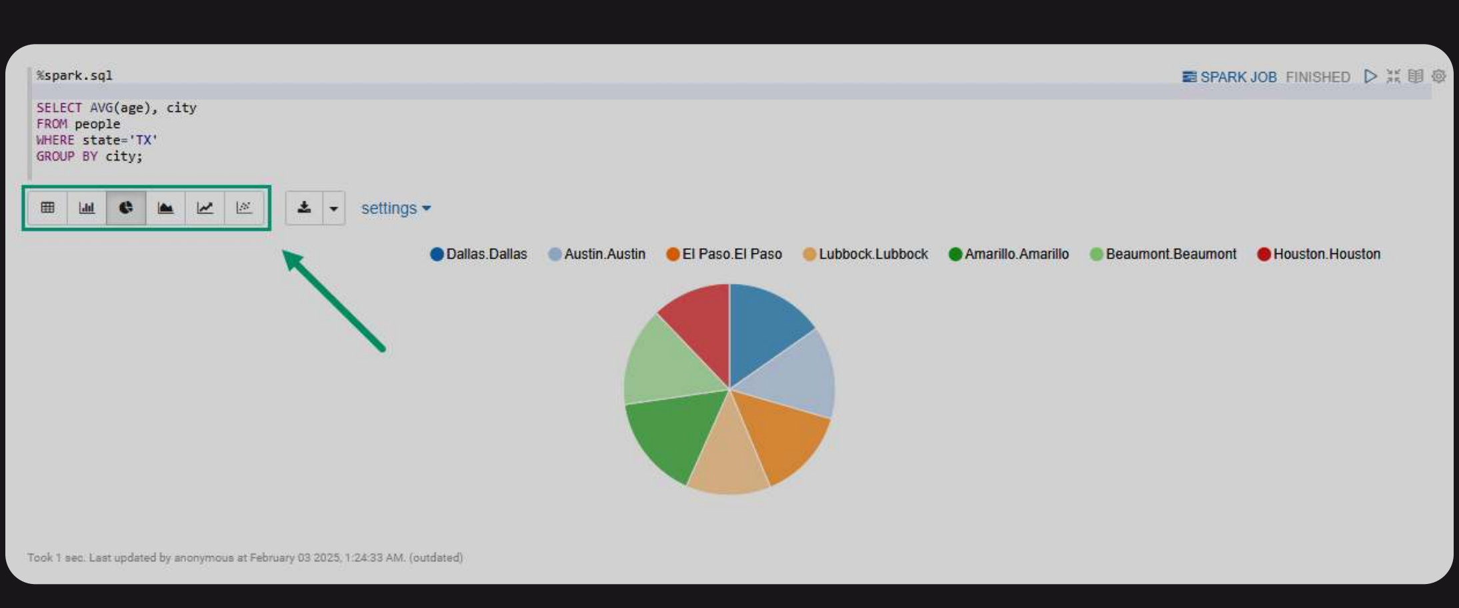
③ Runs SQL on the temporary view. The query results are stored in a new DataFrame.

By specifying the `%spark.sql` interpreter on the first line of the paragraph, you can visualize the queries to a Spark DataFrame using built-in widgets (pie charts, graphs, tables, etc.). For example:

```
%spark.sql

SELECT AVG(age), city
FROM people ①
WHERE state='TX'
GROUP BY city;
```

① Before running the query, the `people` object must be registered as a temporary view.

When you run a `%spark.sql` paragraph, the result set is displayed in the form of a table. You can switch between several widgets to visualize the results on a pie chart, line graph, etc., as shown in the image.



```
%spark.sql
SELECT AVG(age), city
FROM people
WHERE state='TX'
GROUP BY city;
```

Took 1 sec. Last updated by anonymous at February 03 2025, 1:24:33 AM. (outdated)

Visualize query results

Create a note using Python

Konstantin Alpashkin

Contents

To Table of Contents

- Specify a Python interpreter
- Zeppelin interpreters for Python
- Shared variables
- Create and run a Python note
  - Python (%python)
  - IPython (%python.ipython)
  - SQL over Pandas DataFrames (%python.sql)

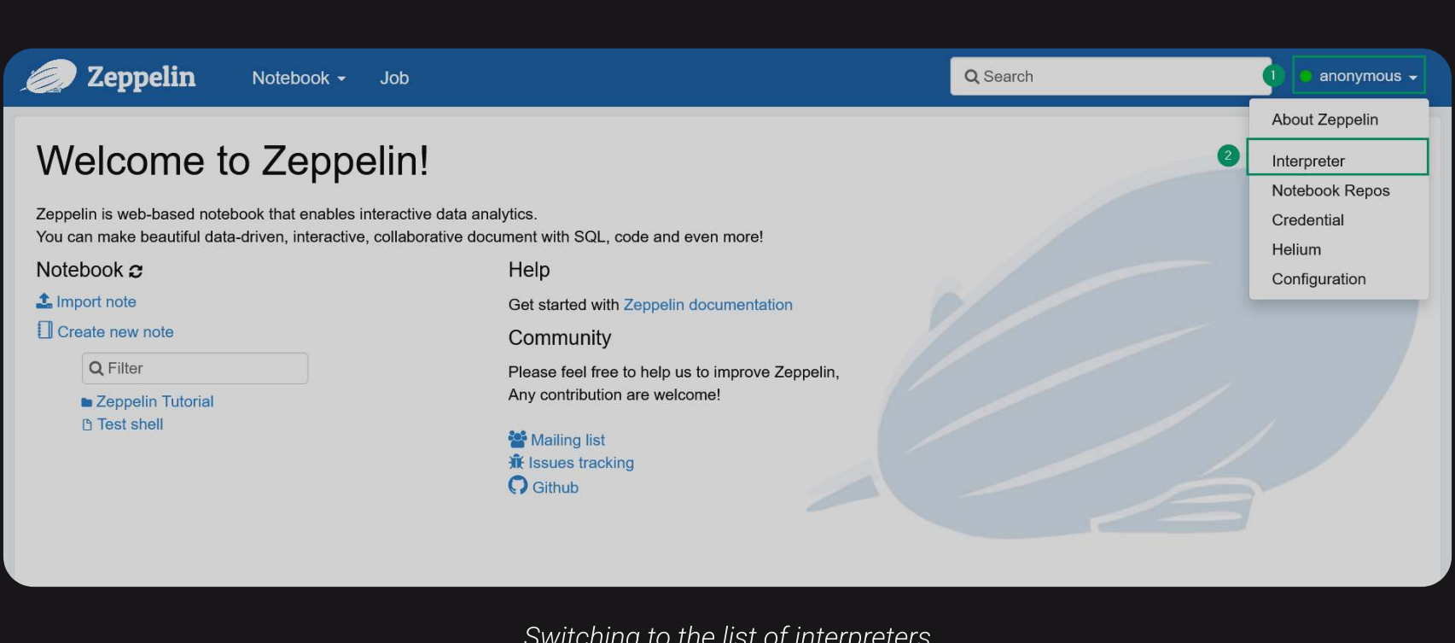
This article shows how to create and run Python notes in Zeppelin. By default, to execute a Python note, the Zeppelin service uses the Python interpreter available in `$PATH`. You can change the Python instance to be used by Zeppelin by using the `zeppelin.interpreter` property. For more information, see Specify a Python interpreter.

Specify a Python interpreter

Various operating systems come with different Python versions available in `$PATH`. An OS may be bundled with Python 2.x, Python 3, or may not include Python at all, so you might want to configure Zeppelin to use a custom Python version.

For this, follow the steps:

1. Select the ***Interpreter*** item in the account menu placed in the top right corner of the Zeppelin home page.


Zeppelin
Notebook
Job
Search
1
anonymous
About Zeppelin
2
Interpreter
Notebook Repos
Credential
Helium
Configuration
Welcome to Zeppelin!
Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!
Notebook
Import note
Create new note
Filter
Zeppelin Tutorial
Test shell
Help
Get started with Zeppelin documentation
Community
Please feel free to help us to improve Zeppelin,
Any contribution are welcome!
Mailing list
Issues tracking
Github

Switching to the list of interpreters

2. Find the `python` interpreter and click ***Edit***.
3. Set the `zeppelin.python` property to point to the desired Python binary.

> **TIP**
>
> The Python interpreter distributed with ADH can be found in */opt/<python_env>*.

4. Save and restart the interpreter configuration.
5. Run a test note to ensure that Zeppelin uses the new Python instance:

```
%python

import sys
print(f"""Current note has been executed using:
    Python location: {sys.executable}
    Python version: {sys.version_info[0]}
    """)
```

Sample output:

```
Current note has been executed using:
    Python location: /opt/python3.10/bin/python3
    Python version: 3
```

Zeppelin interpreters for Python

Zeppelin comes with a built-in `python` interpreter group that includes a set of interpreters listed in the table below. All the listed interpreters support Python 2 and Python 3.

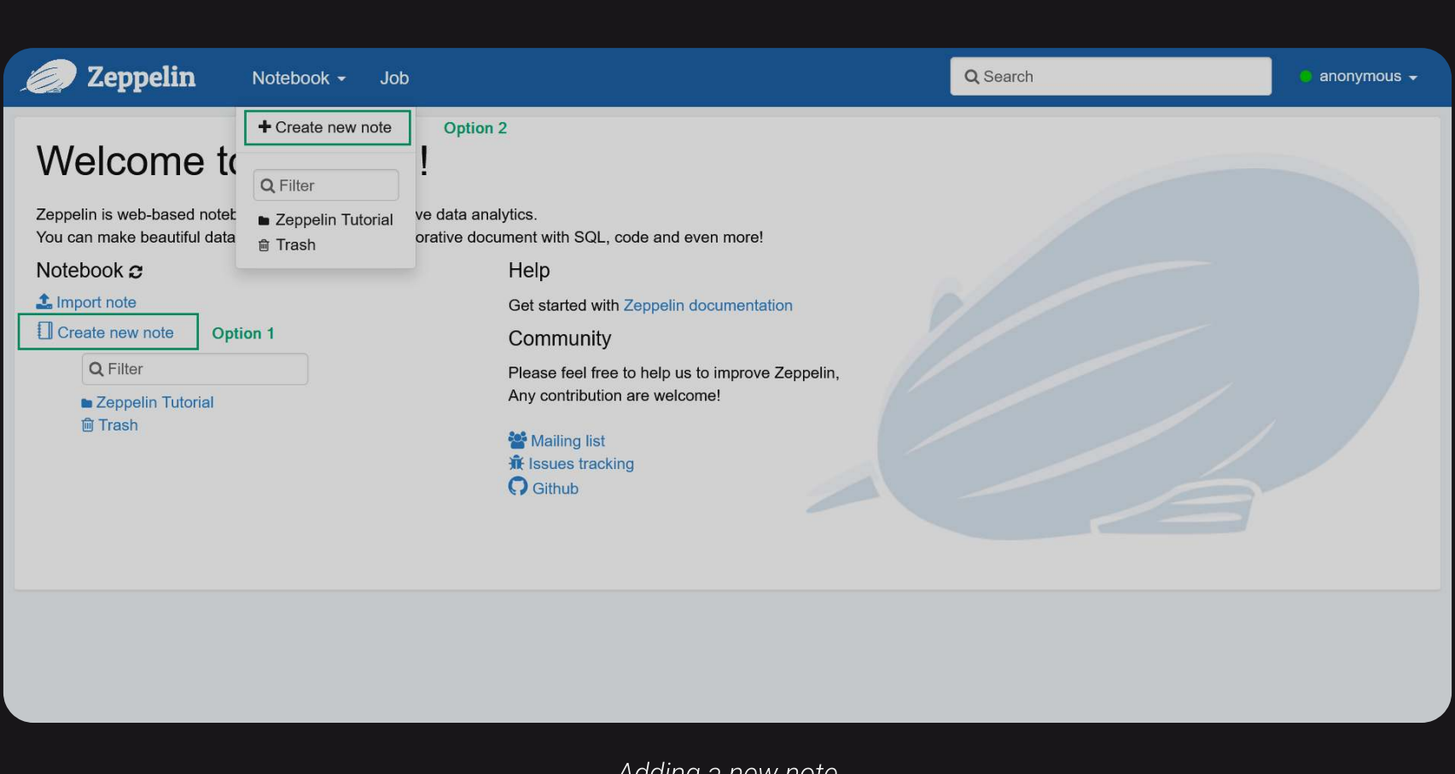
Interpreter	Class	Description
%python	PythonInterpreter	A vanilla Python interpreter with basic dependencies. Requires only Python environment installed. Uses IPython features if the prerequisites are met
%python.ipython	IPythonInterpreter	Provides a fancier Python environment based on IPython. It is the recommended interpreter for using Python in Zeppelin
%python.sql	PythonInterpreterPandasSql	Provides SQL capabilities for querying Pandas DataFrames via `pandasql`

Shared variables

This article assumes that Zeppelin's `python` interpreter is used in the default shared binding mode, i.e. the interpreter is instantiated globally in the shared mode. This means that Zeppelin creates a single JVM process with a single interpreter group to serve all Python notes. As a result, variables declared in one note can be directly accessed in other notes. Variable sharing also works between `%python` and `%python.ipython` notes.

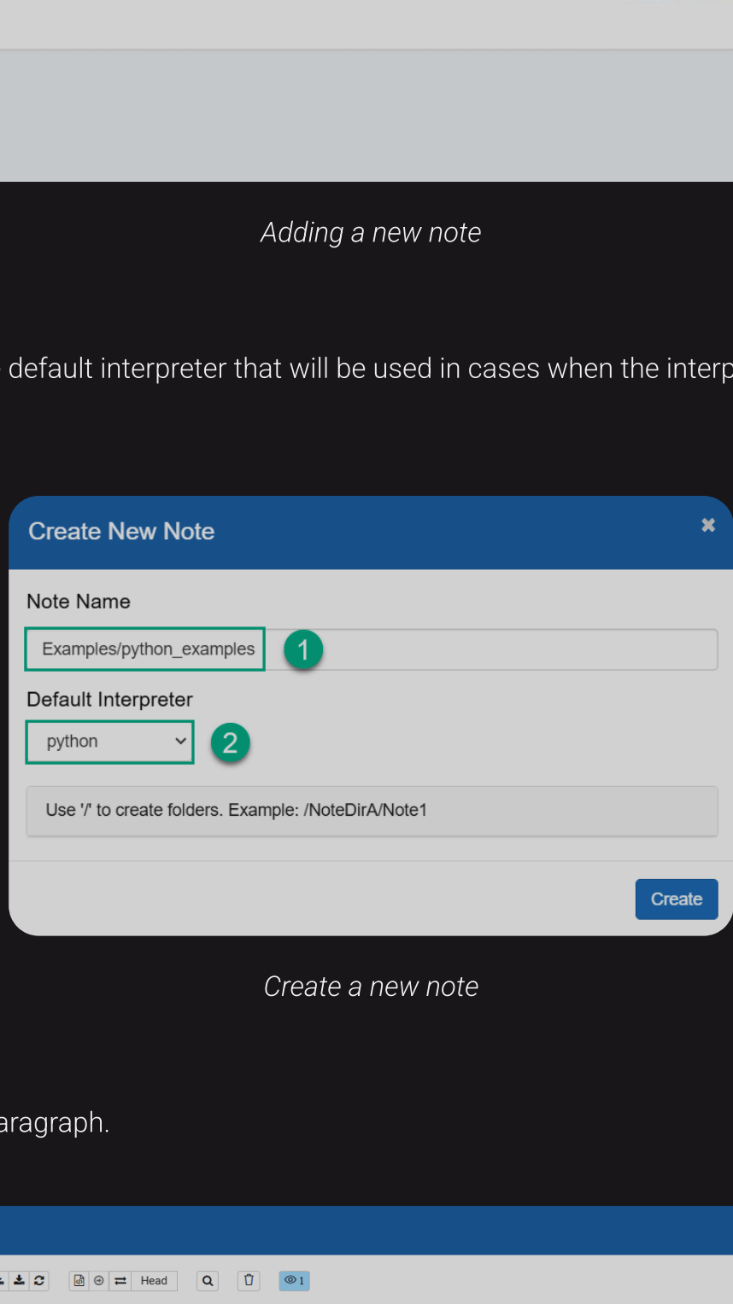
Create and run a Python note

1. In the Zeppelin web UI, click the ***Create new note*** list item or click the main menu item ***Notebook*** → ***Create new note***.



Adding a new note

2. In the opened dialog, enter the note name and select the default interpreter that will be used in cases when the interpreter type is not specified explicitly. Notice that you can create different folders for your notes. Then, click ***Create***.



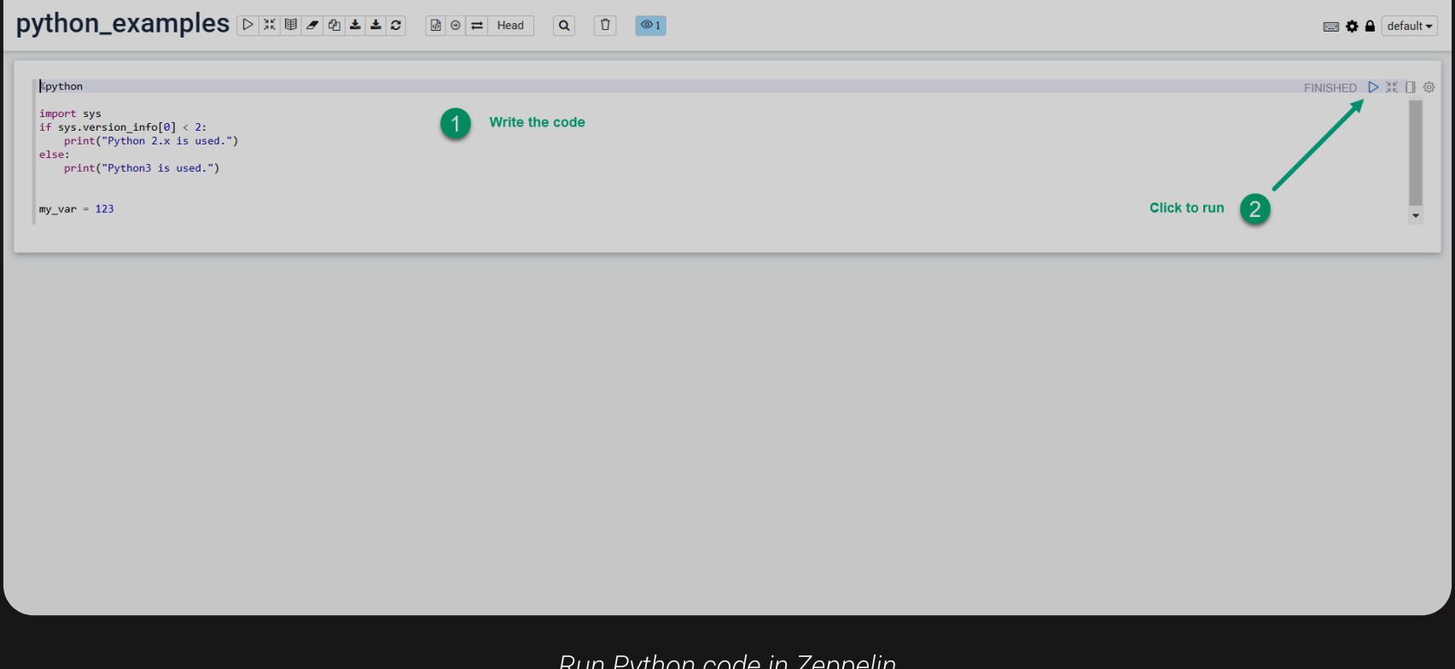
Note Name: Examples/python_examples

Default Interpreter: python

Use '/' to create folders. Example: /NoteDirA/Note1

Create a new note

3. Enter the Python code in a new paragraph and run the paragraph.



```
%python

import sys
if sys.version_info[0] < 2:
    print("Python 2.x is used.")
else:
    print("Python3 is used.")

my_var = 123
```

Run Python code in Zeppelin

The examples of Python notes are provided below. All the notes are available in the .zpln archive, which you can import to Zeppelin (***Zeppelin*** → ***Import Note***). For more examples, explore the Zeppelin tutorial (***Notebook*** → ***Python Tutorial***).

Python (%python)

To use the vanilla Python interpreter for running a Zeppelin note, specify `%python` on the first line of the paragraph. Use `%python` if you need a simple Python execution environment with no extra dependencies.

The following sample note shows basic Pandas DataFrame usage with Zeppelin built-in visualization.

```
%python ①
import pandas as pd ②

txn_data = {'acc_id': [1001,1003,1002,1001,1001,1002,1003,1001,1003],
        'txn_value': [25.00, 30.00, 100.00, 250.00, 98.50, 23.50, 45.00, 60.00, 75.15],
        'txn_date': ['2025-01-07', '2025-01-03', '2025-01-09', '2025-01-02', '2025-01-01', '2025-01-04', '2025-01-08', '2025-01-11', '2025-01-03']}
df = pd.DataFrame(txn_data) ③
z.show(df) ④
```

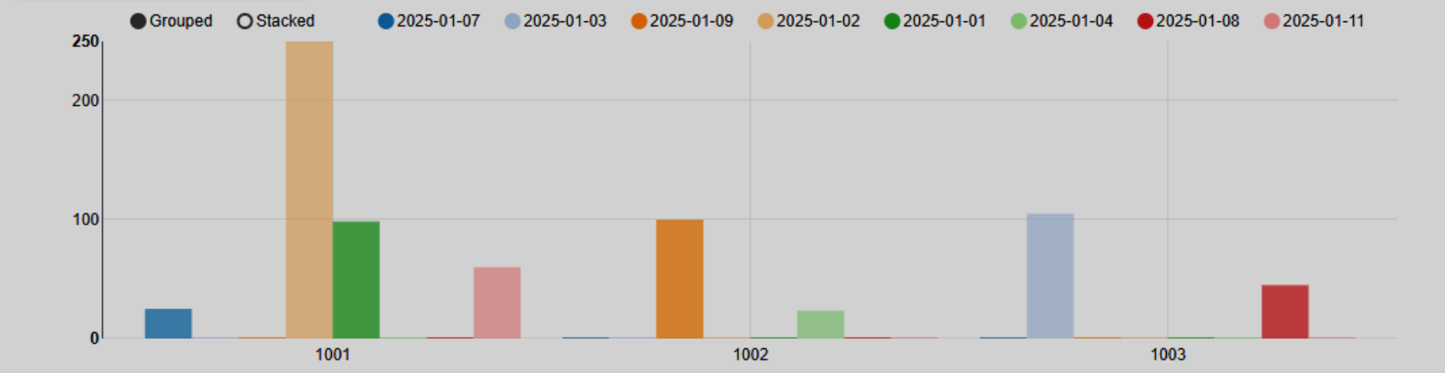
① Specifies the vanilla Python interpreter.

② Before running the note, imported modules must be installed via `pip install <module_name>`.

③ Creates a Pandas DataFrame.

④ Renders the DataFrame using built-in Zeppelin widgets. For more visualization examples, see IPython examples below.

The note execution results are shown below.



Built-in Zeppelin visualization

IPython (%python.ipython)

IPython (`%python.ipython`) is a Jupyter-like, IPython-based interpreter that provides a rich environment for interactive computing, such as:

- Better tab completion, syntax and output highlighting.
- Built-in magic functions like `%timeit`, `%matplotlib inline`, etc.
- Advanced visualization via third-party libraries like Seaborn, Plotly, Bokeh, etc.

To use the `%python.ipython` interpreter, install the following modules to the Python instance used by Zeppelin:

- `jupyter-client`
- `ipython`
- `ipykernel`
- `grpcio`
- `protobuf`

> **NOTE**
>
> Depending on the Python version used by Zeppelin, one may observe conflicts while installing the latest `protobuf` package. A workaround for this is downgrading or installing an older `protobuf` version. For example, as of Python 3.10 used in this article, `protobuf` 3.19.6 has been verified to work properly.

To use IPython features, specify `%python.ipython` on the first line of the paragraph. Example:

```
%python.ipython ①

import pandas as pd

data = {
    'full_name': ['Sarah Connor',
        'Kirk Hammett',
        'Ivan Ivanov'],
    'age': [25, 30, 35],
    'dep': ['it', 'sales', 'hr']}
df = pd.DataFrame(data) ②
print(" Describe DF:".center(50, "*"))
print(df.describe())
print(" DF types: ".center(50, "*"))
print(df.dtypes)
print(" Filter DF: ".center(50, "*"))
df1 = df[df['age'] > 30]
print(df1)
print(" Add column to DF: ".center(50, "*"))
df['salary'] = [100000, 200000, 300000]
print(df)
print(" Magic functions: ".center(50, "*"))
range? ③
%timeit range(100) ④
```

① Explicitly specifies the IPython interpreter.

② Creates a Pandas DataFrame with further operations on it.

③ IPython's magic function that displays the method's help.

④ IPython's magic function for benchmarking the method execution.

Sample note output

```
****************** Describe DF:*******************
        age
count   3.0
mean   30.0
std     5.0
min    25.0
25%    27.5
50%    30.0
75%    32.5
max    35.0
******************* DF types: ********************
full_name    object
age           int64
dep          object
dtype: object
******************* Filter DF: *******************
     full_name  age dep
2  Ivan Ivanov   35  hr
*************** Add column to DF: ****************
      full_name  age    dep  salary
0  Sarah Connor   25     it  100000
1  Kirk Hammett   30  sales  200000
2   Ivan Ivanov   35     hr  300000
**************** Magic functions: ****************
108 ns ± 0.816 ns per loop (mean ± std. dev. of 7 runs, 10,000,000 loops each)
Init signature: range(self, /, *args, **kwargs)
Docstring:
range(stop) -> range object
range(start, stop[, step]) -> range object

Return an object that produces a sequence of integers from start (inclusive)
to stop (exclusive) by step.  range(i, j) produces i, i+1, i+2, ..., j-1.
start defaults to 0, and stop is omitted!  range(4) produces 0, 1, 2, 3.
These are exactly the valid indices for a list of 4 elements.
When step is given, it specifies the increment (or decrement).
Type:           type
Subclasses:
```

Below are examples of visualizing data in Zeppelin using various Python libraries.

Matplotlib

To run the sample note below, install Matplotlib to the Python instance used by Zeppelin. For example:

```
$ /opt/python3.10/bin/python3 -m pip install matplotlib
```

Sample note:

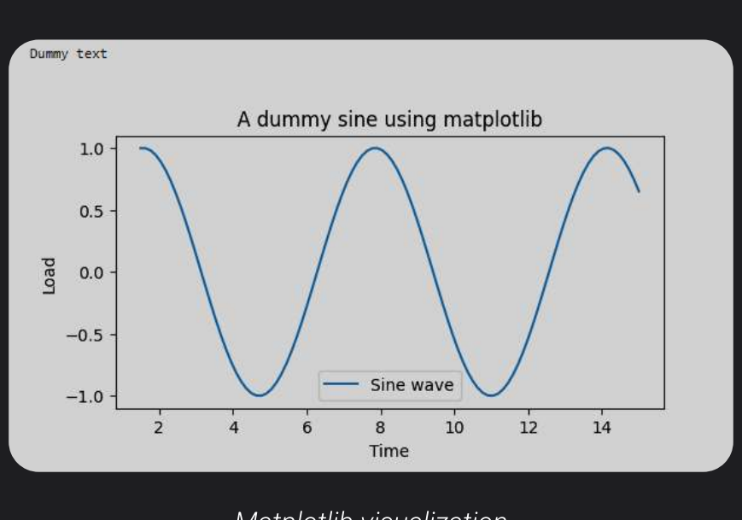
```
%python.ipython ①

import sys
import numpy as np ②
import matplotlib.pyplot as plt

print("Dummy text")

plt.figure(figsize=(6, 3))
x = np.linspace(1.5, 15, 100)
y = np.sin(x)
plt.plot(x, y, label="Sine wave")
plt.xlabel("Time")
plt.ylabel("Load")
plt.title("A dummy sine using matplotlib")
plt.legend()
plt.show() ③
```

① Explicitly specifies the IPython interpreter.

② Before running the note, imported modules must be installed via `pip install <module_name>`.

③ Renders a plot using the `matplotlib` library.

The resulting plot is below.



Matplotlib visualization

Seaborn

To run the sample note below, install Seaborn to the Python instance used by Zeppelin. For example:

```
$ /opt/python3.10/bin/python3 -m pip install seaborn
```

Sample note:

```
%python.ipython ①
%matplotlib inline ②

import seaborn as sns ③
import matplotlib.pyplot as plt
import pandas as pd

txn_data = {'acc_id': [1001,1003,1002,1001,1001,1002,1003,1001,1003],
        'txn_value': [25.00, 30.00, 100.00, 250.00, 98.50, 23.50, 45.00, 60.00, 150.15],
        'txn_date': ['2025-01-07', '2025-01-03', '2025-01-09', '2025-01-02', '2025-01-01', '2025-01-04', '2025-01-08', '2025-01-11', '2025-01-03']}
df = pd.DataFrame(txn_data)
plt.figure(figsize=(6, 4))
sns.boxplot(y='txn_value', x='acc_id', data=df) ④
plt.title("Transactions by account")
plt.xlabel("Account")
plt.ylabel("Transaction value")
plt.show()
```

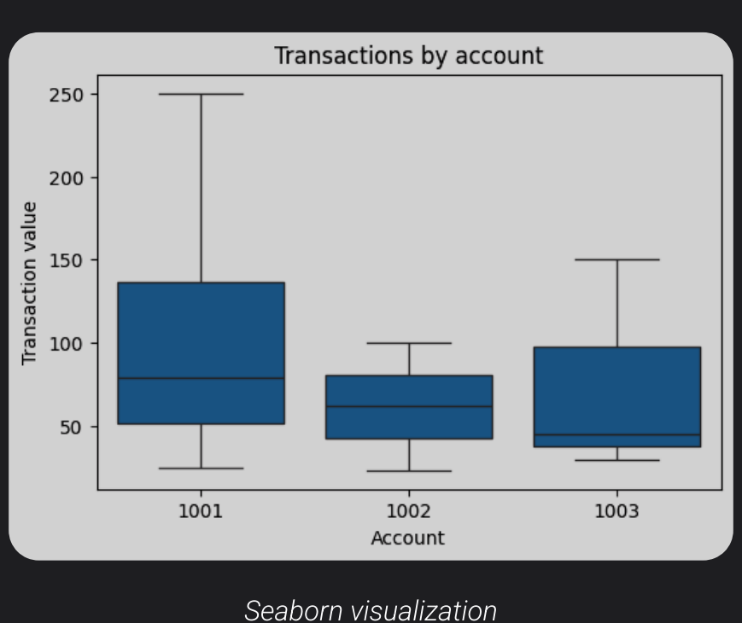
① Explicitly specifies the IPython interpreter.

② IPython's magic command that ensures that the rendered plot is displayed directly inline within the notebook.

③ Before running the note, imported modules must be installed via `pip install <module_name>`.

④ Generates a plot using the `seaborn` library.

The resulting plot is shown below.



Seaborn visualization

Plotly

To use Plotly for visualizing data in Zeppelin, install the following modules to the Python instance used by Zeppelin.

```
$ /opt/python3.10/bin/python3 -m pip install plotly
$ /opt/python3.10/bin/python3 -m pip install nbformat
```

Sample note:

```
%python.ipython ①
%matplotlib inline ②

import plotly.express as px ③
import matplotlib.pyplot as plt ③
import pandas as pd ③

txn_data = {'acc_id': [1001,1003,1002,1001,1001,1002,1003,1001,1003],
        'txn_value': [25.00, 30.00, 100.00, 250.00, 98.50, 23.50, 45.00, 60.00, 150.15],
        'txn_date': ['2025-01-02', '2025-01-03', '2025-01-04', '2025-01-01', '2025-01-01', '2025-01-05', '2025-01-03', '2025-01-06', '2025-01-05']}
df = pd.DataFrame(txn_data)
fig = px.scatter(df, ④
                x='txn_date',
                y='txn_value',
                color='acc_id',
                size="txn_value",
                title="Dummy Plotly scatter chart")
fig.update_layout(width=800, height=600)
fig.show()
```

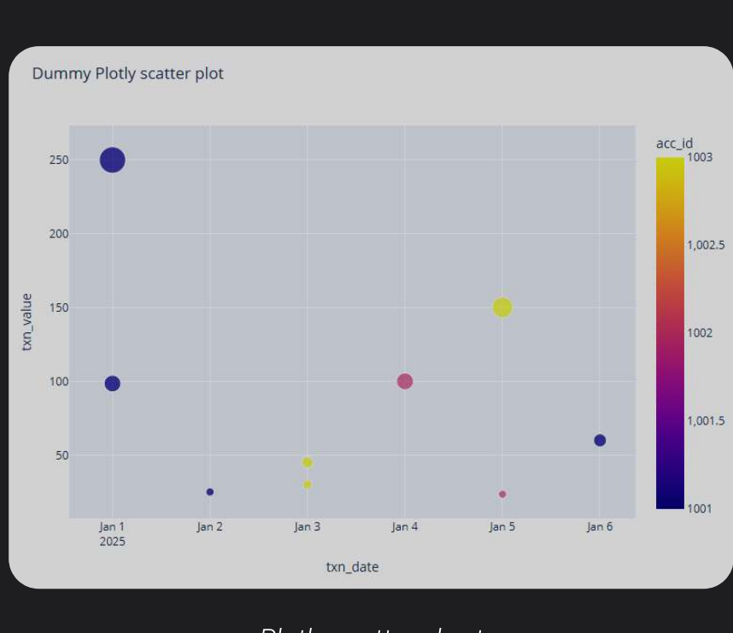
① Explicitly specifies the IPython interpreter.

② IPython's magic command that ensures that the rendered plot is displayed directly inline within the notebook itself.

③ Before running the note, imported modules must be installed via `pip install <module_name>`.

④ Renders a plot using the `Plotly` library.

The resulting plot is shown below.



Plotly scatter chart

SQL over Pandas DataFrames (%python.sql)

By using the `%python.sql` interpreter, you can use SQL to query Pandas DataFrames and visualize their content. This interpreter requires installation of the following pip modules:

- `pandas`
- `pandasql`

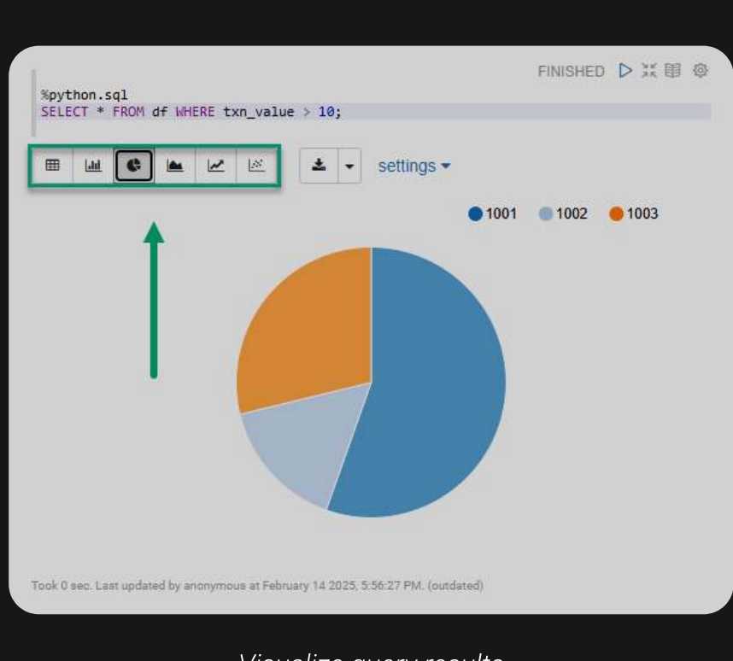
Example note:

```
%python.sql ①
SELECT * FROM df WHERE txn_value > 100; ②
```

① Specifies the interpreter for handling SQL over Pandas DataFrames.

② The `df` DataFrame must be created before running the query (in another Zeppelin paragraph).

Specifying `%spark.sql` on the first line of a paragraph enables the built-in Zeppelin visualization widgets. You can switch between several widgets to visualize the results on a pie chart, line graph, etc.



```
%python.sql
SELECT * FROM df WHERE txn_value > 10;
```

Took 0 sec. Last updated by anonymous at February 14 2025, 5:56:27 PM. (outdated)

Visualize query results

Logging in Zeppelin

Konstantin Alpashkin

Contents

To Table of Contents

- Logs location and format
- Log types
- Configure logging with ADCM
  - Default log4j.properties
- Configure logging via REST API
- Common tasks

Zeppelin server generates text logs that can be useful for analyzing the cause of different errors that may occur while working with interpreters, notes, etc. Under the hood, Zeppelin uses the Log4j framework for logging. To configure Zeppelin logging processes, use the ***Custom log4j.properties*** settings in ADCM. More details on using these settings are available in the Configure logging with ADCM section below.

Logs location and format

By default, Zeppelin logs are located in the */var/log/zeppelin* directory on the hosts where Zeppelin Server is deployed. To view these logs, do the following:

1. Connect to the Zeppelin Server host (via SSH, etc.) and run the command:

```
$ ls -la /var/log/zeppelin
```

   The result can look like this:

```
total 1768
drwxr-xr-x.  2 zeppelin zeppelin   4096 Dec  8 10:22 .
drwxr-xr-x. 19 root     root       4096 Dec  8 06:10 ..
-rw-r--r--.  1 root     root        555 Dec  7 06:54 install-interpreter.log
-rw-r--r--.  1 root     root       1665 Dec  6 16:13 install-interpreter.log.2021-12-06
-rw-r--r--.  1 zeppelin zeppelin   2656 Dec  8 10:30 zeppelin-interpreter-jdbc-zeppelin-bds-adh-2.ru-central1.internal.log
-rw-r--r--.  1 zeppelin zeppelin   5918 Dec  6 12:24 zeppelin-interpreter-jdbc-zeppelin-bds-adh-2.ru-central1.internal.log.2021-
12-06
-rw-r--r--.  1 zeppelin zeppelin  82200 Dec  7 18:15 zeppelin-interpreter-jdbc-zeppelin-bds-adh-2.ru-central1.internal.log.2021-
12-07
-rw-r--r--.  1 zeppelin zeppelin   2376 Dec  7 17:27 zeppelin-interpreter-phoenix2-zeppelin-bds-adh-2.ru-central1.internal.log
-rw-r--r--.  1 zeppelin zeppelin   4638 Dec  8 10:31 zeppelin-interpreter-phoenix-zeppelin-bds-adh-2.ru-central1.internal.log
-rw-r--r--.  1 zeppelin zeppelin 112258 Dec  6 16:11 zeppelin-interpreter-phoenix-zeppelin-bds-adh-2.ru-central1.internal.log.
2021-12-06
-rw-r--r--.  1 zeppelin zeppelin  17404 Dec  7 17:27 zeppelin-interpreter-phoenix-zeppelin-bds-adh-2.ru-central1.internal.log.
2021-12-07
-rw-r--r--.  1 zeppelin zeppelin   3951 Dec  7 18:15 zeppelin-interpreter-python-zeppelin-bds-adh-2.ru-central1.internal.log
-rw-r--r--.  1 zeppelin zeppelin   4134 Dec  6 16:01 zeppelin-interpreter-sh-zeppelin-bds-adh-2.ru-central1.internal.log
-rw-r--r--.  1 zeppelin zeppelin  61835 Dec  8 12:09 zeppelin-zeppelin-bds-adh-2.ru-central1.internal.log
-rw-r--r--.  1 zeppelin zeppelin 722037 Dec  6 16:30 zeppelin-zeppelin-bds-adh-2.ru-central1.internal.log.2021-12-06
-rw-r--r--.  1 zeppelin zeppelin 662981 Dec  7 18:15 zeppelin-zeppelin-bds-adh-2.ru-central1.internal.log.2021-12-07
-rw-r--r--.  1 zeppelin zeppelin  78400 Dec  8 06:11 zeppelin-zeppelin-bds-adh-2.ru-central1.internal.out
```

> **NOTE**
>
> Logs are stored on the local file system of the host, not in HDFS.

2. To view the content of a specific log file, run the following:

```
$ tail /var/log/zeppelin/zeppelin-zeppelin-bds-adh-2.ru-central1.internal.log
```

   The output can be similar to:

```
 INFO [2021-12-08 10:31:02,382] ({pool-2-thread-4} RemoteInterpreterManagedProcess.java[start]:115) - Thrift server for callback
will start. Port: 38183
 INFO [2021-12-08 10:31:02,882] ({pool-2-thread-4} RemoteInterpreterManagedProcess.java[start]:190) - Run interpreter process
[/usr/lib/zeppelin/bin/interpreter.sh, -d, /usr/lib/zeppelin/interpreter/jdbc, -c, 10.92.6.9, -p, 38183, -r, :, -l,
/usr/lib/zeppelin/local-repo/phoenix, -g, phoenix]
 INFO [2021-12-08 10:31:04,138] ({pool-11-thread-1} RemoteInterpreterManagedProcess.java[callback]:123) - RemoteInterpreterServer
Registered: CallbackInfo(host:10.92.6.9, port:34481)
 INFO [2021-12-08 10:31:04,140] ({pool-2-thread-4} RemoteInterpreter.java[call]:168) - Create RemoteInterpreter org.apache.
zeppelin.jdbc.JDBCInterpreter
 INFO [2021-12-08 10:31:04,239] ({pool-2-thread-4} RemoteInterpreter.java[call]:142) - Open RemoteInterpreter org.apache.zeppelin.
jdbc.JDBCInterpreter
 INFO [2021-12-08 10:31:04,240] ({pool-2-thread-4} RemoteInterpreter.java[pushAngularObjectRegistryToRemote]:436) - Push local
angular object registry from ZeppelinServer to remote interpreter group phoenix:shared_process
 INFO [2021-12-08 10:31:04,830] ({pool-2-thread-4} NotebookServer.java[afterStatusChange]:2314) - Job 20211207-082338_940138511
is finished successfully, status: FINISHED
 INFO [2021-12-08 10:31:04,836] ({pool-2-thread-4} VFSNotebookRepo.java[save]:196) - Saving note:2GS5T8MS7
 INFO [2021-12-08 10:31:04,838] ({pool-2-thread-4} SchedulerFactory.java[jobFinished]:120) - Job 20211207-082338_940138511
finished by scheduler org.apache.zeppelin.interpreter.remote.RemoteInterpreter-phoenix:shared_process-shared_session
 INFO [2021-12-08 12:09:00,355] ({qtp837764579-66} NotebookServer.java[onClose]:372) - Closed connection to 10.92.2.6 : 54643.
(1001) null
```

Log types

There are several types of logs generated by Zeppelin. Their description is listed below.

Log name	Attributes	Description
zeppelin-zeppelin-<host_name>.log	`<host_name>` — Zeppelin host name	Contains information about basic operations with connections, jobs, notes: opening, closing, saving, etc.
zeppelin-zeppelin-<host_name>.out	`<host_name>` — Zeppelin host name	Stores Zeppelin console output for performed operations, i.e. errors, returned in the web UI
zeppelin-interpreter-<interpreter_name>-zeppelin-<host_name>.log	▪ `<host_name>` — Zeppelin host name. ▪ `<interpreter_name>` — interpreter name.	Contains information reflecting the activity of a specific interpreter
install-interpreter.log	—	Contains information about installing new interpreters

Configure logging with ADCM

With the use of Log4j, configuring Zeppelin logging assumes editing the *log4j.properties* configuration file. This file is located in the */etc/zeppelin/conf* directory on the host where the Zeppelin Server component is installed, and Zeppelin reads this file during the startup. However, instead of editing the file manually on each host, ADCM provides a way to update the Log4j configuration on all ADH hosts at once. For this:

1. In ADCM, go to ***Clusters*** → ***<your_cluster_name>*** → ***Services*** → ***Zeppelin*** → ***Primary Configuration*** and enable the ***Show advanced*** option.
2. Click ***Custom log4j.properties*** and edit the configuration. The Log4j configuration specified via ADCM UI will overwrite the contents of the */etc/zeppelin/conf/log4j.properties* file during the service restart. For detailed reference on supported Log4j properties, see Log4j documentation.
3. Save the Zeppelin service configuration.
4. Restart the Zeppelin service.

Default log4j.properties

Below is the default *log4j.properties* file used by Zeppelin. The major properties are highlighted to help you get started with Log4j configuration.

```
#
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements.  See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership.  The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License.  You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
log4j.rootLogger = INFO, dailyfile ①

log4j.appender.stdout = org.apache.log4j.ConsoleAppender ②
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%5p [%d] ({%t} %F[%M]:%L) - %m%n ⑤

log4j.appender.dailyfile.DatePattern=.yyyy-MM-dd
log4j.appender.dailyfile.Threshold = INFO
log4j.appender.dailyfile = org.apache.log4j.DailyRollingFileAppender ③
log4j.appender.dailyfile.File = ${zeppelin.log.file} ④
log4j.appender.dailyfile.layout = org.apache.log4j.PatternLayout
log4j.appender.dailyfile.layout.ConversionPattern=%5p [%d] ({%t} %F[%M]:%L) - %m%n ⑤
```

① Sets the root logging level (`INFO`) for the default logger. Events with lower severity are not included in logs. For example, using `DEBUG` generates more verbose logs, whereas using `ERROR` will force Zeppelin to log only errors and more critical events.

② Defines a ConsoleAppender that outputs log events to STDOUT.

③ Defines a DailyRollingFileAppender that writes log events to a file, which is rotated every day.

④ Specifies a path to the log file. The default `${zeppelin.log.file}` property points to */var/log/zeppelin*.

⑤ Specifies a pattern for converting event objects to strings. The default pattern has the following structure:

```
[Severity] [Date] ([Thread] [File name][Method][Line number]) [Log message]
```

Configure logging via REST API

The Zeppelin service exposes a REST API through which you can interact with Zeppelin, including logs configuration commands. The REST endpoint to change the logging level is as follows:

```
PUT http://<zeppelin-host>:<zeppelin-port>/api/log/level/<LOG_LEVEL>
```

Where:

- `<zeppelin-host>` is the address of the host with the running Zeppelin Server.
- `<zeppelin-port>` is the port number used by the Zeppelin REST API. The port number can be changed with the `zeppelin.server.port` property in ADCM. Defaults to `8180`.
- `<LOG_LEVEL>` is the logging level to be set. The possible severity levels are: `ALL`, `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`, `FATAL`, and `OFF`.

Example:

```
$ curl -X PUT http://localhost:8180/api/log/level/WARN
```

The response:

```
{"status":"OK"}
```

Common tasks

Below are some common tasks that may arise while configuring Zeppelin logging with Log4j.

Change the default log level

To change the root logging level (defaults to `INFO`), use the `log4j.rootLogger=<LEVEL>,<APPENDER>` property in the ***Custom log4j.properties*** section in ADCM. You can set a root log level, which will be inherited by non-root Log4j loggers, or you can set log levels individually for a specific logger, for example `log4j.logger.org.apache.zeppelin.interpreter=<OTHER_LEVEL>`.

Change default logs location

By default, Zeppelin logs are stored under the */var/log/zeppelin* directory. To change this directory, modify the `log4j.appender.<appender-name>.File` property where `<appender-name>` is the name of the Log4j appender bound to the root logger.

Add a custom logger

The following example shows how to add a custom logger that writes logs to a separate directory. Also, the example shows how to send log events from a Spark interpreter code to the log file.

1. Add the following properties to your ***Custom log4j.properties*** section in ADCM. The major properties are highlighted with comments.

```
# ...
# New logger definition:
log4j.logger.MyCustomLogger=DEBUG, mycustomappender ①
log4j.appender.mycustomappender=org.apache.log4j.RollingFileAppender ②
log4j.appender.mycustomappender.File = /var/log/zeppelin/customlogger/test.log ③
log4j.appender.mycustomappender.rollingPolicy.FileNamePattern=/var/log/zeppelin/customlogger/test.%i.log ④
log4j.appender.mycustomappender.layout=org.apache.log4j.PatternLayout
log4j.appender.mycustomappender.layout.conversionPattern=[%p] %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %c{10} - %m %n
log4j.appender.mycustomappender.MaxFileSize=10MB ⑤
log4j.appender.mycustomappender.MaxBackupIndex=10 ⑥
{% endraw %}
```

   ① Defines a new logger and binds an appender.

   ② Defines a RollingFileAppender to back up the log files when they reach a certain size.

   ③ Specifies the location for log files generated by the logger.

   ④ Specifies the file name pattern when a log file gets rolled over.

   ⑤ Sets the maximum log file size for making a rollover.

   ⑥ Sets the maximum number of backup log files to keep.

2. Save the configuration and restart the Zeppelin service.
3. Run a sample Spark notebook to generate some log entries. For this, create a new Spark note using Zeppelin UI and run the following code in the notebook:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

val logger: Logger = LoggerFactory.getLogger("MyCustomLogger");
logger.info("Test info message");
```

4. Check the logs generated by the custom logger. For this, connect to the Zeppelin Server host and run the command:

```
$ ls -l /var/log/zeppelin/customlogger
```

   The output:

```
-rw-r--r-- 1 zeppelin zeppelin 114 Nov 25 13:14 test.log
```

   The *test.log* file contains the only log record generated from the Spark notebook:

```
[INFO] 2024-11-22 13:14:28.569 [FIFOScheduler-interpreter_453440904-Worker-1] MyCustomLogger - Test info message
```

Zeppelin performance tuning

Konstantin Alpashkin

Contents

To Table of Contents

- Memory
- Interpreter binding mode
- Spark optimizations
- Logging
- Browser
- Other

This article describes optimization techniques for the Zeppelin service and highlights key factors that affect the Zeppelin performance.

Memory

The major factor that affects the Zeppelin service is the amount of RAM available for the Zeppelin Server component. You can speed up the Zeppelin service performance by allocating more RAM for the Zeppelin JVM process. To increase `-Xms` / `-Xmx` parameters, use ***Zeppelin Server Heap Memory*** settings in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Zeppelin*** → ***Primary configuration***) and restart the service.

When estimating the amount of memory that will satisfy your setup, take into account the following major characteristics:

- The number of users that will connect to the Zeppelin server in parallel.
- The binding mode used by interpreters.

Interpreter binding mode

Zeppelin interpreters run as separate JVM processes that execute notes code and communicate with the Zeppelin daemon. Zeppelin provides several interpreter binding modes that define ways for launching new interpreter processes. For a multi-user environment, this mode is the most RAM-expensive. Be sure the ADH host with the Zeppelin Server component can handle the number of interpreter JVMs for each user note. These modes are shared, scoped, and isolated.

- **Shared**.

  In this mode, a single JVM process with one interpreter session serves all notebooks. This approach ensures less memory consumption for a multi-user environment and also provides cross-note collaboration capabilities, such as accessing variables created in other notes. The latter can be especially beneficial for reusing "heavy" objects like `SparkContext`, `SparkSession`, etc.

  To increase the amount of RAM allocated by the Zeppelin Server to each interpreter process, specify the `ZEPPELIN_INTP_MEM` parameter in the ***Custom zeppelin-env.sh*** field in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Zeppelin*** → ***Primary configuration*** and enable ***Show advanced***). For example:

  ```
  ZEPPELIN_INTP_MEM="-Xms2048m -Xmx4096m -XX:MaxMetaspaceSize=1024m"
  ```

- **Scoped**.

  In this mode, Zeppelin runs a single JVM per interpreter and executes each note in its own dedicated session, depending on whether ***Per Note*** or ***Per User*** scope is selected.

- **Isolated**.

  In this mode, each note spawns a new interpreter JVM process (in the ***Per note*** scope). The overall memory consumption increases, however, Zeppelin users can run notes in a fully isolated environment, avoiding possible conflicts. Additionally, the restart of an interpreter by one user does not affect others.

Tweaking with binding modes can help optimize resources consumed by Zeppelin and select an approach that best fits your setup.

Spark optimizations

When working with Spark interpreters, to speed up Spark jobs execution, you can use the optimization techniques described in the Spark performance tuning article. To specify Spark configuration properties, use the ***Custom zeppelin-env.sh*** field in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Zeppelin*** → ***Primary configuration*** and enable ***Show advanced***). For example:

```
export SPARK_SUBMIT_OPTIONS="--executor-memory 4G"
```

> **NOTE**
>
> The first run of a Spark note is always slower since the Spark interpreter has to instantiate heavy context objects and create Application Master.

Logging

Setting the Zeppelin log level to `DEBUG` may slow down the UI response, especially when saving an interpreter configuration during the creation of a new notebook. Enable the `DEBUG` level only for troubleshooting.

To change the logging level in Zeppelin, use the ***Custom log4j.properties*** settings in ADCM (***Clusters*** → ***<clusterName>*** → ***Services*** → ***Zeppelin*** → ***Primary configuration*** and enable ***Show advanced***).

Browser

Use a modern web browser capable of loading large notebooks (10+ MB), visualizing tens and hundreds of megabytes of output data, supporting third-party visualization libraries, etc. For example, older versions of IE have been reported to have issues with login/authentication actions.

Clearing massive paragraphs' output periodically will reduce the load on the browser/JVM since Zeppelin stores the last execution output internally.

Other

- Restart interpreters periodically. Long-running interpreters may slow down over time.
- Limit the number of output rows returned by SQL queries. For example, for Spark interpreters, this can be achieved using the `zeppelin.spark.maxResult` interpreter configuration property.
- Use `%table` for more efficient visualization of large result sets.

Zeppelin service management via ADCM

Konstantin Alpashkin

To Table of Contents

The ADCM UI provides actions to manage the Zeppelin service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the Zeppelin service are listed in the table below.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute Zeppelin components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM

Connect to ZooKeeper via CLI

Daria Barysheva

To Table of Contents

The simplest way to begin working with ZooKeeper is to use its command line interface (CLI). This console is available on each ZooKeeper node immediately after its installation. To start working with CLI, run the *zkCli.sh* script located in the */bin* directory:

```
$ sudo /usr/lib/zookeeper/bin/zkCli.sh
```

The output

```
2021-12-15 11:04:55,901 [myid:] - INFO  [main:Environment@100] - Client environment:zookeeper.version=3.4.14-
f42ef4b166f1e5c63117c75aed651589a0cce1b2, built on 08/06/2019 14:49 GMT
2021-12-15 11:04:55,903 [myid:] - INFO  [main:Environment@100] - Client environment:host.name=bds-adh-1.ru-central1.internal
2021-12-15 11:04:55,904 [myid:] - INFO  [main:Environment@100] - Client environment:java.version=1.8.0_312
2021-12-15 11:04:55,905 [myid:] - INFO  [main:Environment@100] - Client environment:java.vendor=Red Hat, Inc.
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:java.home=/usr/lib/jvm/java-1.8.0-openjdk-
1.8.0.312.b07-1.el7_9.x86_64/jre
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:java.class.
path=/usr/lib/zookeeper/bin/../zookeeper-
server/target/classes:/usr/lib/zookeeper/bin/../build/classes:/usr/lib/zookeeper/bin/../zookeeper-server/target/lib/*.
jar:/usr/lib/zookeeper/bin/../build/lib/*.jar:/usr/lib/zookeeper/bin/../lib/slf4j-log4j12-1.7.25.
jar:/usr/lib/zookeeper/bin/../lib/slf4j-api-1.7.25.jar:/usr/lib/zookeeper/bin/../lib/netty-3.10.6.Final.
jar:/usr/lib/zookeeper/bin/../lib/log4j-1.2.17.jar:/usr/lib/zookeeper/bin/../lib/jline-0.9.94.
jar:/usr/lib/zookeeper/bin/../lib/audience-annotations-0.5.0.jar:/usr/lib/zookeeper/bin/../zookeeper-3.4.14.
jar:/usr/lib/zookeeper/bin/../zookeeper-server/src/main/resources/lib/*.jar:/usr/lib/zookeeper/bin/../conf:
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:java.library.
path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:java.io.tmpdir=/tmp
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:java.compiler=<NA>
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:os.name=Linux
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:os.arch=amd64
2021-12-15 11:04:55,906 [myid:] - INFO  [main:Environment@100] - Client environment:os.version=3.10.0-1160.11.1.el7.x86_64
2021-12-15 11:04:55,907 [myid:] - INFO  [main:Environment@100] - Client environment:user.name=root
2021-12-15 11:04:55,907 [myid:] - INFO  [main:Environment@100] - Client environment:user.home=/root
2021-12-15 11:04:55,907 [myid:] - INFO  [main:Environment@100] - Client environment:user.dir=/home/dasha
2021-12-15 11:04:55,908 [myid:] - INFO  [main:ZooKeeper@442] - Initiating client connection, connectString=localhost:2181
sessionTimeout=30000 watcher=org.apache.zookeeper.ZooKeeperMain$MyWatcher@4b85612c
Welcome to ZooKeeper!
2021-12-15 11:04:55,928 [myid:] - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@1025] - Opening socket
connection to server localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error)
JLine support is enabled
2021-12-15 11:04:55,973 [myid:] - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@879] - Socket connection
established to localhost/127.0.0.1:2181, initiating session
2021-12-15 11:04:55,981 [myid:] - INFO  [main-SendThread(localhost:2181):ClientCnxn$SendThread@1299] - Session establishment
complete on server localhost/127.0.0.1:2181, sessionid = 0x100005560090006, negotiated timeout = 30000

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[zk: localhost:2181(CONNECTED) 0]
```

The ZooKeeper CLI prompt starts with `[zk: localhost:2181(CONNECTED) 0]`. All subsequent commands should be written after it.

To logout from ZooKeeper CLI, use the following command:

```
quit
```

Connect to ZooKeeper via API

Elena Kostyuchenko

Contents

To Table of Contents

ZooKeeper client

Java

C

ZooKeeper provides official API with Java and C bindings. APIs for some of the other languages are provided by the ZooKeeper community.

The ZooKeeper API enables applications to connect to a server, execute commands, manipulate data, and coordinate tasks.

ZooKeeper client

An application that interacts with a ZooKeeper server is called a ZooKeeper client.

The prefered algorithm for client interaction with the ZooKeeper server looks like this:

1. Create a connection to the server. The server will assign a session ID for the client.
2. Send heartbeats to the server. Otherwise, the session ID expires and the client will have to reconnect.
3. Perform the necessary operations, for example, get or set the znodes.
4. Disconnect from the server. If the client is inactive for a prolonged time, the server will automatically disconnect the client.

To learn more about developing for Zookeeper, read the ZooKeeper Programmer's Guide.

Java

To connect to a ZooKeeper server using the Java API, you need to set up an environment and install the dependencies. The example code below creates a connection to the ZooKeeper server, outputs confirmation upon successful connection, and returns an error message if it was unable to connect.

The developer tools used in the process are Vi text editor, virtual machine with CentOS 7, and Maven compiler. A different setup will require calibration.

1. Install Maven.
2. Write a connector class:

```java
package my.example.zookeeper;

import java.io.IOException;
import java.util.concurrent.CountDownLatch;

import org.apache.zookeeper.KeeperException;
import org.apache.zookeeper.WatchedEvent;
import org.apache.zookeeper.Watcher;
import org.apache.zookeeper.Watcher.Event.KeeperState;
import org.apache.zookeeper.ZooKeeper;
import org.apache.zookeeper.AsyncCallback.StatCallback;
import org.apache.zookeeper.KeeperException.Code;
import org.apache.zookeeper.data.Stat;

public class ZooKeeperConnection {

   private ZooKeeper zoo;
   final CountDownLatch connectedSignal = new CountDownLatch(1);

   public ZooKeeper connect(String host) throws IOException,InterruptedException {

      zoo = new ZooKeeper(host,5000,new Watcher() {

         public void process(WatchedEvent we) {

            if (we.getState() == KeeperState.SyncConnected) {
               connectedSignal.countDown();
            }
         }
      });

      connectedSignal.await();
      return zoo;
   }

   public void close() throws InterruptedException {
      zoo.close();
   }
}
```

3. Create a *pom.xml* file for building the project with Maven.

Sample pom.xml file

```xml
<project>
        <modelVersion>4.0.0</modelVersion>
        <groupId>my.example.zookeeper</groupId>
        <artifactId>zoo-connect</artifactId>
        <version>0.1.0-SNAPSHOT</version>
        <packaging>jar</packaging>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
  <dependencies>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-hdfs</artifactId>
            <version>2.7.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.zookeeper</groupId>
            <artifactId>zookeeper</artifactId>
            <version>3.4.11</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>2.7.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-core</artifactId>
            <version>1.2.1</version>
        </dependency>

        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-client</artifactId>
            <version>2.7.1</version>
        </dependency>
  </dependencies>
<build>
    <plugins>
      <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.1.0</version>
            <configuration>
              <archive>
                <manifest>
                  <addClasspath>true</addClasspath>
                  <mainClass>my.example.zookeeper.Main</mainClass>
                </manifest>
              </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>assemble-all</id>
                    <phase>package</phase>
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
      <plugin>
        <!-- Build an executable JAR -->
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.1.0</version>
        <configuration>
          <archive>
            <manifest>
              <addClasspath>true</addClasspath>
              <mainClass>my.example.zookeeper.Main</mainClass>
            </manifest>
             <addClasspath>true</addClasspath>
              <mainClass>my.example.zookeeper.Main</mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

4. Build the project:

```
$ mvn package
```

5. Run the created Java class:

```
$ java -jar <java-class>.jar
```

   Where `<java-class>` is the name of your Java connector program.

The output will include information about successful connection:

```
23/12/14 19:43:34 INFO zookeeper.ZooKeeper: Initiating client connection, connectString=127.0.0.1 sessionTimeout=2181 watcher=my.
example.zookeeper.Main$MyWatcher@433c675d
23/12/14 19:43:34 INFO zookeeper.ClientCnxn: Opening socket connection to server 127.0.0.1/127.0.0.1:2181. Will not attempt to
authenticate using SASL (unknown error)
23/12/14 19:43:34 INFO zookeeper.ClientCnxn: Socket connection established to 127.0.0.1/127.0.0.1:2181, initiating session
23/12/14 19:43:34 INFO zookeeper.ClientCnxn: Session establishment complete on server 127.0.0.1/127.0.0.1:2181, sessionid =
0x1000000cebe0004, negotiated timeout = 4000
23/12/14 19:43:34 INFO zookeeper.ZooKeeper: Session: 0x1000000cebe0004 closed
```

C

To connect to a ZooKeeper server using the C API, you need to set up an environment and install the dependencies. The example code below creates a connection to the ZooKeeper server, outputs confirmation upon successful connection, and returns an error message if it was unable to connect.

The developer tools used in the process are Vi text editor, virtual machine with Ubuntu 23.10, and GCC compiler. A different setup will require calibration.

Another example of a C client is available in the Apache ZooKeeper repository.

Before building a ZooKeeper client in C, download the zookeeper.h library or build it yourself using the `ant` build tool. If you are building a multithreaded client, compile with `-DTHREADED` compiler flag to enable the multi-threaded version of the library, and then link against the zookeeper_mt library. If you are building a single-threaded client, do not compile with `-DTHREADED`, and link against the zookeeper_st library.

To create a sample ZooKeeper client in C:

1. Write a connector program:

```c
#include <stdio.h>
#include <stdlib.h>
#include <zookeeper/zookeeper.h>
#include <errno.h>

static int completed = 0;

void watcher(zhandle_t *zkH, int type, int state, const char *path, void *watcherCtx) {
    if (type == ZOO_SESSION_EVENT) {
        if (state == ZOO_CONNECTED_STATE) {
        } else if (state == ZOO_NOTCONNECTED_STATE ) {
        } else if (state == ZOO_EXPIRED_SESSION_STATE) {
            completed = 1;
            zookeeper_close(zkH);
        }
    }
}

void aget_strings_completion(int rc, const struct String_vector *strings, const void *data) {
    int i;
    if (strings) {
        for (i=0; i < strings->count; i++) {
            printf("\t%s\n", strings->data[i]);
        }
    }
    completed = 1;
}

int main(int argc, char *argv[]) {
    zoo_set_debug_level(ZOO_LOG_LEVEL_DEBUG);
    zhandle_t *handler = zookeeper_init("127.0.0.1:2181", watcher, 30000, NULL, NULL, 0);
    if (!handler) {
        return EXIT_FAILURE;
    }
    int rc = zoo_aget_children(handler, "/", 1, aget_strings_completion, NULL);
    if (rc) {
        fprintf(stderr, "Error %d\n", rc);
        return EXIT_FAILURE;
    }
    while (!completed) {
    }
    zookeeper_close(handler);
    return EXIT_SUCCESS;
}
```

2. Compile the code. For example, using the following command:

```
$ gcc main.c -o <program-name> -lzookeeper_mt
```

   Where `<program-name>` is the name of your program.

With the client console output configured, you can see the following:

```
2023-12-18 12:23:11,953:6904(0x7fa0e9d146c0):ZOO_INFO@check_events@2992: initiated connection to server 127.0.0.1:2181
2023-12-18 12:23:11,965:6904(0x7fa0e9d146c0):ZOO_INFO@finalize_session_establishment@2866: session establishment complete on
server 127.0.0.1:2181, sessionId=0x1000000cebe0005, negotiated timeout=30000
2023-12-18 12:23:11,965:6904(0x7fa0e9d146c0):ZOO_DEBUG@finalize_session_establishment@2876: Calling a watcher for a
ZOO_SESSION_EVENT and the state=ZOO_CONNECTED_STATE
2023-12-18 12:23:11,965:6904(0x7fa0e95136c0):ZOO_DEBUG@process_completions@3341: Calling a watcher for node [], type = -1
event=ZOO_SESSION_EVENT
2023-12-18 12:23:11,966:6904(0x7fa0e9d146c0):ZOO_DEBUG@zookeeper_process@3503: Queueing asynchronous response
2023-12-18 12:23:11,966:6904(0x7fa0e95136c0):ZOO_DEBUG@deserialize_response@3233: Calling COMPLETION_STRINGLIST for
xid=0x658039b0 failed=0 rc=0
        zookeeper
        arenadata
        hadoop-ha
2023-12-18 12:23:11,966:6904(0x7fa0e95136c0):ZOO_DEBUG@do_completion@483: completion thread terminated
2023-12-18 12:23:11,966:6904(0x7fa0e9d146c0):ZOO_DEBUG@do_io@460: IO thread terminated
2023-12-18 12:23:11,966:6904(0x1a373c0):ZOO_INFO@zookeeper_close@3854: Closing zookeeper sessionId=0x1000000cebe0005 to 127.0.0.
1:2181
```

Work with znodes

Daria Barysheva

Contents

To Table of Contents

Step 1. Create a znode

Step 2. Get znode data

Step 3. Set data to the znode

Step 4. Create znode children

Step 5. List znode children

Step 6. Get znode metadata

Step 7. Watch a znode for changes

Step 8. Remove the znode

> **NOTE**
>
> Before executing the following commands, read about ZooKeeper znodes and watches.

Step 1. Create a znode

To create a new znode, use the `create` command, then define the full path to the new znode, and then write the data to be stored in this znode. Depending on the znode type, additional flags can be required:

- **Persistent**. Use the default call of the `create` command without additional flags:

```
create /z_persistent "Test configuration"
```

The output:

```
Created /z_persistent
```

- **Sequential**. Use the `-s` flag after the `create` keyword:

```
create -s /z_sequential "Test configuration 2"
```

> **NOTE**
>
> ZooKeeper adds the 10-digit postfix to the original znode name. This change guarantees the uniqueness of the sequential znode name.

The output is listed below:

```
Created /z_sequential0000000007
```

- **Ephemeral**. Use the `-e` flag after the `create` keyword:

```
create -e /z_ephemeral "Test configuration 3"
```

The output:

```
Created /z_ephemeral
```

Step 2. Get znode data

To get the data stored in the specified znode, use the `get` command with the full path to this znode:

```
get /z_persistent
```

You can see that the command output contains not only the data stored in the specified znode, but also its metadata — so-called stat:

```
Test configuration
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x2000000019
mtime = Wed Dec 15 11:15:27 UTC 2021
pZxid = 0x2000000019
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 18
numChildren = 0
```

Below you can find the examples of getting data for two other znodes.

> **IMPORTANT**
>
> When applying the `get` command to sequential znodes, use their full names including the 10-digit numbers assigned by ZooKeeper automatically. This rule also applies to the set and rmr commands.

```
[zk: localhost:2181(CONNECTED) 7]get /z_sequential0000000007
Test configuration 2
cZxid = 0x200000001c
ctime = Wed Dec 15 11:17:47 UTC 2021
mZxid = 0x200000001c
mtime = Wed Dec 15 11:17:47 UTC 2021
pZxid = 0x200000001c
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 20
numChildren = 0
[zk: localhost:2181(CONNECTED) 8]get /z_ephemeral
Test configuration 3
cZxid = 0x200000001d
ctime = Wed Dec 15 11:18:35 UTC 2021
mZxid = 0x200000001d
mtime = Wed Dec 15 11:18:35 UTC 2021
pZxid = 0x200000001d
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x100005560090006
dataLength = 20
numChildren = 0
```

> **NOTE**
>
> Starting ZooKeeper 3.5, the `get /znode` command does not output znode data for container nodes. Use `get -s /znode` instead.

Step 3. Set data to the znode

To assign a new data value to the specified znode, write the `set` command, the full path to this znode, and then the data to be stored:

```
set /z_persistent "The test configuration has been updated with the command set"
```

The output is similar to the following:

```
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x200000001e
mtime = Wed Dec 15 11:24:46 UTC 2021
pZxid = 0x2000000019
cversion = 0
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 60
numChildren = 0
```

Now, if you run the `get` command, you will receive the updated data value of the znode:

```
get /z_persistent
```

The output:

```
The test configuration has been updated with the command set
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x200000001e
mtime = Wed Dec 15 11:24:46 UTC 2021
pZxid = 0x2000000020
cversion = 2
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 60
numChildren = 2
```

Step 4. Create znode children

To add child nodes for the specified znode, you can use the standard command `create` and define the path to the parent znode instead of the root path `/`.

```
create /z_persistent/child1 "Test child 1"
```

The examples below create two children for the parent znode `z_persistent`:

- Create the first child:

```
create /z_persistent/child1 "Test child 1"
```

The output:

```
Created /z_persistent/child1
```

- Create the second child:

```
create /z_persistent/child2 "Test child 2"
```

The output:

```
Created /z_persistent/child2
```

Step 5. List znode children

To display the children of the specified znode, use the `ls` command with the full path to the parent znode:

```
ls /z_persistent
```

The output is similar to the following:

```
[child2, child1]
```

You can also get the children of the root znode, using the `/` path:

```
ls /
```

The result is listed below:

```
[zookeeper, Arenadata.Hadoop-1.solr.server, z_sequential0000000007, hadoop-ha, z_ephemeral, hbase, z_persistent]
```

Additionally, you can use the `ls2` command to get not only the children of the specified znode, but also its metadata.

```
ls2 /
```

The output:

```
[zookeeper, Arenadata.Hadoop-1.solr.server, hadoop-ha, z_ephemeral, z_sequential0000000007, hbase, z_persistent]
cZxid = 0x0
ctime = Thu Jan 01 00:00:00 UTC 1970
mZxid = 0x0
mtime = Thu Jan 01 00:00:00 UTC 1970
pZxid = 0x2000000030
cversion = 17
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 0
numChildren = 7
```

Step 6. Get znode metadata

To get metadata of the specified znode (so-called stat), use the `stat` command with the full path to this znode. In fact, this command outputs the same information as `get`, except for the stored data itself:

```
stat /z_persistent
```

The output of running the command for the previously created znodes is shown below:

```
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x200000001e
mtime = Wed Dec 15 11:24:46 UTC 2021
pZxid = 0x2000000020
cversion = 2
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 60
numChildren = 2
[zk: localhost:2181(CONNECTED) 15] stat /z_sequential0000000007
cZxid = 0x200000001c
ctime = Wed Dec 15 11:17:47 UTC 2021
mZxid = 0x200000001c
mtime = Wed Dec 15 11:17:47 UTC 2021
pZxid = 0x200000001c
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 20
numChildren = 0
[zk: localhost:2181(CONNECTED) 16] stat /z_ephemeral
cZxid = 0x200000001d
ctime = Wed Dec 15 11:18:35 UTC 2021
mZxid = 0x200000001d
mtime = Wed Dec 15 11:18:35 UTC 2021
pZxid = 0x200000001d
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x100005560090006
dataLength = 20
numChildren = 0
```

Step 7. Watch a znode for changes

Watches allow a client to get notifications about changes in the monitored znodes. To create a new Watch, run the `get` command with the additional argument, which is the watcher name.

> **TIP**
>
> You can create a Watch using the following commands: `get`, `stat`, `ls`, and `ls2`.

```
get /z_persistent 1
```

The output is the same as for the `get` standard call:

```
The test configuration has been updated with the command set
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x200000001e
mtime = Wed Dec 15 11:24:46 UTC 2021
pZxid = 0x2000000020
cversion = 2
dataVersion = 1
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 60
numChildren = 2
```

Next, run the `set` command to change the znode data:

```
set /z_persistent "Updated"
```

This time, ZooKeeper prints out not only the regular output of the `set` command, but also the notification telling the client about changes applied to the monitored znode:

```
WATCHER::

WatchedEvent state:SyncConnected type:NodeDataChanged path:/z_persistent
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x2000000021
mtime = Wed Dec 15 11:37:02 UTC 2021
pZxid = 0x2000000020
cversion = 2
dataVersion = 2
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 7
numChildren = 2
```

Remember that each Watch is triggered only once. Next calls of the `set` command will not result in sending a notification, until the client creates a new Watch.

```
[zk: localhost:2181(CONNECTED) 24] set /z_persistent "Updated 2"
cZxid = 0x2000000019
ctime = Wed Dec 15 11:15:27 UTC 2021
mZxid = 0x2000000022
mtime = Wed Dec 15 11:37:51 UTC 2021
pZxid = 0x2000000020
cversion = 2
dataVersion = 3
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 9
numChildren = 2
```

Step 8. Remove the znode

To remove a specified znode permanently, use the `rmr` command and then write the path to this znode.

```
rmr /z_persistent
```

Now, if you try to get the data from the deleted znode or its children, you will see that these znodes do not exist anymore:

```
[zk: localhost:2181(CONNECTED) 26] get /z_persistent
Node does not exist: /z_persistent
[zk: localhost:2181(CONNECTED) 27] get /z_persistent/child1
Node does not exist: /z_persistent/child1
```

In conclusion, let’s check that ephemeral znodes are removed automatically after the client disconnection. To do this, run the `quit` command and start the new client session by running *zkCli.sh*. Then call the `ls /` command to get all children of the root znode. You will find out the following:

- The persistent znode `/z_persistent` is removed, as we have deleted it in the example above.
- The sequential znode `/z_sequential0000000007` is still available.
- The ephemeral znode `/z_ephemeral` has been automatically removed.

```
[zk: localhost:2181(CONNECTED) 0] ls /
```

The output:

```
[zookeeper, Arenadata.Hadoop-1.solr.server, z_sequential0000000007, hadoop-ha, hbase]
```

Logging in ZooKeeper

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Grep logs

Logging levels

Logging configuration

Overview

ZooKeeper records file system processes output and events in text logs. This information can be useful when diagnosing technical issues.

Each ZooKeeper component stores logs locally, on the host it occupies. The logs have the *.log* extension and are located in the */var/log/zookeeper/* directory. In the same directory, you can find *.out* files that contain component restart information.

The logs naming convention is `zookeeper-<user>-<component>-<host>.log`.

Where:

- `<user>` — user running ZooKeeper;
- `<component>` — component name, for example, ZooKeeper server;
- `<host>` — FQDN of the component host.

The ZooKeeper uses for logging SLF4J and Log4J frameworks.

Grep logs

You can search through the logs for a specific information, like error messages. To do this, connect to the host with the component whose logs you want to inspect, and use a `grep` command.

For example:

```
$ cat /var/log/zookeeper/zookeeper-zookeeper-cli-elenas-adh3.ru-central1.internal.log | grep -i -A3 -B1 error
```

This command searches for messages containing the word `error` in the CLI log, located on the `elenas-adh3.ru-central1.internal` host. The `i` option allows you to ignore case distinctions. The `-A3 -B1` options expand the output to one line before and three lines after the line containing the error.

The example output:

```
2023-11-28 12:39:02,986 [myid:] - INFO  [main:ClientCnxn@1681] - zookeeper.request.timeout value is 0. feature enabled=
2023-11-28 12:39:02,991 [myid:elenas-adh3.ru-central1.internal:2181] - INFO  [main-SendThread(elenas-adh3.ru-central1.
internal:2181):ClientCnxn$SendThread@1125] - Opening socket connection to server elenas-adh3.ru-central1.internal/127.0.0.1:2181.
Will not attempt to authenticate using SASL (unknown error)
2023-11-28 12:39:03,041 [myid:elenas-adh3.ru-central1.internal:2181] - INFO  [main-SendThread(elenas-adh3.ru-central1.
internal:2181):ClientCnxn$SendThread@972] - Socket connection established, initiating session, client: /127.0.0.1:53166, server:
elenas-adh3.ru-central1.internal/127.0.0.1:2181
2023-11-28 12:39:03,065 [myid:elenas-adh3.ru-central1.internal:2181] - INFO  [main-SendThread(elenas-adh3.ru-central1.
internal:2181):ClientCnxn$SendThread@1415] - Session establishment complete on server elenas-adh3.ru-central1.internal/127.0.0.
1:2181, sessionid = 0x1000010f7e50000, negotiated timeout = 30000
2023-11-28 12:39:03,855 [myid:] - INFO  [main:Environment@109] - Client environment:zookeeper.version=3.5.10-
c4d50586fc8f6fdbdc95b05e9b21b49ec0bd8566, built on 11/03/2022 08:23 GM
```

Logging levels

ZooKeeper supports the following logging levels (from least to most informative):

1. `FATAL` — indicates that an operation can’t continue execution and will terminate.
2. `ERROR` — notifies that a program is not working correctly or has stopped.
3. `WARN` — warns about potential problems. This doesn’t mean that a program is not working, but raises a concern.
4. `INFO` — informs regarding the program lifecycle or state.
5. `DEBUG` — prints debugging information about internal states of the program.
6. `TRACE` — prints messages tracing the execution flow of a program.

The Log4j loggers also accept logging level values: `OFF` — for switching off the logging, and `ALL` — for allowing all types of messages.

Enabling one level of logging will enable this level and all levels above it. For example, if you set the logging level to `WARN`, then only warnings, errors, and fatal messages would get into the log files, but not `INFO`, `DEBUG`, and `TRACE`.

Logging configuration

You can configure logging settings for ZooKeeper via the *log4j.properties* configuration file located on each ZooKeeper server host at *etc/zookeeper/conf/*.

The *log4j.properties* file contains logger settings, appender settings, and other logging properties. The parameters you set for a logger influence what a program reports to the system. The parameters you set for the appenders influence the information from loggers that ends up in the log files or console.

The configuration file begins with root logger default properties. This logger processes all messages in ZooKeeper since it’s the only logger configured. The default settings are: `INFO, CONSOLE, ROLLINGFILE`. This means that messages below the `INFO` level are discarded and the rest of the messages will end up in both console and log files.

The default appenders are:

- `CONSOLE` — responsible for the console output, e.g., command output information;
- `ROLLINGFILE` — responsible for updating log files instead of appending to a single log.

You can use appenders layout class to change the log messages format. The default pattern layout describes: message level, date, thread information, and caller information in addition to the message itself.

For more detailed information on Log4j architecture, see the Apache Logging Services documentation.

The default ZooKeeper log4j.properties file

```
# Define some default values that can be overridden by system properties
zookeeper.root.logger="{{ services.zookeeper.config['zookeeper_env_content']['ZOO_LOG4J_PROP'] }}"
zookeeper.console.threshold=INFO
zookeeper.log.dir="{{ services.zookeeper.config['zookeeper_env_content']['ZOO_LOG_DIR'] }}"
zookeeper.log.file=zookeeper.log
zookeeper.log.threshold=INFO
zookeeper.tracelog.dir="{{ services.zookeeper.config['zookeeper_env_content']['ZOO_LOG_DIR'] }}"
zookeeper.tracelog.file=zookeeper.log

#
# ZooKeeper Logging Configuration
#

# Format is "<default threshold> (, <appender>)+

# DEFAULT: console appender only
log4j.rootLogger=${zookeeper.root.logger}

# Example with rolling log file
#log4j.rootLogger=DEBUG, CONSOLE, ROLLINGFILE

# Example with rolling log file and tracing
#log4j.rootLogger=TRACE, CONSOLE, ROLLINGFILE, TRACEFILE

#
# Log INFO level and above messages to the console
#
log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.Threshold=${zookeeper.console.threshold}
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d{ISO8601} [myid:%X{myid}] - %-5p [%t:%C{1}@%L] - %m%n

#
# Add ROLLINGFILE to rootLogger to get log file output
#    Log DEBUG level and above messages to a log file
log4j.appender.ROLLINGFILE=org.apache.log4j.RollingFileAppender
log4j.appender.ROLLINGFILE.Threshold=${zookeeper.log.threshold}
log4j.appender.ROLLINGFILE.File=${zookeeper.log.dir}/${zookeeper.log.file}

# Max log file size of 10MB
log4j.appender.ROLLINGFILE.MaxFileSize=10MB
# uncomment the next line to limit number of backup files
#log4j.appender.ROLLINGFILE.MaxBackupIndex=10

log4j.appender.ROLLINGFILE.layout=org.apache.log4j.PatternLayout
log4j.appender.ROLLINGFILE.layout.ConversionPattern=%d{ISO8601} [myid:%X{myid}] - %-5p [%t:%C{1}@%L] - %m%n


#
# Add TRACEFILE to rootLogger to get log file output
#    Log DEBUG level and above messages to a log file
log4j.appender.TRACEFILE=org.apache.log4j.FileAppender
log4j.appender.TRACEFILE.Threshold=TRACE
log4j.appender.TRACEFILE.File=${zookeeper.tracelog.dir}/${zookeeper.tracelog.file}

log4j.appender.TRACEFILE.layout=org.apache.log4j.PatternLayout
### Notice we are including log4j's NDC here (%x)
log4j.appender.TRACEFILE.layout.ConversionPattern=%d{ISO8601} [myid:%X{myid}] - %-5p [%t:%C{1}@%L][%x] - %m%n
```

To edit *log4j.properties* on all hosts at once, use ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at ZooKeeper.
3. Select ***Components*** and click on ***Zookeeper Server***.
4. Toggle the ***Show advanced*** option and find the ***log4j.properties template*** field.
5. Make the necessary changes to the default file.
6. Confirm changes to ZooKeeper configuration by clicking ***Save***.
7. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

NOTE

The *log4j.properties* file contains default values for logging parameters and they can be overwritten by ZooKeeper daemons on restart. If it happens, you can set the parameters using environmental variables in the *zookeeper-env.sh* file.

To change logging properties in the *zookeeper-env.sh* file via ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at ZooKeeper.
3. Toggle the ***Show advanced*** option and find the ***zookeeper-env.sh*** field.
4. Select the required parameter for example, ***ZOO_LOG4J_PROP***, and make the necessary changes.
5. Confirm changes to ZooKeeper configuration by clicking ***Save***.
6. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

Configure ZooKeeper

Elena Kostyuchenko

Contents

To Table of Contents

- Configuration files
  - zoo.cfg
  - zookeeper-env.sh
  - log4j.properties
- Change configuration via ADCM
- External quorum
- Recommendations

Configuration files

ZooKeeper has three main configuration files stored in the *etc/zookeeper/conf/* directory. You can edit them manually, by connecting to the ZooKeeper component host and using a text editor, or via the Arenadata Cluster Manager (ADCM).

When editing manually, make sure you updated the configuration file on all hosts with ZooKeeper components.

> NOTE
>
> ZooKeeper supports dynamic reconfiguration. This feature enables changing the configuration without restarting the server, but it might interfere with the ADCM.

This article doesn't cover the advanced configuration options. For more information about them, see the ZooKeeper Administrator's Guide.

zoo.cfg

The *zoo.cfg* configuration file contains the main ZooKeeper settings. If ZooKeeper has been installed via ADCM, it already contains the minimal configuration parameters. If ZooKeeper has been installed manually, you have to configure the following parameters for a minimal configuration:

- `clientPort` — the port for client connections to ZooKeeper. For SSL connections, configure the `secureClientPort` property. `clientPort` specifies the port for regular connections, while the `secureClientPort` property specifies the port for SSL connections. When both are specified, it enables the mixed-mode.
- `dataDir` — the location for ZooKeeper to store the in-memory database snapshots and, unless specified otherwise, the transaction log of updates to the database.
- `tickTime` — the length of a single tick, which is the basic time unit used by ZooKeeper, as measured in milliseconds. It is used to regulate heartbeats, and timeouts. For example, the minimum session timeout will be two ticks.

See more details about main *zoo.cfg* properties in the ***Main*** section of ZooKeeper settings. For information on other properties, see the ***zoo.cfg*** section.

zookeeper-env.sh

The *zookeeper-env.sh* file is used to define configuration parameters related to a ZooKeeper operating environment, such as `Java_HOME`.

You can also configure properties of other configuration files by setting environmental variables in *zookeeper-env.sh*.

For more information on *zookeeper-env.sh* contents, see the ***zookeeper-env.sh*** section of ZooKeeper settings.

log4j.properties

The *log4j.properties* configuration file contains logging settings. For more information on how to configure it, see the Logging in ZooKeeper article.

Change configuration via ADCM

To configure ZooKeeper using ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at ZooKeeper.
3. Find the desired property on the service configuration page or navigate to the ***Components*** page. Additional properties can be revealed by toggling the ***Show advanced*** option.
4. Make the necessary changes and click ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

External quorum

You can use an external ZooKeeper quorum for highly available services in your cluster.

To achieve that, specify external ZooKeeper hosts' FQDNs in a service configuration using custom configuration values in ADCM.

For example, here's how to integrate a highly available HBase with an external ZooKeeper quorum in ADCM:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click on ***HBase***.
3. Select the ***Custom hbase-site.xml*** parameter, specify `hbase.zookeeper.quorum` as the field name, and specify the FQDNs of external ZooKeeper quorum hosts as field values.
4. Confirm changes by clicking ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.

The default `hbase.zookeeper.quorum` value will be rewritten by the custom one on all hosts.

Recommendations

Here's a list of things to avoid and some additional configuration options that can improve your cluster's performance:

- When deploying many applications interfacing to a single ZooKeeper server, you might want to apply an optional `chroot` suffix to the connection string. This will run the client commands while interpreting all paths relative to this root (similar to the Unix `chroot` command). The example connection string: `127.0.0.1:4545/app/a`.
- To avoid the fsync warnings in the ZooKeeper log, set up a dedicated disk for the `dataLogDir` directory that is separate from the `dataDir` directory. Set the `dataLogDir` parameter in the *zoo.cfg* file.
- For a better performance, put the transaction log (`dataDir`) on a dedicated device. If the transaction log is placed on a busy device, it will adversely affect performance.
- Keep the servers list consistent between the clients and the ZooKeeper servers.

ZooKeeper service management via ADCM

Elena Kostyuchenko

To Table of Contents

The ADCM UI provides actions to manage the ZooKeeper service and its components. For information on how to run service actions, refer to ADH service actions.

The actions available for the ZooKeeper service are listed in the table below.

For more information about the service and its components, see the ZooKeeper article.

Action	Description
Add/Remove components	Running this action opens the component-host mapping interface where you can add, remove, and distribute ZooKeeper components
Check	Runs service-specific tests to check the health of the service and its components
Start	Starts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied on the service startup. Otherwise, the service starts without applying configurations from ADCM
Stop	Stops the service
Remove	Removes the service from the cluster. This action should be used to remove already installed services. Whereas the 🗑 control can be used to remove a non-mapped service (a service which components have not been distributed among cluster hosts)
Restart	Restarts the service. When you run this action, the option ***Apply configs from ADCM*** is available. If it is set to `true`, all service configurations defined in ADCM settings will be applied during the service restart. Otherwise, the service restarts without applying configurations from ADCM. The service supports the ***Rolling restart*** option (enabled by default) that allows you to restart ZooKeeper Server components one by one rather than all at once. This feature helps to avoid the service downtime and provides more flexible restart configuration. The ***Rolling restart*** option has the following parameters: ▪ ***delay*** — the delay in seconds between restarting individual ZooKeeper Server components. ▪ ***max_failed_instances_number*** — the maximum number of ZooKeeper Server instances allowed to fail during restart. Upon reaching this value, the ***Restart*** ADCM action fails. ▪ ***health_checks*** — indicates whether to perform health checks of the restarted ZooKeeper Servers.

Arenadata Hyperwave cluster actions

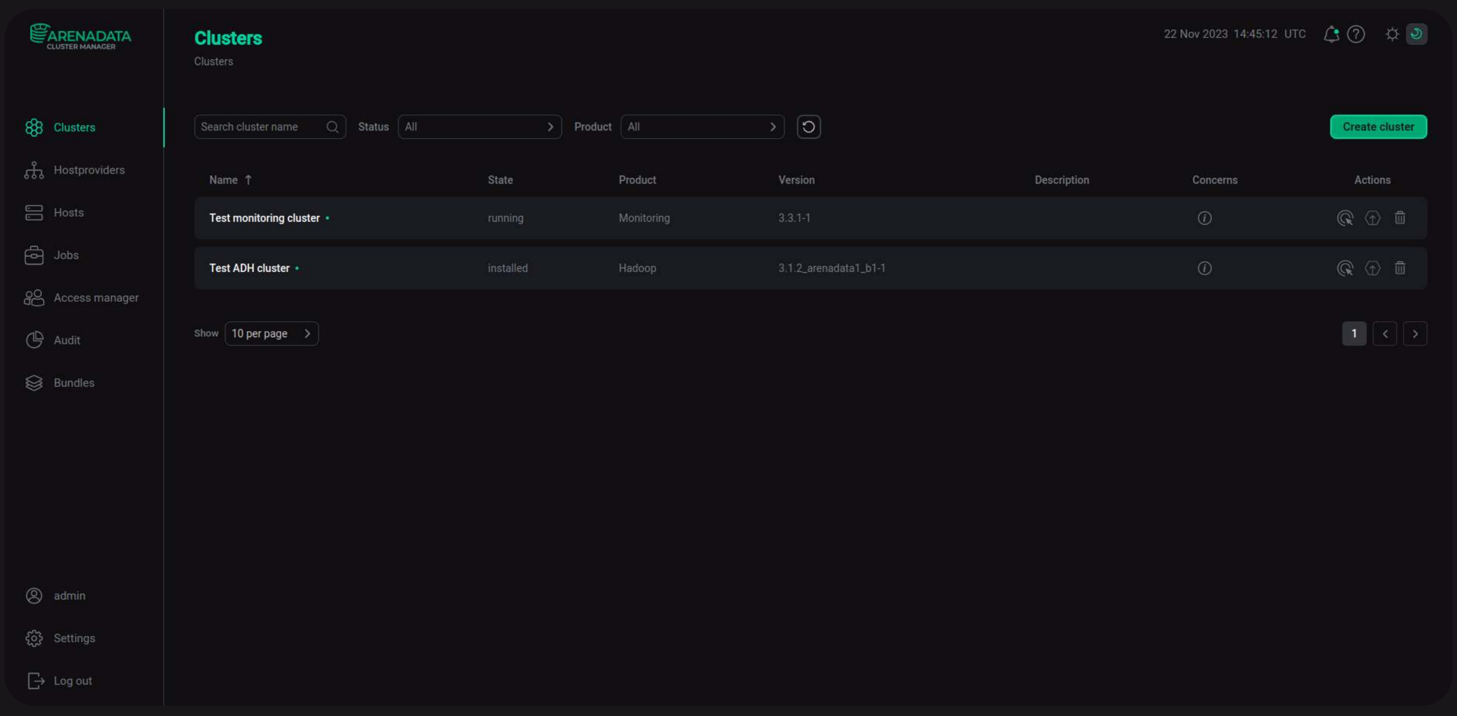
Eugenia Kuzina

Contents

To Table of Contents

Actions

This article describes the actions available for the ADH cluster in the ADCM UI. You can find cluster actions on the ***Clusters*** page.



Clusters

Clusters

22 Nov 2023 14:45:12 UTC

Status: All

Product: All

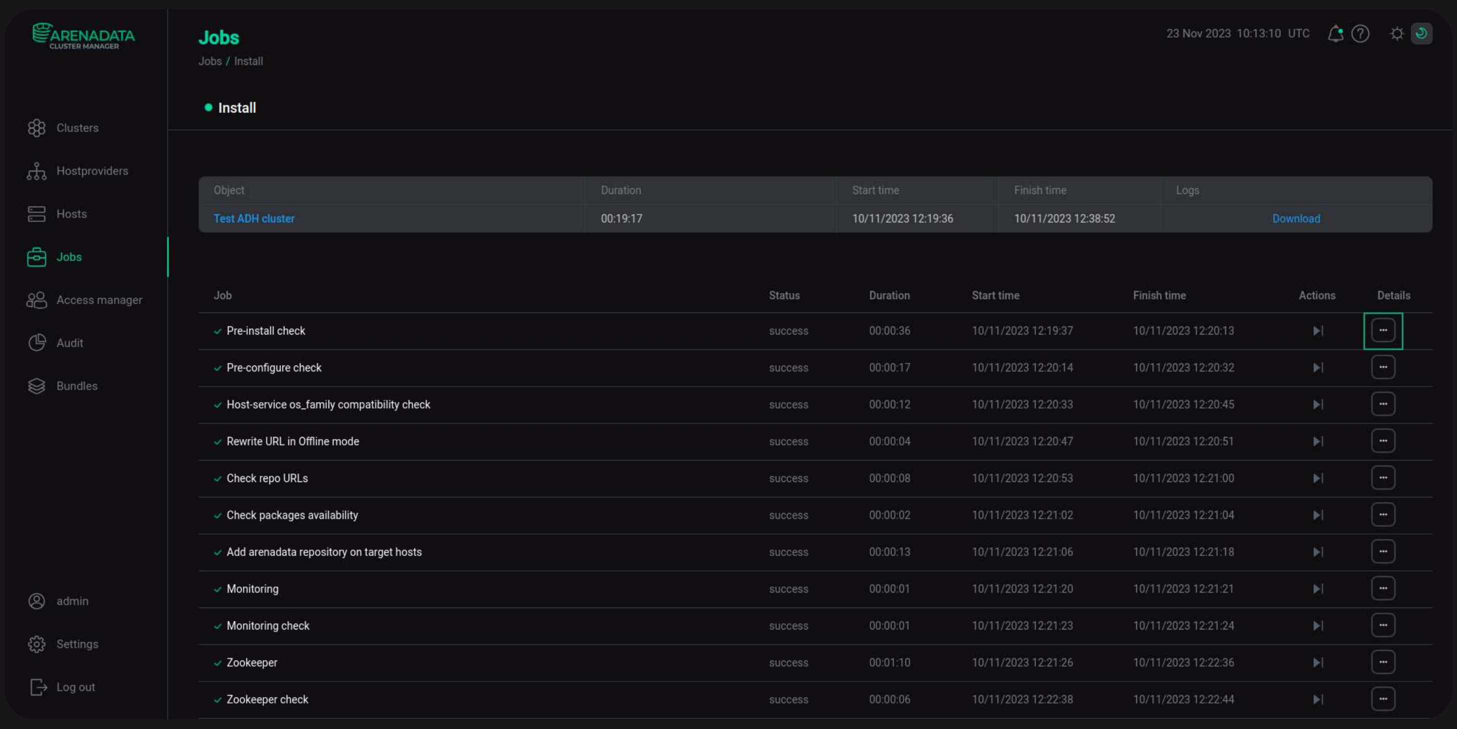
Name ↑	State	Product	Version	Description	Concerns	Actions
Test monitoring cluster	running	Monitoring	3.3.1-1		ⓘ	
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1		ⓘ	

The Clusters page

The ***Clusters*** page displays a table with the following columns:

- ***Name*** is a name specified during the cluster creation.
- ***State*** is the current cluster state (for example, `created`, `installed`, `running`).
- ***Product*** — product name.
- ***Version*** — version of a bundle used to install the cluster.
- ***Description*** — description specified during the cluster creation.
- ***Concerns*** — displays an icon that indicates a state of running processes in the cluster. The icon ⓘ shows that the cluster works correctly. If critical errors are detected in the configuration or a job is running, the icon image is changed to ⓘ. When you hover over it, a pop-up window appears showing a description with a link that you can follow to perform the required configuration or get details.
- ***Actions*** — shows icons for managing a cluster:
  - — opens the drop-down list that offers actions to manage the cluster.
  - — indicates whether a new version of a bundle is available and allows you to upgrade the cluster. See Cluster upgrade.
  - — deletes information about the cluster from ADCM. This action does not remove ADH services or make any changes to the hosts that belong to the cluster.

You can use icons described above to perform cluster actions. When an action starts, ADCM shows its execution process and result on the ***Jobs*** page. On this page, click a job name to open a page with detailed information on an individual job. If the job has inner steps, click the ellipsis in the ***Details*** column for additional information on the current step.



Jobs

Jobs / Install

23 Nov 2023 10:13:10 UTC

● Install

Object	Duration	Start time	Finish time	Logs
Test ADH cluster	00:19:17	10/11/2023 12:19:36	10/11/2023 12:38:52	Download

Job	Status	Duration	Start time	Finish time	Actions	Details
✓ Pre-install check	success	00:00:36	10/11/2023 12:19:37	10/11/2023 12:20:13	▶|	…
✓ Pre-configure check	success	00:00:17	10/11/2023 12:20:14	10/11/2023 12:20:32	▶|	…
✓ Host-service os_family compatibility check	success	00:00:12	10/11/2023 12:20:33	10/11/2023 12:20:45	▶|	…
✓ Rewrite URL in Offline mode	success	00:00:04	10/11/2023 12:20:47	10/11/2023 12:20:51	▶|	…
✓ Check repo URLs	success	00:00:08	10/11/2023 12:20:53	10/11/2023 12:21:00	▶|	…
✓ Check packages availability	success	00:00:02	10/11/2023 12:21:02	10/11/2023 12:21:04	▶|	…
✓ Add arenadata repository on target hosts	success	00:00:13	10/11/2023 12:21:06	10/11/2023 12:21:18	▶|	…
✓ Monitoring	success	00:00:01	10/11/2023 12:21:20	10/11/2023 12:21:21	▶|	…
✓ Monitoring check	success	00:00:01	10/11/2023 12:21:23	10/11/2023 12:21:24	▶|	…
✓ Zookeeper	success	00:01:10	10/11/2023 12:21:26	10/11/2023 12:22:36	▶|	…
✓ Zookeeper check	success	00:00:06	10/11/2023 12:22:38	10/11/2023 12:22:44	▶|	…

The "Jobs" page

Actions

The following actions are available for the ADH cluster:

- Check
- Manage credential encryption
- Manage Kerberos
- Manage SSL
- Prepare hosts
- Reinstall status-checker
- Start
- Stop

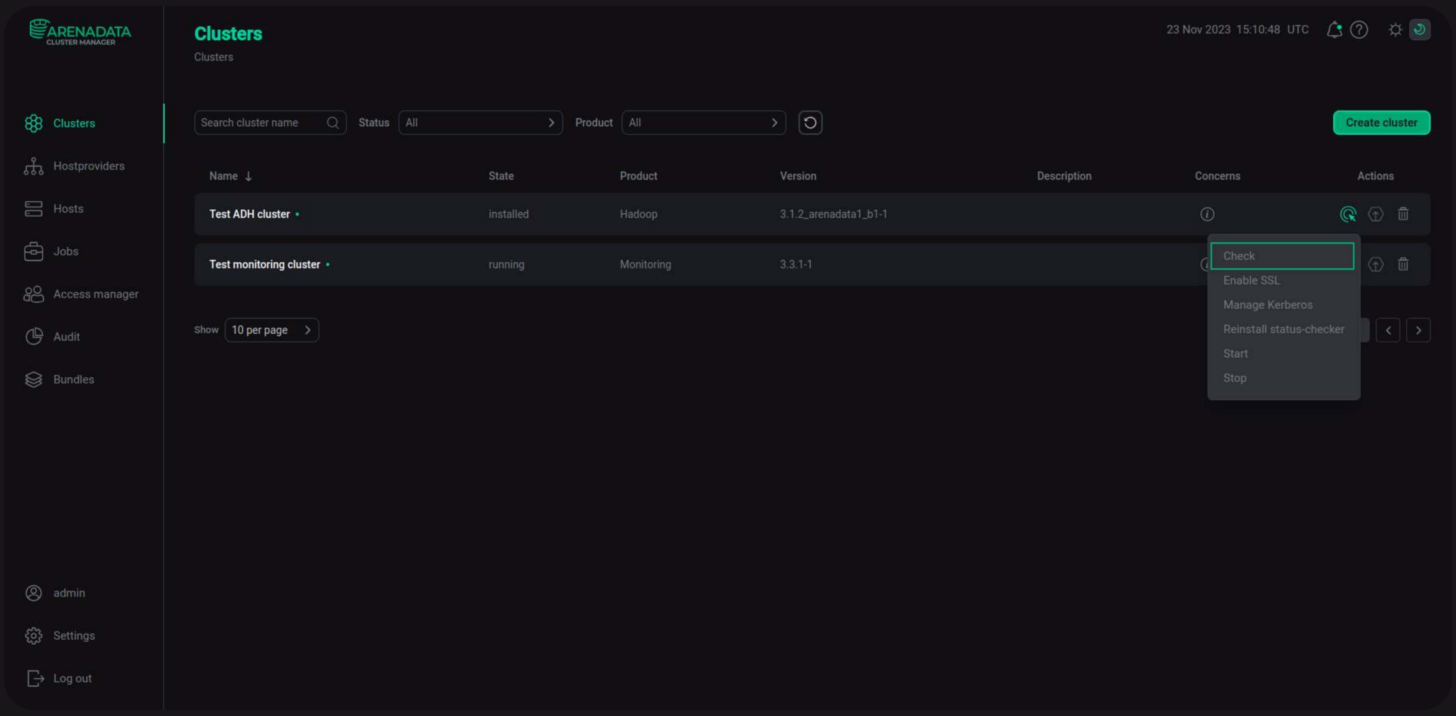
Check

Eugenia Kuzina

To Table of Contents

The ***Check*** action verifies whether ADH services work correctly. It sequentially runs service-specific tests for each service to check the health of the service and its components.

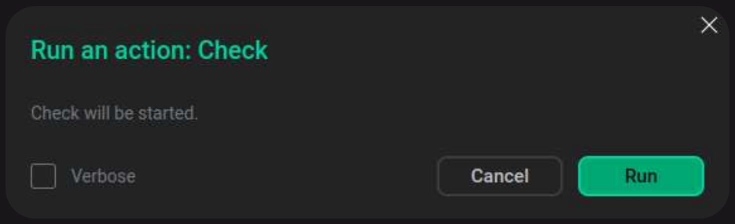
To run an action, go to ***Clusters*** page, click in the ***Actions*** column, and select the action from the drop-down list that appears.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters
23 Nov 2023 15:10:48 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Check
Enable SSL
Manage Kerberos
Reinstall status-checker
Start
Stop
Show
10 per page

Name ↓	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			
Test monitoring cluster	running	Monitoring	3.3.1-1			

Execute the "Check" action

When you choose an action, ADCM displays a dialog window to confirm the choice. In this dialog window, you can select the ***Verbose*** checkbox to see additional execution details.


Run an action: Check
☐ Verbose
Cancel
Run

Check will be started.

The "Run an action" window

You can find execution process and result of the action on the ***Jobs*** page. On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs.

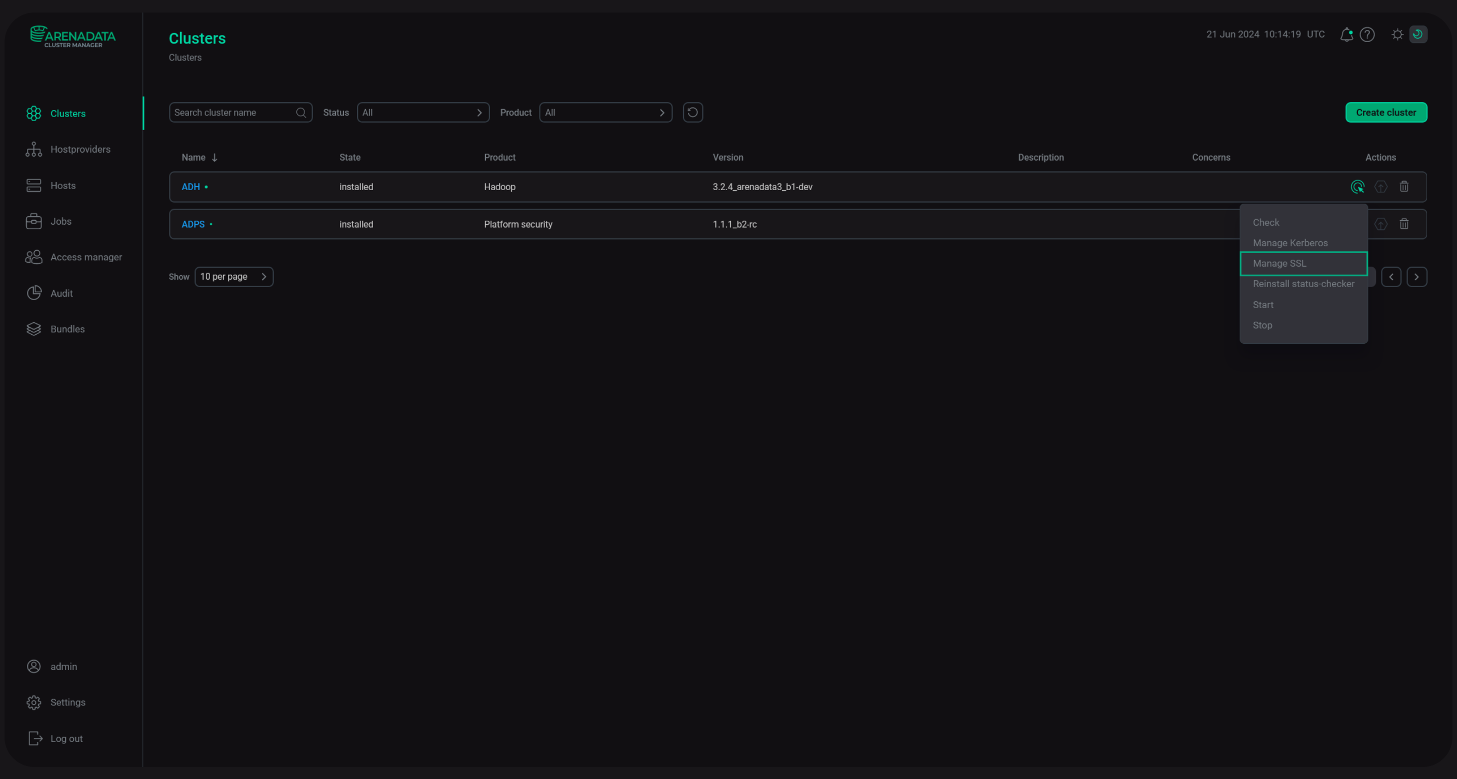
Manage SSL

Sergei Tikhomirov, Eugenia Kuzina

To Table of Contents

The ***Manage SSL*** action allows you to control the SSL encryption of all services in the ADH cluster. For additional information on SSL settings, see Requirements for SSL certificates.

To run an action, go to ***Clusters*** page, click in the ***Actions*** column, and select the action from the drop-down list that appears.



Name	State	Product	Version	Description	Concerns	Actions
ADH	installed	Hadoop	3.2.4_arenadata3_b1-dev			
ADPS	installed	Platform security	1.1.1_b2-rc			

ARENADATA
CLUSTER MANAGER
Clusters
Clusters
21 Jun 2024 10:14:19 UTC
Search cluster name
Status All
Product All
Create cluster
Check
Manage Kerberos
Manage SSL
Reinstall status-checker
Start
Stop
Show 10 per page
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

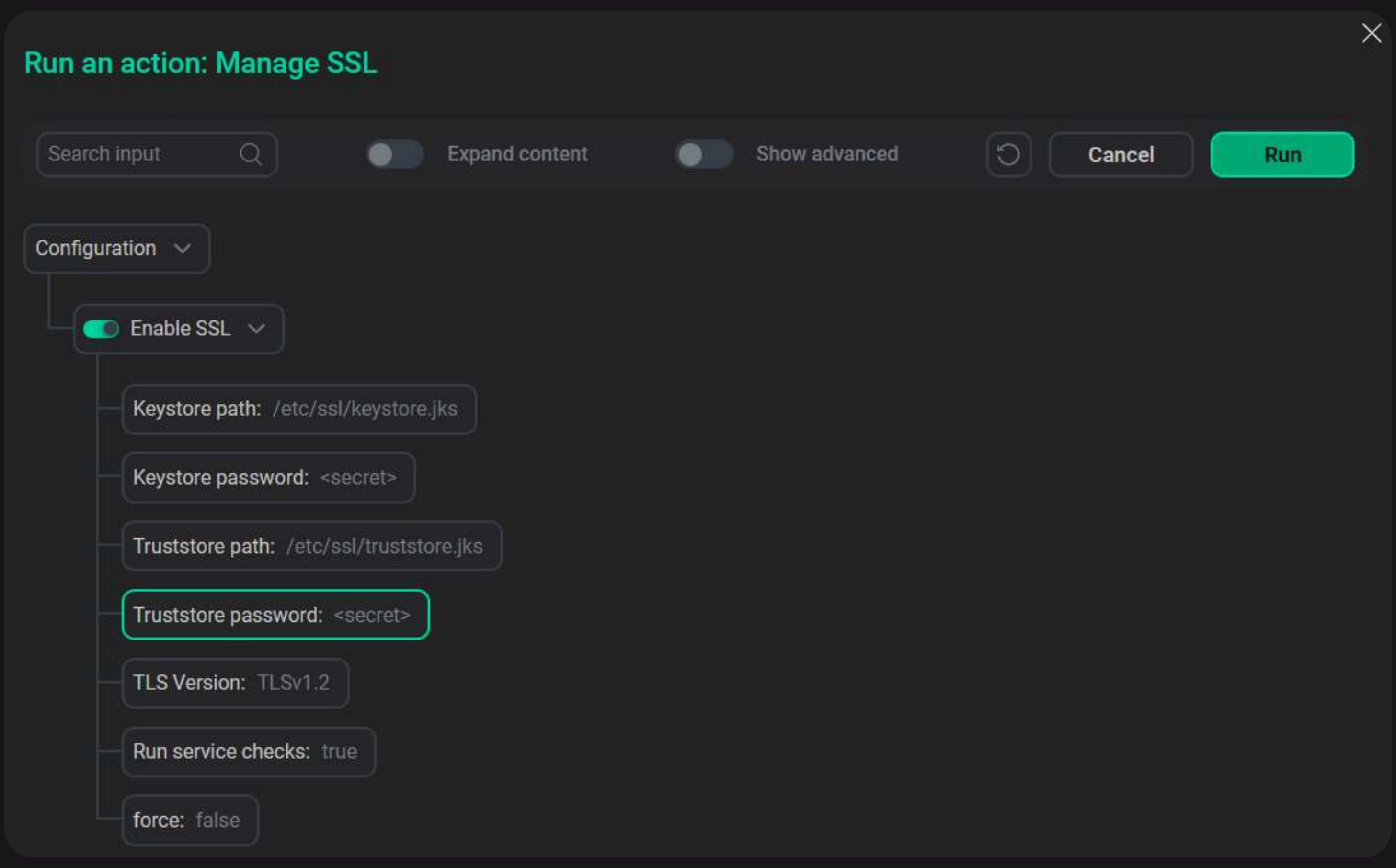
Execute the "Manage SSL" action

When you choose an action, ADCM displays a dialog window where you can specify SSL settings listed in the table below.

SSL settings

Parameter	Description	Default value
Keystore path	A path to the keystore file	/etc/ssl/keystore.jks
Keystore password	A password for the keystore file	—
Truststore path	A path to the truststore file	/etc/ssl/truststore.jks
Truststore password	A password for the truststore file	—
TLS Version	Version of the TLS protocol. See Set TLS version for an ADH cluster	TLSv1.2
Run service checks	Runs the service checks after the setting change	True
force	Defines whether current SSL parameters should be overwritten with the specified values	False

To enable SSL, run the action with the ***Enable SSL*** parameter group active and the parameters filled in.



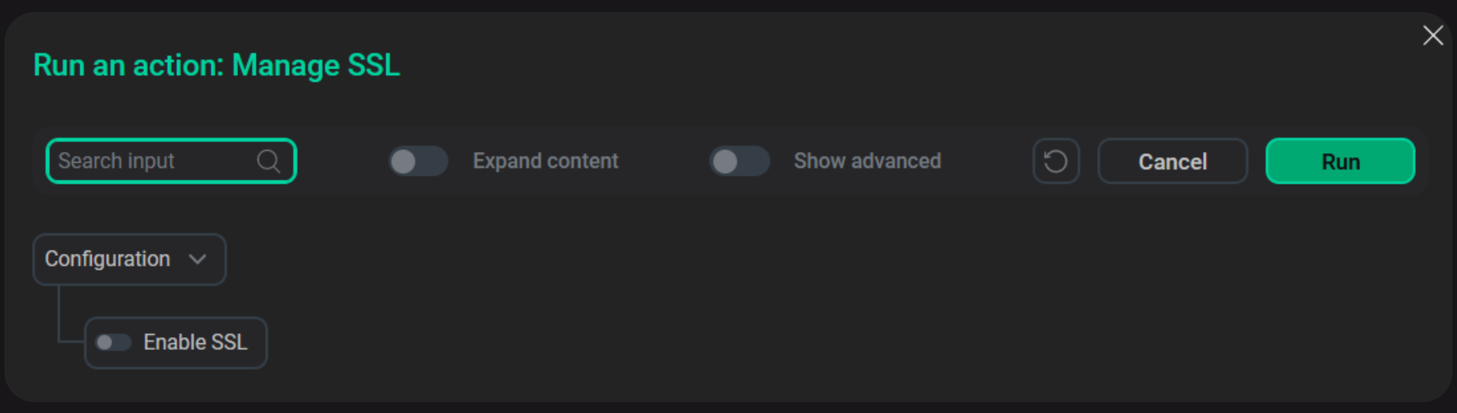
Keystore path: /etc/ssl/keystore.jks
Keystore password: <secret>
Truststore path: /etc/ssl/truststore.jks
Truststore password: <secret>
TLS Version: TLSv1.2
Run service checks: true
force: false

Run an action: Manage SSL
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Enable SSL

Enable SSL with the "Manage SSL" action

After you specify the parameters, click ***Run***. In the action confirmation window that appears, you can select the ***Verbose*** checkbox to see additional execution details.

To disable SSL, run the ***Manage SSL*** action with the ***Enable SSL*** parameter group inactive.



Run an action: Manage SSL
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Enable SSL

Disable SSL with the "Manage SSL" action

During the execution of the ***Manage SSL*** action, ADCM stops all services one by one, reconfigures them to enable or disable SSL encryption, then starts the services and checks them.

You can find execution process and result of the action on the ***Jobs*** page. On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs.

IMPORTANT

If the Ranger Solr plugin is enabled, make sure to manually restart Solr after enabling/disabling of SSL.

Manage Credential Encryption

Sergei Tikhomirov

Contents

To Table of Contents

Enable encryption with a cluster action

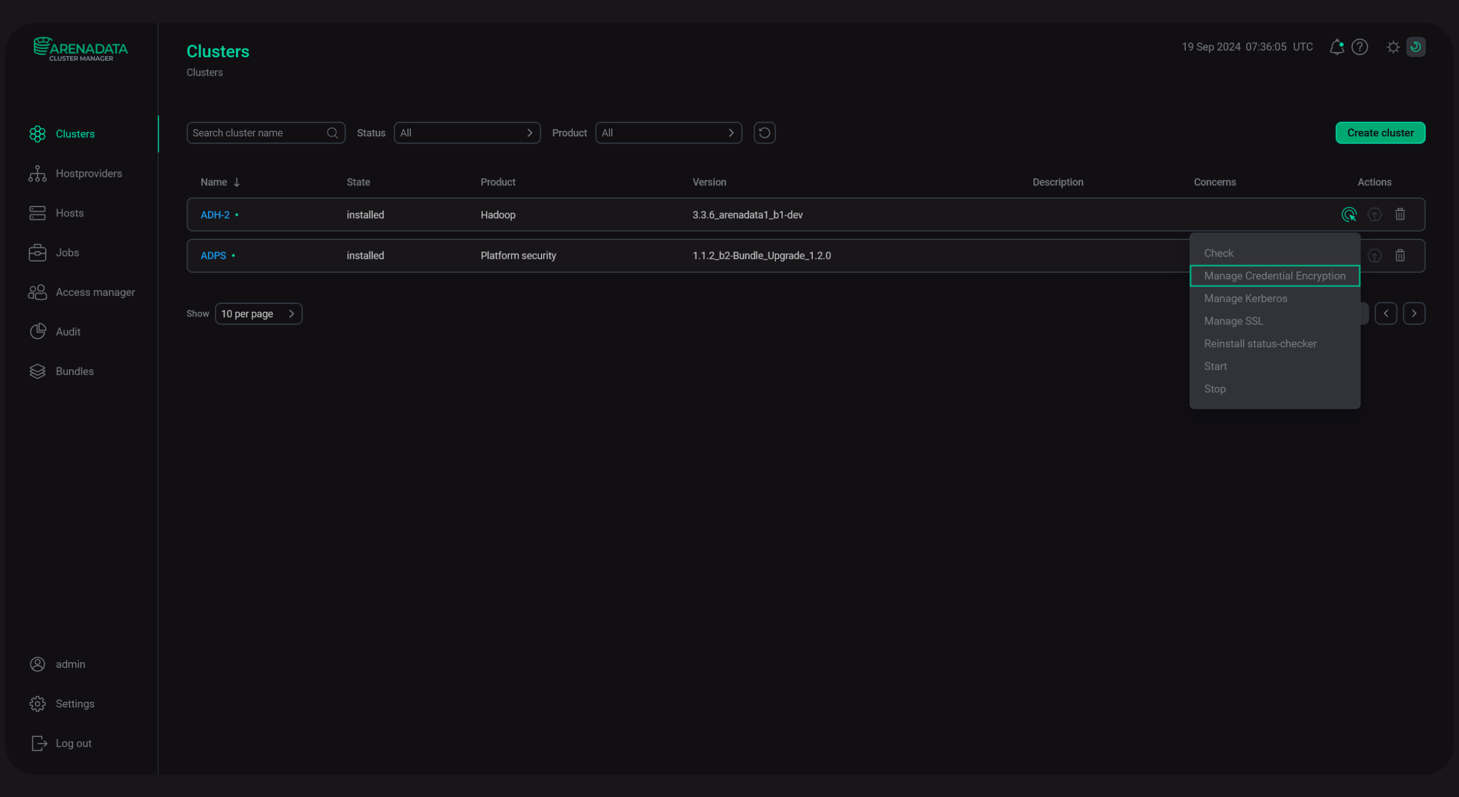
Enable encryption for a service

The ***Manage Credential Encryption*** action allows you to control ADH services configuration encryption. The encryption is implemented via a Hadoop or KMS credential provider that supports storing passwords in the JCEKS (an encrypted container with passwords, i.e. credential store) instead of configuration files. By default, the credential store is protected by a password that is stored in the environment for each encrypted service, but you can keep that password in a file if such option is available for a service.

Enable encryption with a cluster action

To enable the credential encryption, follow the steps below:

1. Head to the ***Clusters*** page and run the ***Manage Credential Encryption*** action for your ADH cluster.



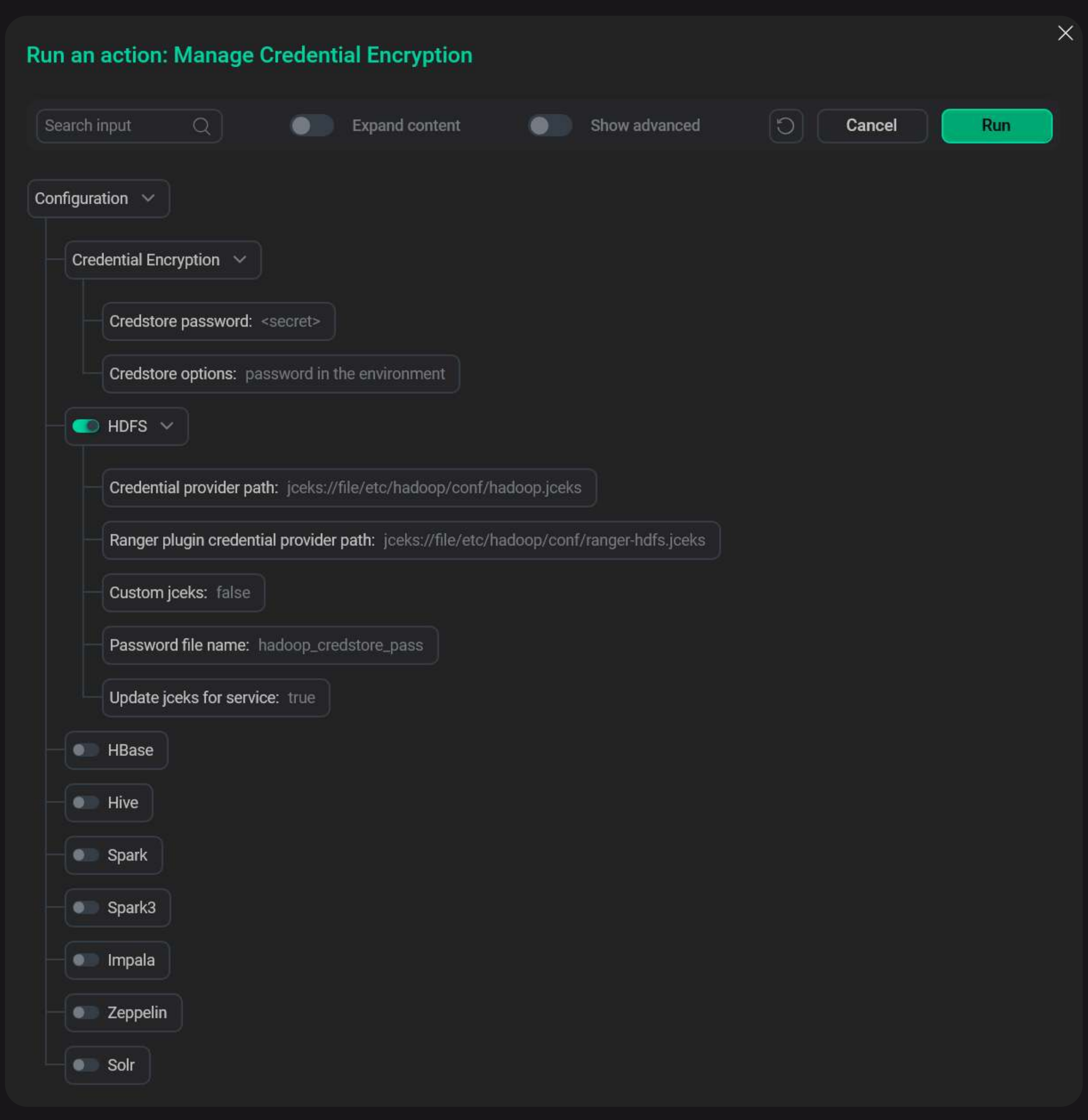
ARENADATA CLUSTER MANAGER
Clusters
19 Sep 2024 07:36:05 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Check
Manage Credential Encryption
Manage Kerberos
Manage SSL
Reinstall status-checker
Start
Stop
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
ADH-2	installed	Hadoop	3.3.6_arenadata1_b1-dev			
ADPS	installed	Platform security	1.1.2_b2-Bundle_Upgrade_1.2.0			

Run the "Manage Credential Encryption" action

2. In the opened window, set the encryption settings:

   - ***Credstore password*** — the encryption provider password.
   - ***Credstore options*** — the way to store the encryption provider password. Three options are available: `no password`, `password-file`, and `password in the environment`.



Run an action: Manage Credential Encryption
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Credential Encryption
HDFS
HBase
Hive
Spark
Spark3
Impala
Zeppelin
Solr

Credstore password: <secret>
Credstore options: password in the environment
Credential provider path: jceks://file/etc/hadoop/conf/hadoop.jceks
Ranger plugin credential provider path: jceks://file/etc/hadoop/conf/ranger-hdfs.jceks
Custom jceks: false
Password file name: hadoop_credstore_pass
Update jceks for service: true

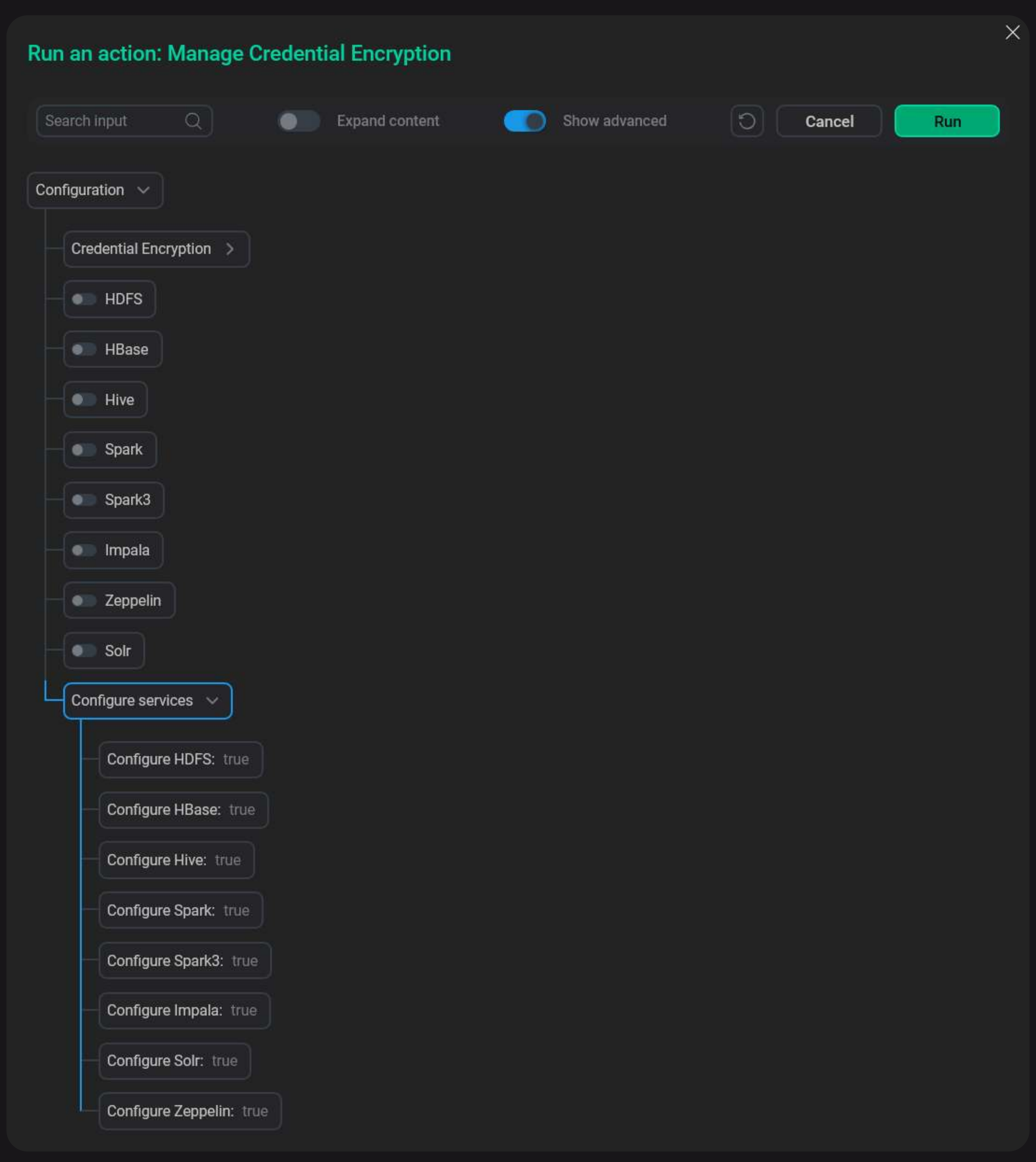
The credential encryption parameters

3. In the same window, select the services for which you want to activate encryption and provide the necessary data:

   - ***Credential provider path*** — a credential provider path for creating the JCEKS files with secrets. Can be a local or HDFS path.
   - ***Ranger plugin credential provider path*** — a Ranger plugin credential provider path for creating the JCEKS files with secrets.
   - ***Custom jceks*** — if enabled, allows you to use your own credential store. Otherwise — generates JCEKS files automatically.
   - ***Password file name*** — a filename in service's classpath for storing the password file. It is accounted for only if the `password-file` value is selected in the ***Credstore options*** parameter.
   - ***Update jceks for service*** — defines whether to recreate the JCEKS files for this service. Takes effect only if the JCEKS files were previously created. Activate if you have changed the SSL (keystore, truststore) or LDAP passwords, the DB connection password, or any other password in a service configuration.

     You can enable the encryption for HDFS (with YARN), HBase, Hive, Impala, Kyuubi, Spark/Spark3, Zeppelin, and Solr. The parameter list above is similar for each service — some services don't require all the parameters.

4. The action also has an advanced mode, enabling which allows you to reconfigure services and recreate JCEKS files for them. Here, you can unselect the unaffected services so that they will not be reconfigured (which makes sense since their configuration hasn't been changed).

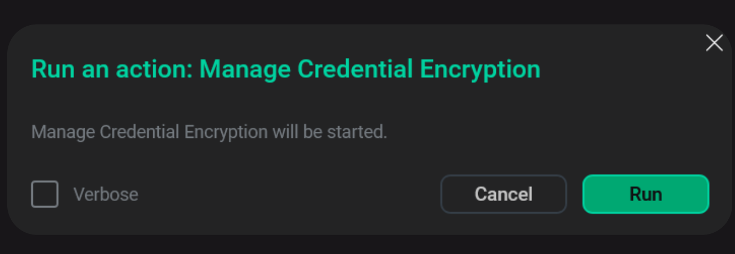


Run an action: Manage Credential Encryption
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Credential Encryption
HDFS
HBase
Hive
Spark
Spark3
Impala
Zeppelin
Solr
Configure services

Configure HDFS: true
Configure HBase: true
Configure Hive: true
Configure Spark: true
Configure Spark3: true
Configure Impala: true
Configure Solr: true
Configure Zeppelin: true

Advanced mode

5. Click ***Run*** and confirm the action in a pop-up window.



Run an action: Manage Credential Encryption
Verbose
Cancel
Run

Manage Credential Encryption will be started.

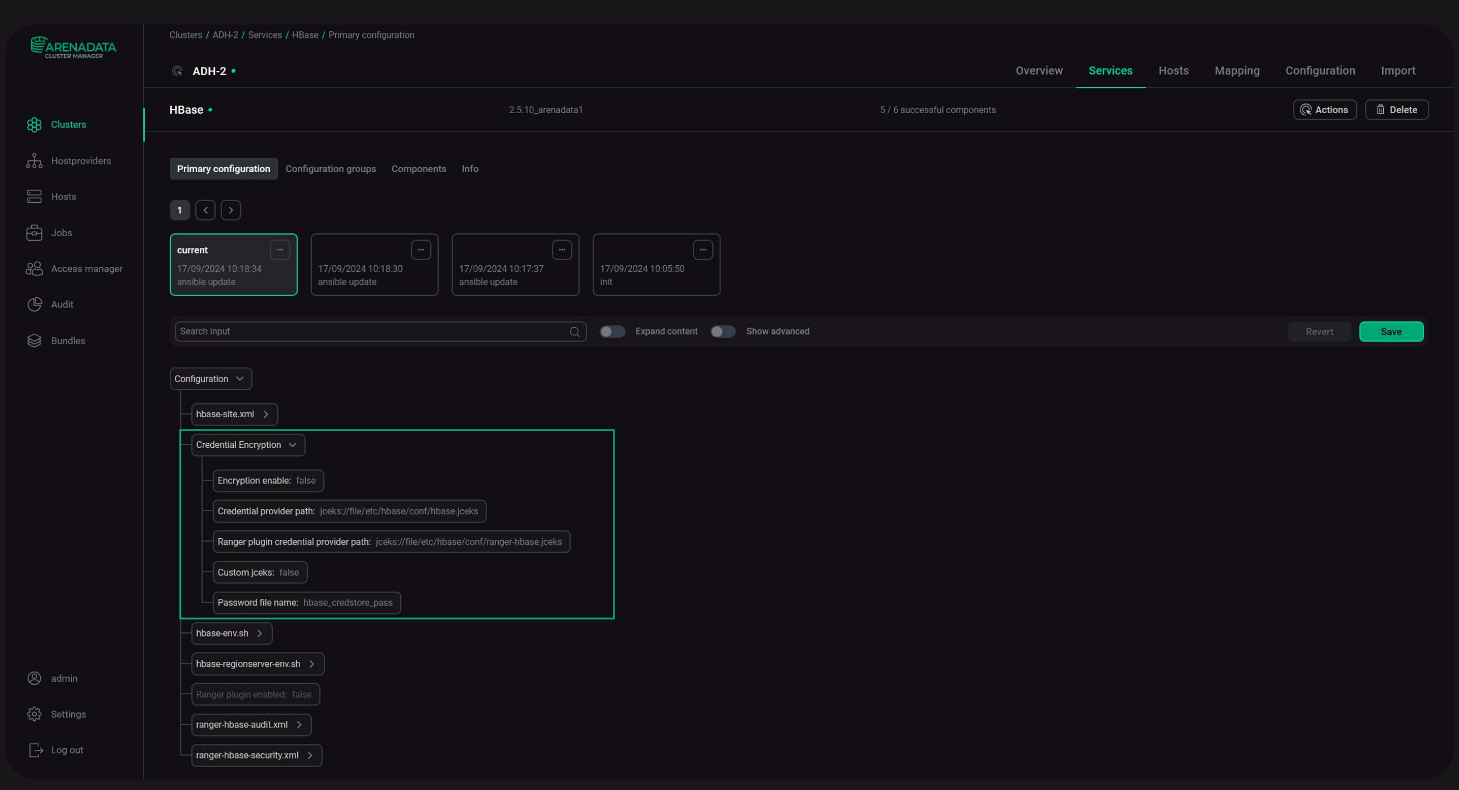
Confirmation window

Enable encryption for a service

There is an alternate way to enable the credential encryption for a service that supports it. Instead of launching the ***Manage Credential Encryption*** action, you can do the following:

1. Head to the ***Services*** tab of your ADH cluster and select a service for which you want to enable encryption.

2. Expand the ***Credential Encryption*** parameter group and configure the parameters shown below.



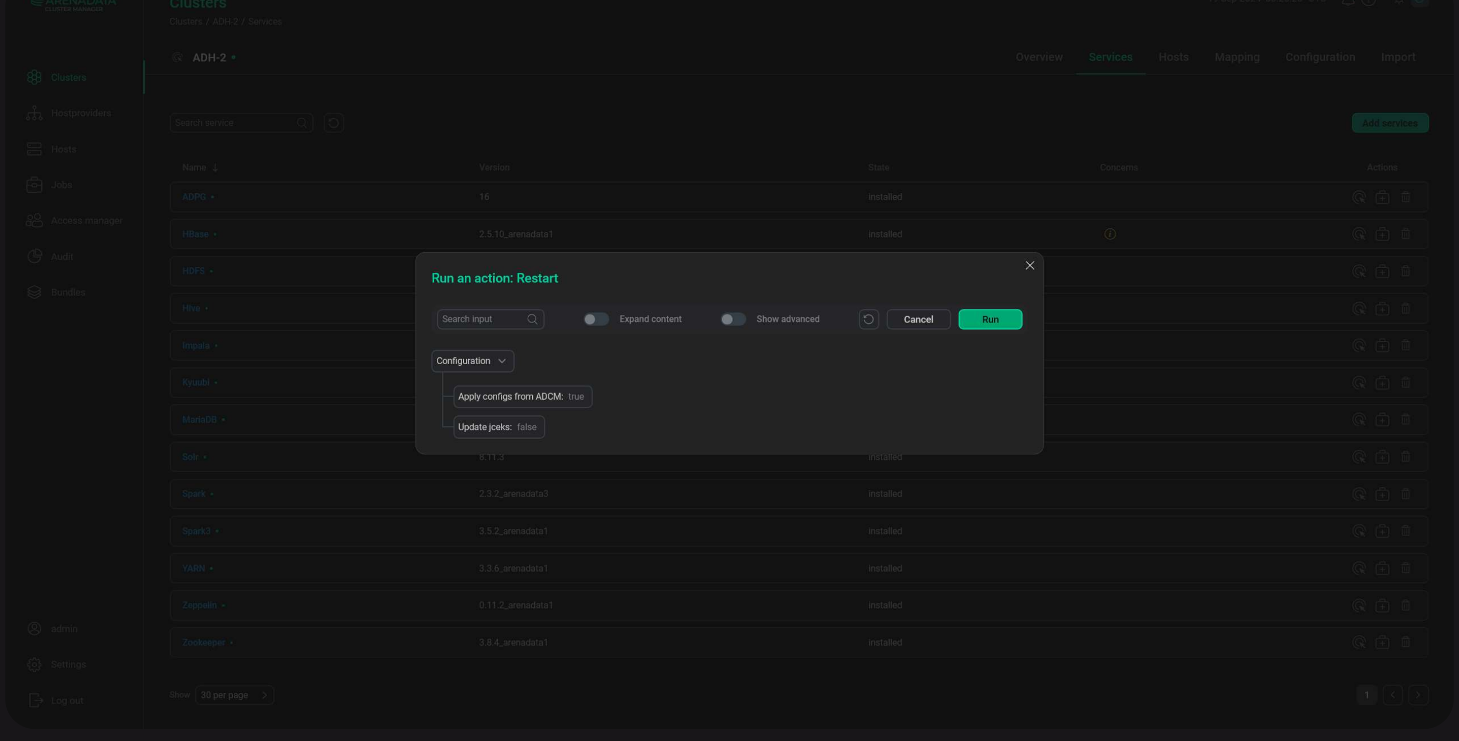
ARENADATA CLUSTER MANAGER
Clusters / ADH-2 / Services / HBase / Primary configuration
ADH-2
Overview
Services
Hosts
Mapping
Configuration
Import
HBase
2.5.10_arenadata1
5 / 6 successful components
Actions
Delete
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
Components
Info
current
17/09/2024 10:18:34
ansible update
17/09/2024 10:18:30
ansible update
17/09/2024 10:17:37
ansible update
17/09/2024 10:05:50
init
Search input
Expand content
Show advanced
Revert
Save
Configuration
hbase-site.xml
Credential Encryption
hbase-env.sh
hbase-regionserver-env.sh
ranger-hbase-audit.xml
ranger-hbase-security.xml

Encryption enable: false
Credential provider path: jceks://file/etc/hbase/conf/hbase.jceks
Ranger plugin credential provider path: jceks://file/etc/hbase/conf/ranger-hbase.jceks
Custom jceks: false
Password file name: hbase_credstore_pass
Ranger plugin enabled: false

Service configuration

The parameter list is similar to the one in the ***Manage Credential Encryption*** action, but there is one new flag parameter — ***Encryption enable***. To enable credential encryption, you need to activate it and save the service configuration.

3. Restart the service. The ***Update jceks*** parameter appears in the action menu only after enabling the ***Encryption enable*** configuration parameter. It is needed during the first encryption activation via the ***Restart*** action (as opposed to the ***Manage encryption*** action, which creates a JCEKS archive with passwords automatically with no additional actions required).



Run an action: Restart
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Apply configs from ADCM: true
Update jceks: false

The "Update jceks" parameter

To update the JCEKS files (when a service password or provider settings have changed), activate it and click ***Run***.

Manage Kerberos

Sergei Tikhomirov

Contents

To Table of Contents

Manage Kerberos action

Custom kerberization settings

Custom keytab recommendations

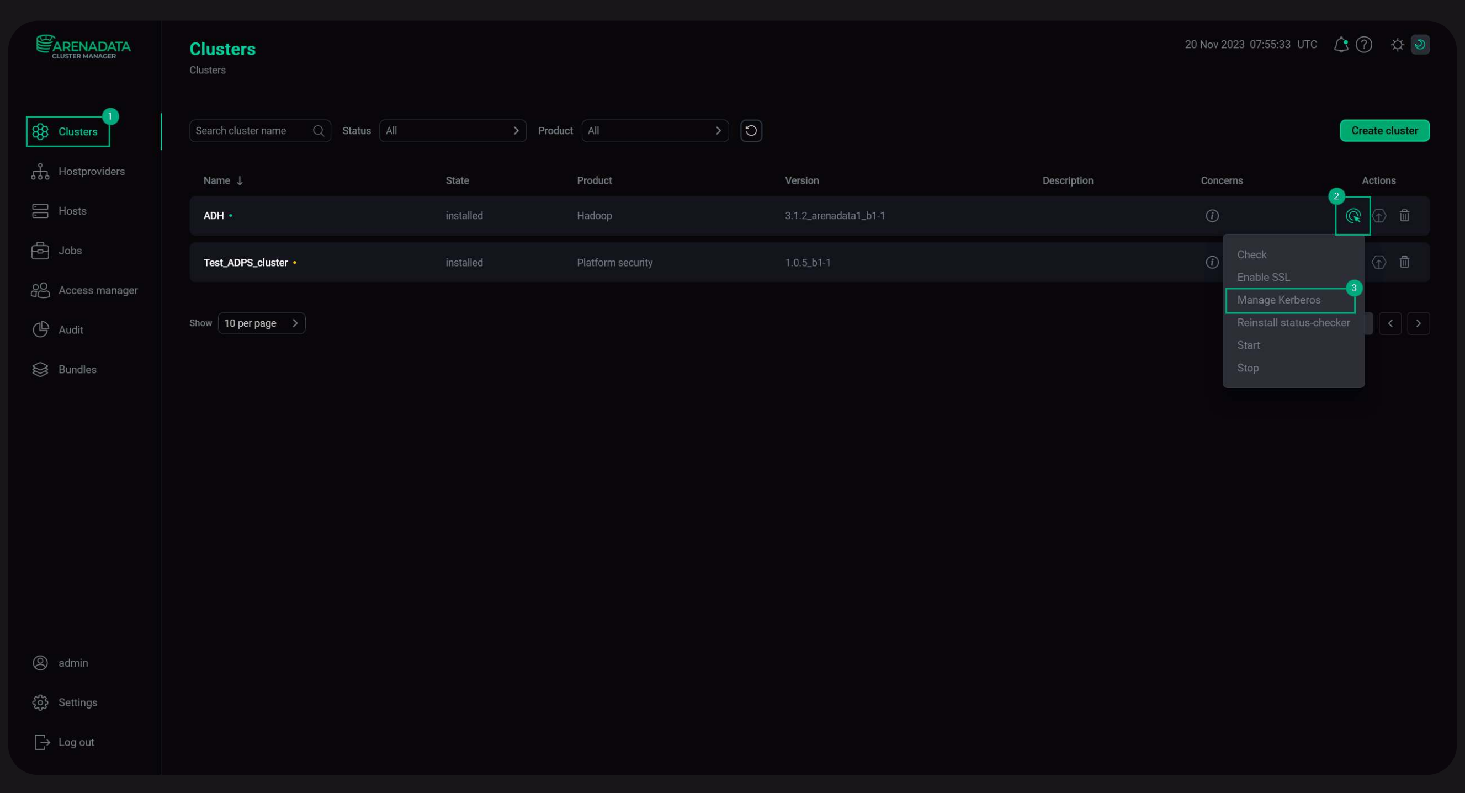
Manage Kerberos action

The ***Manage Kerberos*** action encapsulates enabling, reconfiguring, and disabling of Kerberos. To run it, you need an installed ADH cluster (see Get started with Arenadata Hyperwave) and one or more KDCs (Key Distribution Center). Before you proceed to managing Kerberos, it is recommended to read the Kerberos overview article about kerberization requirements.

Among other things, ***Manage Kerberos*** is designed to allow the following scenarios:

- Enabling Kerberos without administrator credentials using the existing client, principals and/or keytabs.
- Quickly changing the cluster state from kerberized to non-kerberized in case kerberization fails in the beginning (e.g. bad kadmin credentials) without having to reconfigure services, remove keytabs and principals.
- Postponed service reconfiguration — configure Kerberos at the moment and reconfigure service later.

To run the action in the ADCM web UI, go to the ***Clusters*** page. Select an installed and prepared ADH cluster, and choose the ***Manage Kerberos*** action.



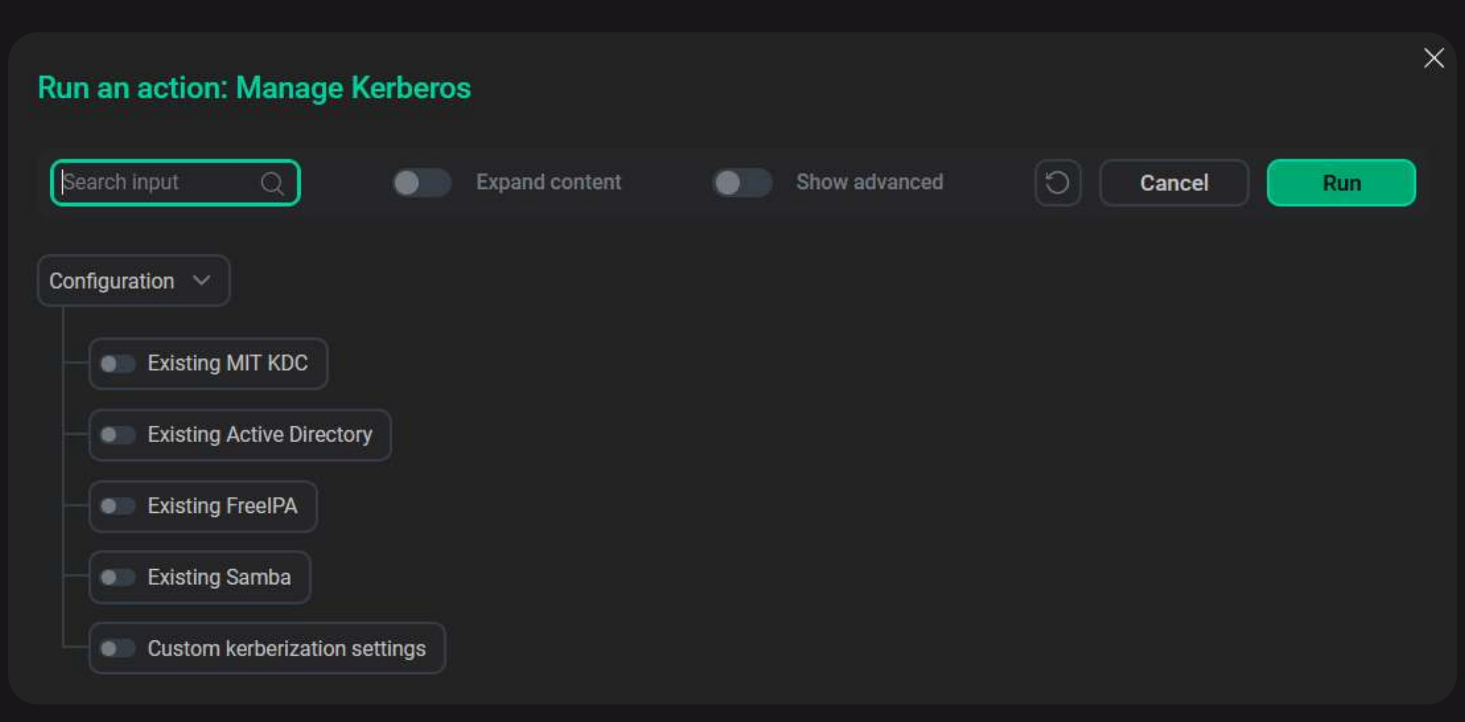
ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
20 Nov 2023 07:55:33 UTC
Search cluster name
Status All
Product All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
ADH
installed
Hadoop
3.1.2_arenadata1_b1-1
Test_ADPS_cluster
installed
Platform security
1.0.5_b1-1
Check
Enable SSL
Manage Kerberos
Reinstall status-checker
Start
Stop
Show 10 per page
1
2
3

Manage Kerberos

The pop-up window suggests several options to run the action:

- Using an existing MIT KDC.
- Using an existing MS Active Directory.
- Using an existing FreeIPA.
- Using an existing Samba.
- Using the Custom kerberization settings option.

To correctly launch the action you should choose just one out of the suggested KDC types: ***Existing MIT KDC***, ***Existing Active Directory***, ***Existing FreeIPA***, or ***Existing Samba***. Each of these options can be combined with the ***Custom kerberization settings*** option.



Run an action: Manage Kerberos
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
Existing Active Directory
Existing FreeIPA
Existing Samba
Custom kerberization settings

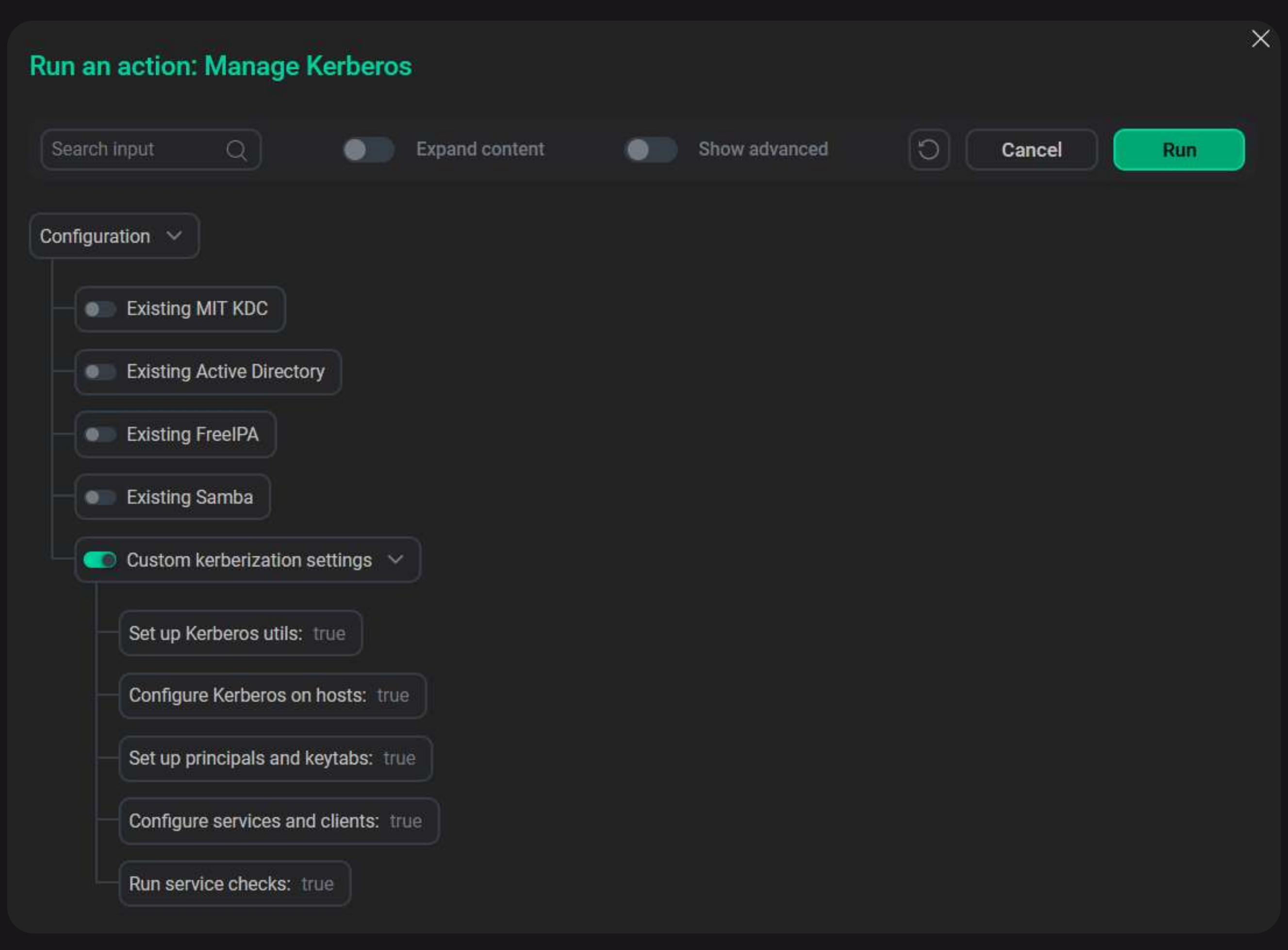
Ways to manage Kerberos

IMPORTANT

Running the action with one KDC type enabled will trigger Kerberos activation.

Custom kerberization settings

The ***Custom kerberization settings*** option allows the user to choose kerberization steps, for example, creation of principals and keytabs.



Run an action: Manage Kerberos
Search input
Expand content
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
Existing Active Directory
Existing FreeIPA
Existing Samba
Custom kerberization settings

Set up Kerberos utils: true
Configure Kerberos on hosts: true
Set up principals and keytabs: true
Configure services and clients: true
Run service checks: true

Custom kerberization settings parameters

Custom kerberization settings parameter description

Parameter	Description	Default value
Set up Kerberos utils	Enables installation or removal of Kerberos clients and utils. Affects the ***Expand*** and ***Install*** actions	True
Configure Kerberos on hosts	Enables cluster configuration, including *krb5.conf*, *ldap.conf*	True
Set up principals and keytabs	Enables creation, recreation, or removal of principals and keytabs. Passwords for principals are generated randomly before keytab creation. Affects the ***Expand*** and ***Install*** actions. ADCM bundle will set up owner and permissions for keytabs only if this checkbox is selected in the cluster configuration. In case of absence of admin permissions, a customer should provide the prepared keytabs with correctly set owner and permissions (see Custom keytab recommendations)	True
Configure services and clients	Enables updating of services and clients configuration	True
Run service checks	Enables service check runs	True

Custom keytab recommendations

Below is the table with recommendations for owners, groups, and permissions for keytabs.

Keytab recommendations

Component short name	Keytab owner	Keytab group	Permissions
zookeeper	zookeeper	zookeeper	600
hdfs-datanode	hdfs	hadoop	600
hdfs-namenode	hdfs	hadoop	600
hdfs-journalnode	hdfs	hadoop	600
hdfs-secondarynamenode	hdfs	hadoop	600
hdfs-zkfc	hdfs	hadoop	600
HTTP	hdfs	hadoop	640
httpfs	httpfs	httpfs	600
hdfs	hdfs	hadoop	640
yarn	yarn	hadoop	640
yarn-resourcemanager	yarn	hadoop	600
yarn-nodemanager	yarn	hadoop	600
yarn-timelineserver	yarn	hadoop	600
mapreduce-historyserver	mapred	hadoop	600
hbase	hbase	hadoop	640
hbase-phoenix_queryserver	phoenix	phoenix	600
hbase-thrift2	hbase	hbase	600
hive	hive	hive	640
solr	solr	solr	600
spark	spark	spark	640
livy	livy	hadoop	600
airflow	airflow	airflow	600
flink	flink	flink	600
zeppelin	zeppelin	zeppelin	600
impala	impala	impala	600

Prepare hosts

Elena Kostyuchenko

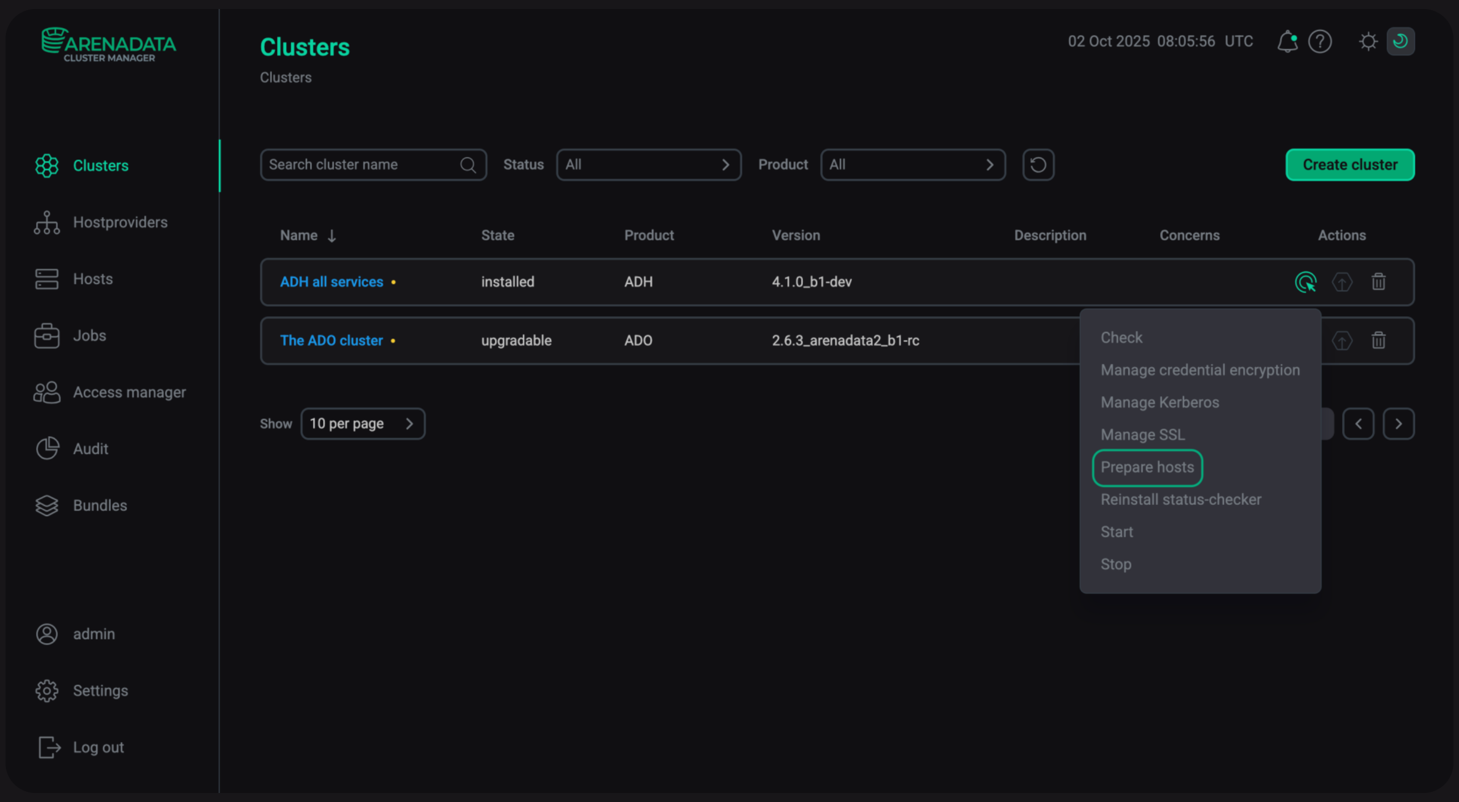
To Table of Contents

The ***Prepare hosts*** action installs all packages declared in the cluster bundle on hosts, thus preparing new hosts before adding service components. By installing the necessary dependencies first, it helps to avoid an extensive lock period for the cluster when adding new services. During this action, no components are assigned or started on new hosts.

When adding new hosts to a cluster, it is recommended to run the ***Prepare hosts*** action before installing any service components. If components are installed without preparing the hosts, the same operations will be performed during component installation but the cluster will be locked longer.

You can activate the ***Prepare hosts*** action via cluster action on the ***Clusters*** page or as a host group action. To run an action, go to ***Clusters*** page, click in the ***Actions*** column, and select the action from the drop-down list that appears.

Starting with ADCM 2.8.0, hosts can also be prepared via the ***Actions*** menu on the ***Hosts*** page.

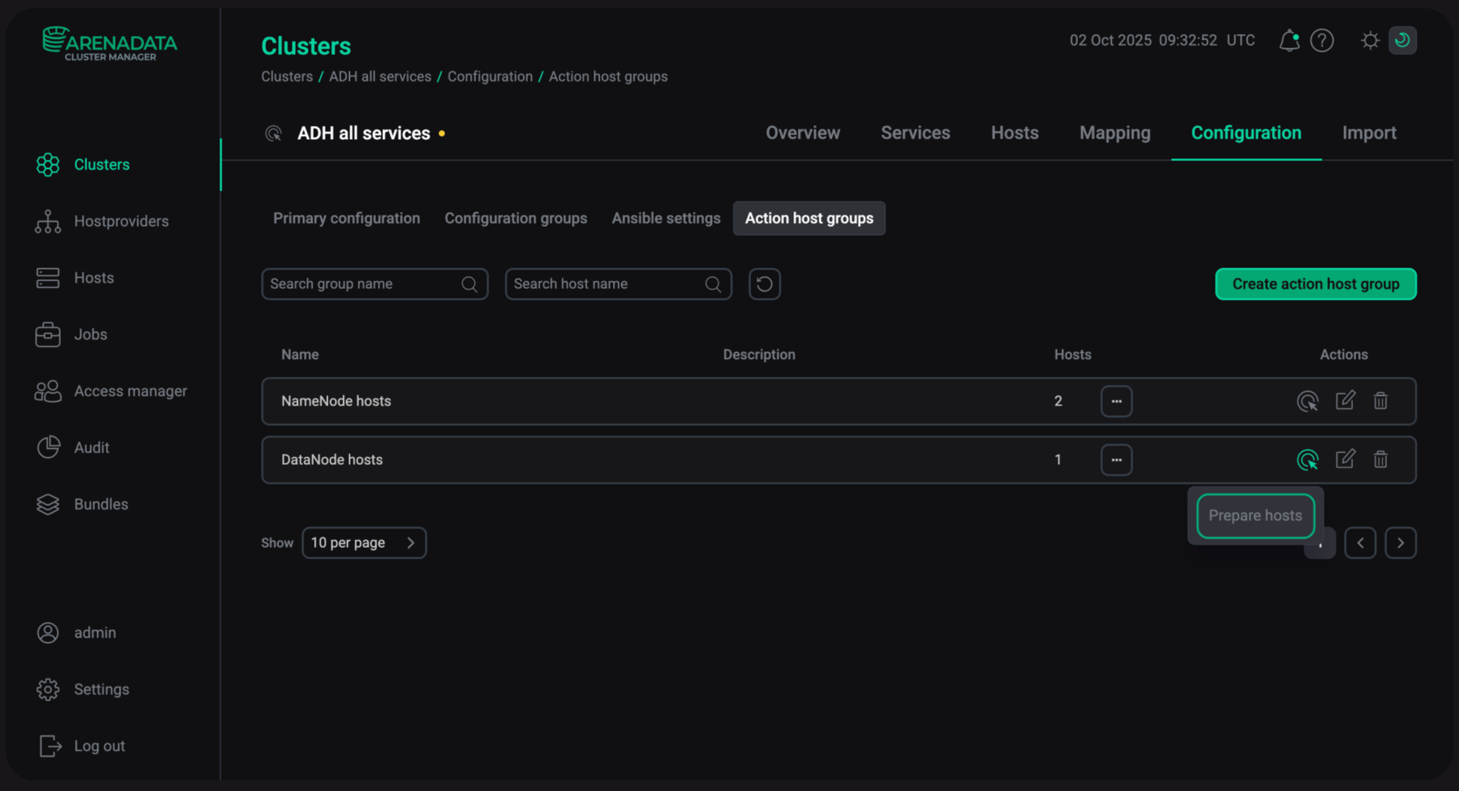

ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
02 Oct 2025 08:05:56 UTC
Search cluster name
Status
All
Product
All
Create cluster
Check
Manage credential encryption
Manage Kerberos
Manage SSL
Prepare hosts
Reinstall status-checker
Start
Stop
Show
10 per page

Name	State	Product	Version	Description	Concerns	Actions
ADH all services	installed	ADH	4.1.0_b1-dev			
The ADO cluster	upgradable	ADO	2.6.3_arenadata2_b1-rc			

Execute the "Prepare hosts" action via Clusters page

To prepare a group of hosts:

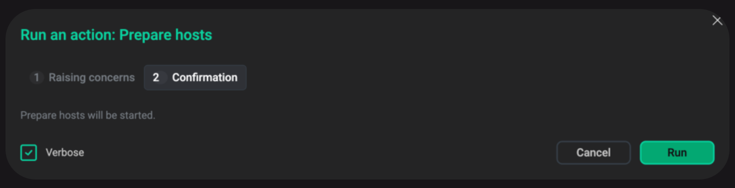
1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Configuration*** tab and select ***Action host groups***.
3. For the selected group of hosts, click the ***Actions*** drop-down menu and select ***Prepare hosts***.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / ADH all services / Configuration / Action host groups
02 Oct 2025 09:32:52 UTC
ADH all services
Overview
Services
Hosts
Mapping
Configuration
Import
Primary configuration
Configuration groups
Ansible settings
Action host groups
Search group name
Search host name
Create action host group
Prepare hosts
Show
10 per page

Name	Description	Hosts	Actions
NameNode hosts		2	
DataNode hosts		1	

Execute the "Prepare hosts" action via host groups

When you choose an action, ADCM displays a dialog window to confirm the choice. In this dialog window, you can select the ***Verbose*** checkbox to see additional execution details.


Run an action: Prepare hosts
1 Raising concerns
2 Confirmation
Cancel
Run

Prepare hosts will be started.

☑ Verbose

The "Run an action" window

You can find execution process and result of the action on the ***Jobs*** page. On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs.

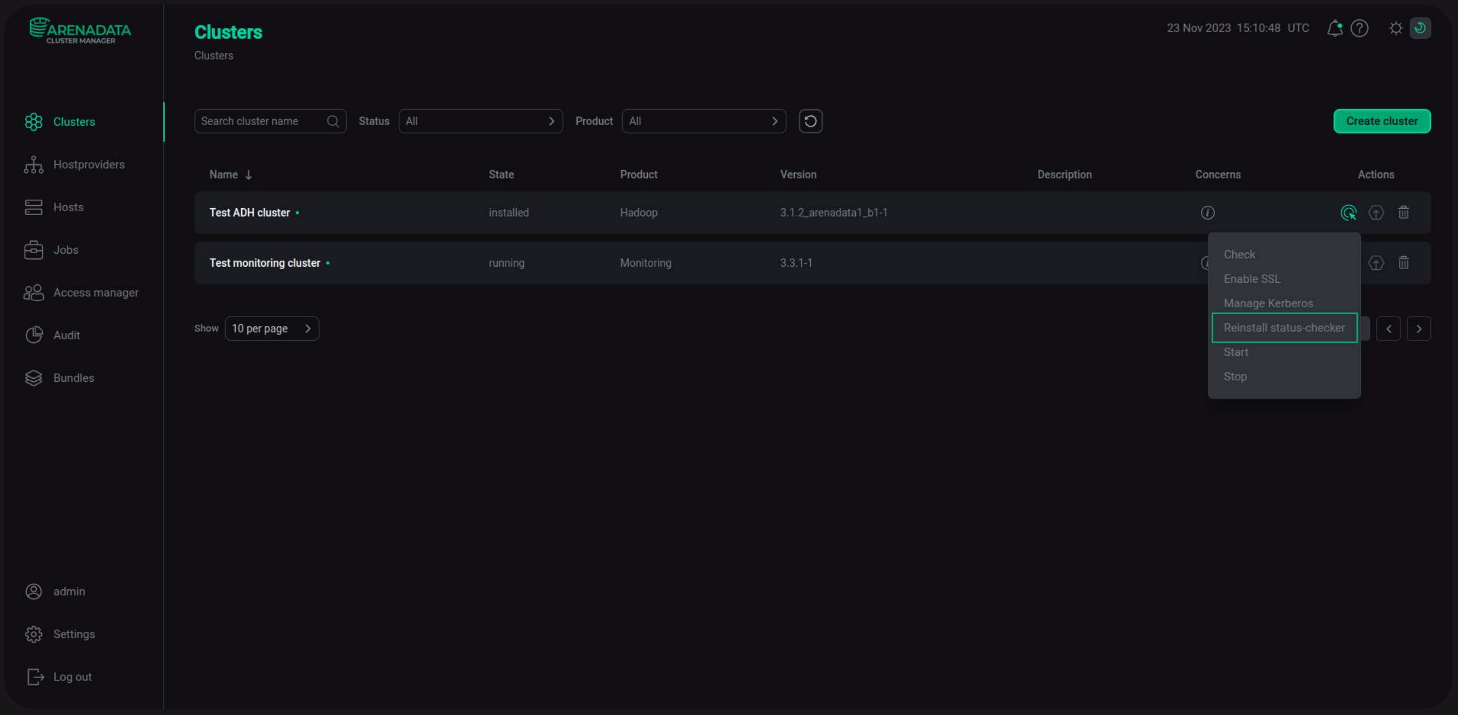
Reinstall status-checker

Eugenia Kuzina

To Table of Contents

The ***Reinstall status-checker*** action reinstalls status-checker for all services one by one.

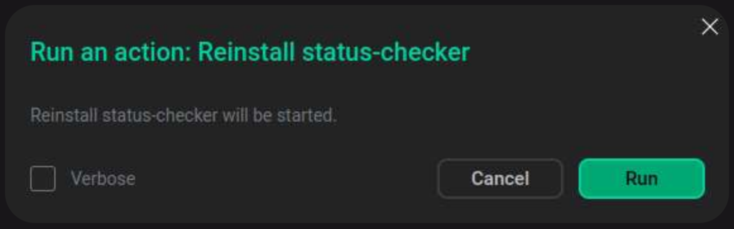
To run an action, go to ***Clusters*** page, click in the ***Actions*** column, and select the action from the drop-down list that appears.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters
23 Nov 2023 15:10:48 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Check
Enable SSL
Manage Kerberos
Reinstall status-checker
Start
Stop
Show
10 per page

Name ↓	State	Product	Version	Description	Concerns	Actions
Test ADH cluster •	installed	Hadoop	3.1.2_arenadata1_b1-1			
Test monitoring cluster •	running	Monitoring	3.3.1-1			

Execute the "Reinstall status-checker" action

When you choose an action, ADCM displays a dialog window to confirm the choice. In this dialog window, you can select the ***Verbose*** checkbox to see additional execution details.


Run an action: Reinstall status-checker
☐ Verbose
Cancel
Run

Reinstall status-checker will be started.

The "Run an action" window

You can find execution process and result of the action on the ***Jobs*** page. On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs.

Start

Eugenia Kuzina

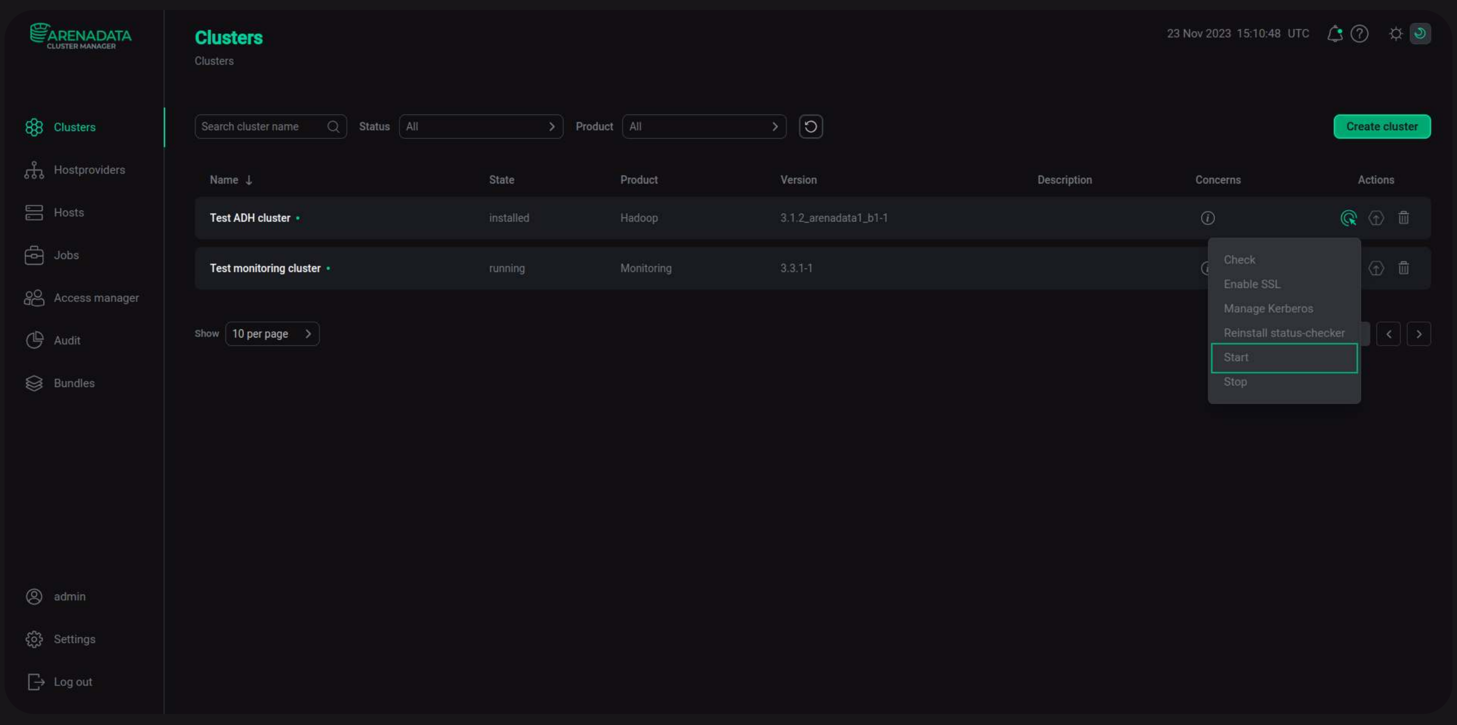
To Table of Contents

IMPORTANT

If you installed a monitoring cluster and want to use it with the ADH cluster, start the monitoring cluster first. To do this, run the ***Start*** action of the monitoring cluster in the same way as described for the ADH cluster.

The ***Start*** action starts all the services installed in the ADH cluster, one by one, in the correct order.

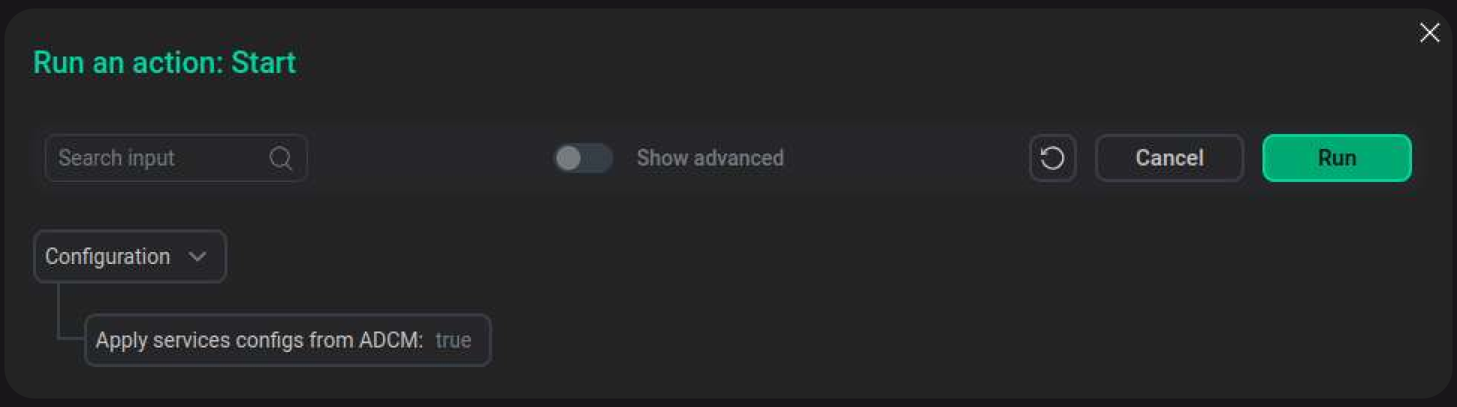
To run an action, go to ***Clusters*** page, click in the ***Actions*** column, and select the action from the drop-down list that appears.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
23 Nov 2023 15:10:48 UTC
Search cluster name
Status
Product
Create cluster
Show
10 per page
Check
Enable SSL
Manage Kerberos
Reinstall status-checker
Start
Stop

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			
Test monitoring cluster	running	Monitoring	3.3.1-1			

Execute the "Start" action

The dialog window of the ***Start*** action contains the ***Apply services configs from ADCM*** setting that allows you to determine whether configuration options specified in ADCM UI should be applied to ADH services.


Run an action: Start
Search input
Show advanced
Cancel
Run
Configuration

Apply services configs from ADCM: true

The "Run an action" window

Click ***Run*** to start the cluster. In the action confirmation window that appears, you can select the ***Verbose*** checkbox to see additional execution details.

You can find execution process and result of the action on the ***Jobs*** page. On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs.

Alternatively, you can use the ***Start*** service action of a particular service to manually start only required services. It is necessary to run services in the order shown in the table below.

Number in order	Service name
1	Monitoring (if you use the monitoring cluster)
2	Zookeeper
3	MySQL
4	HDFS
5	YARN
6	HBase
7	Hive
8	Spark
9	Flink
10	Solr
11	Airflow

Most ADH services require the following running services:

- Zookeeper
- HDFS
- YARN

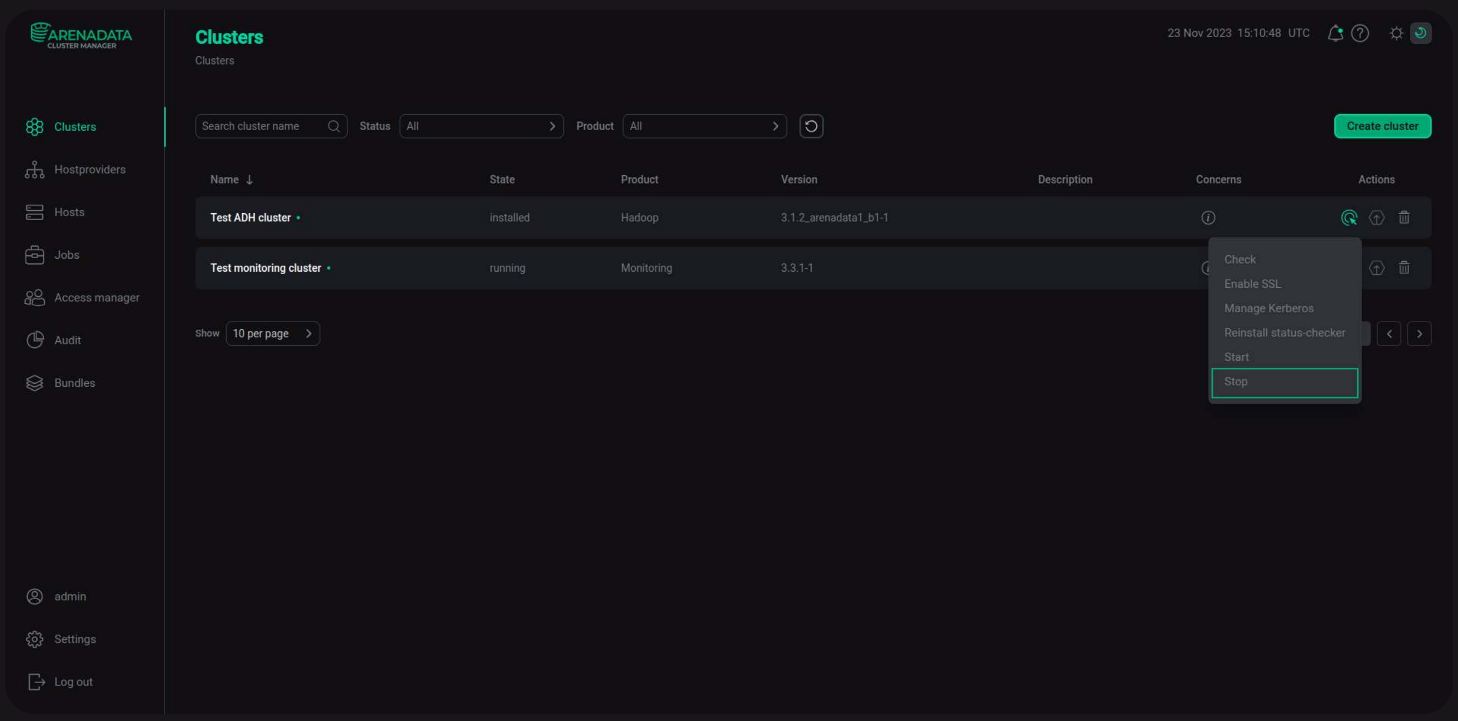
Stop

Eugenia Kuzina

To Table of Contents

The ***Stop*** action stops all the services installed in the ADH cluster in the correct order.

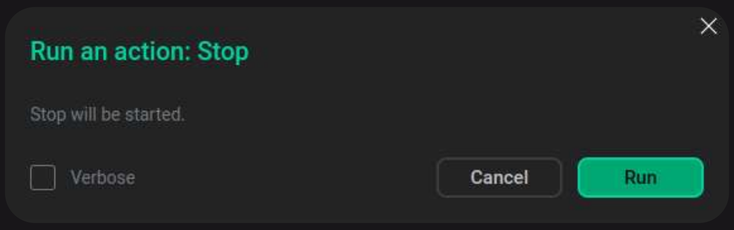
To run an action, go to ***Clusters*** page, click in the ***Actions*** column, and select the action from the drop-down list that appears.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
23 Nov 2023 15:10:48 UTC
Search cluster name
Status
All
Product
All
Create cluster
Check
Enable SSL
Manage Kerberos
Reinstall status-checker
Start
Stop
Show
10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADH cluster	installed	Hadoop	3.1.2_arenadata1_b1-1			
Test monitoring cluster	running	Monitoring	3.3.1-1			

Execute the "Stop" action

When you choose an action, ADCM displays a dialog window to confirm the choice. In this dialog window, you can select the ***Verbose*** checkbox to see additional execution details.


☐ Verbose
Cancel
Run

Run an action: Stop

Stop will be started.

The "Run an action" window

You can find execution process and result of the action on the ***Jobs*** page. On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs.

Alternatively, you can use the ***Stop*** service action of a particular service to manually stop only required services or the entire cluster. It is necessary to stop services in the order shown in the table below.

Number in order	Service name
1	Airflow
2	Solr
3	Flink
4	Spark
5	Hive
6	HBase
7	YARN
8	HDFS
9	MySQL
10	Zookeeper
11	Monitoring

Cluster upgrade

Konstantin Alpashkin

Contents

To Table of Contents

Bundle naming scheme

Available bundle versions

Upgrade process

- Major version upgrade
- Minor version upgrade

This section explains ADH bundles versioning and provides guidance on upgrading an ADH cluster to a newer version.

Bundle naming scheme

A typical ADH bundle has the following naming pattern.

Before 3.x

```
adcm_cluster_hadoop_v<X>.<Y>.<Z>_b<N>-<M>_<E>.tgz
```

Where:

- `<X>.<Y>.<Z>` — the package version.
- `b<N>-<M>` — the version of the ansible scripts describing the bundle.
- `<E>` — indicates the edition, either Community or Enterprise.

3.x and later

```
adcm_cluster_hadoop_v<X>_arenadata<A>_b<N>-<M><E>.tgz
```

Where:

- `<X>` — the package version that indicates the core service (HDFS) version.
- `<A>` — the consecutive release number for the given `<X>` version.
- `b<N>-<M>` — the version of the ansible scripts describing the bundle.
- `<E>` — indicates the edition, either Community or Enterprise.

For example:

- `adcm_cluster_hadoop_v2.1.8_b1-1_enterprise.tgz` — the package version is 2.1.8, the bundle code version is b1-1, and the bundle edition is Enterprise.
- `adcm_cluster_hadoop_v3.1.2_arenadata1_b1-enterprise.tgz` — the package version is 3.1.2_arenadata1, the first 3.x release, the bundle code version is b1, and the bundle edition is Enterprise.

Available bundle versions

Below you can view available versions of ADH/ADPS bundles.

ADH bundles

Bundle version	Code versions
2.1.4	b1 — b11
2.1.6	b1 — b4
2.1.7	b1
2.1.8	b3
2.1.10	b1
3.1.2.1	b1 — b2
3.2.4.1	b1 — b3
3.2.4.2	b1 — b2
3.2.4.3	b1

ADPS bundles

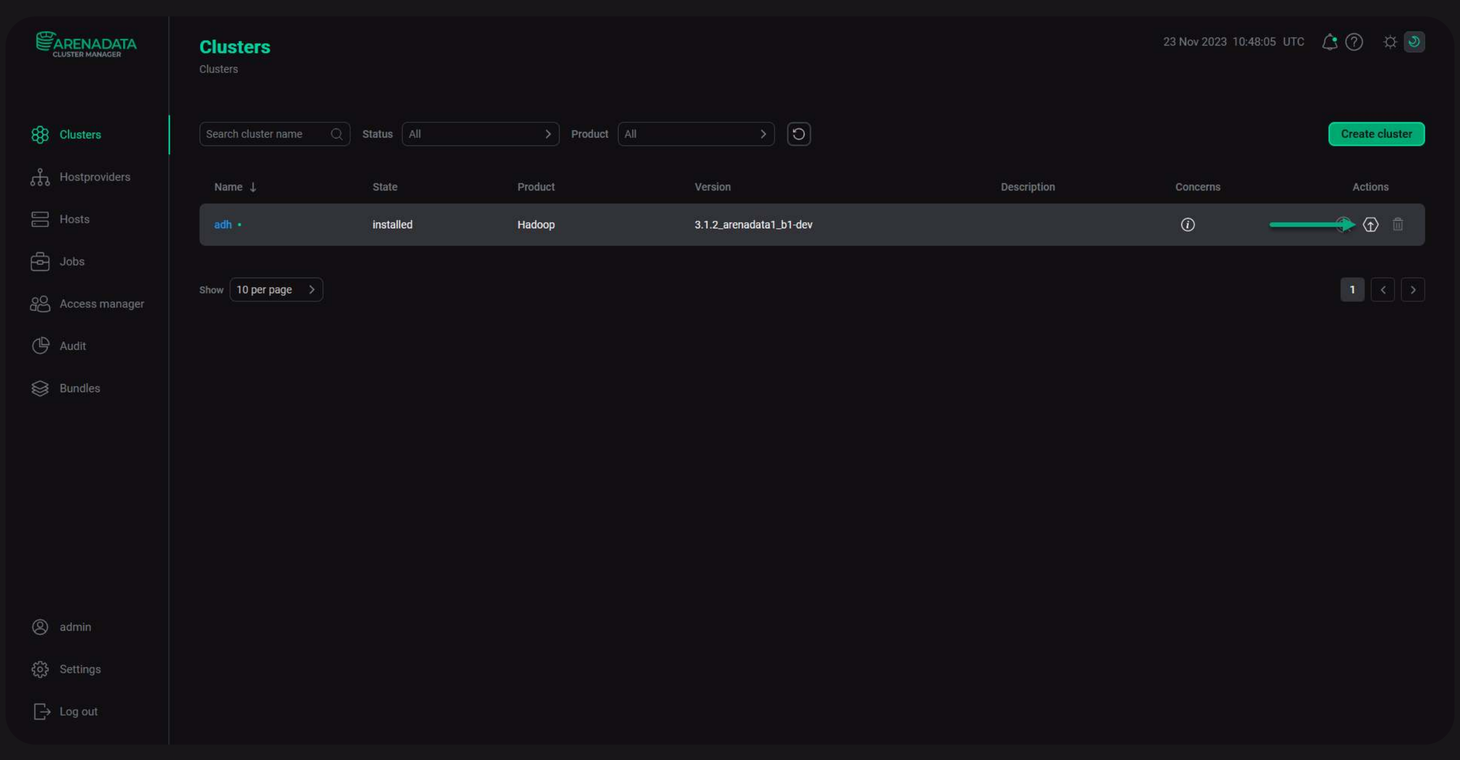
Bundle version	Code versions
1.0.2	b1 — b9
1.0.3	b1 — b2
1.0.4	b1 — b4
1.0.5	b1 — b2
1.1.0	b1 — b2
1.1.1	b1 — b2
1.1.2	b1
1.2.0	b1

Upgrade process

NOTE

Before updating your cluster, ensure that the configurations on the hosts match the ADCM configurations. If there were manual edits on the hosts, you must bring them into a consistent state yourself.

To upgrade an ADH cluster, you should upload the new ADH bundle to ADCM. Once uploaded, ADCM recognizes the fresh bundle version and the ***Upgrade*** cluster action becomes available in the ADCM UI.



Name	State	Product	Version	Description	Concerns	Actions
adh	installed	Hadoop	3.1.2_arenadata1_b1-dev			

Upgrade action

CAUTION

When you upgrade an ADH cluster, the products must be upgraded in the following order:

1. ADCM
2. ADPS
3. ADH

Consider the basic rules below when upgrading to a newer major/minor version.

Major version upgrade

A major version upgrade assumes the update of packages content and the bundle. Upgrades to a new major version must be done from the **maximum** minor version of the previous major release. Below are some migration examples:

- ✅ 2.1.6_b4 → 2.1.7_b1. Valid upgrade.
- 2.1.8_b3 → 2.1.10_b1. Valid upgrade.
- 3.1.2_arenadata1_b1 → 3.1.2_arenadata2_b1. Valid upgrade.
- 3.1.2_arenadata2_b1 → 3.2.4_arenadata1_b1. Valid if 3.1.2_arenadata2_b1 is the last version within 3.1.2.
- ❌ 2.1.6_b3 → 2.1.7_b1. Forbidden. 2.1.6_b3 is not the latest bundle within the 2.1.6 release.
- 2.1.6_b4 → 2.1.8_b1. Forbidden. Cannot skip the 2.1.7 minor version.
- 3.1.2_arenadata1_b1 → 3.1.2_arenadata3_b1. Forbidden. Cannot skip the 3.1.2_arenadata2_bX version.

Minor version upgrade

A minor version upgrade assumes the update of bundle code. Upgrades to a new minor release can be done from any version upwards within the same major release. For example:

- ✅ 2.1.4_b1 → 2.1.4_b2
- 2.1.4_b1 → 2.1.4_b11
- ✅ 3.1.2_arenadata1_b1 → 3.1.2_arenadata1_b5

NOTE

The above upgrade rules are also applicable for ADPS bundles.

Add a host to a cluster

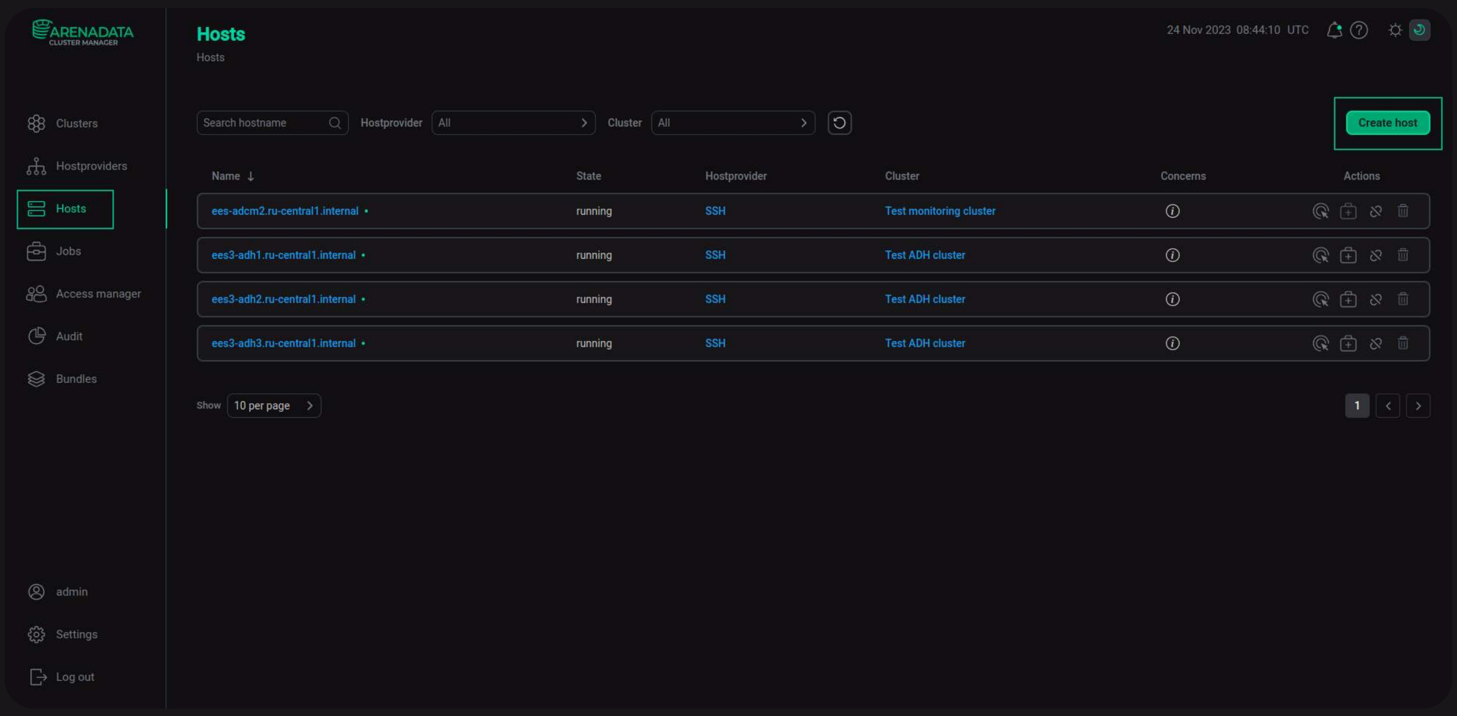
Albert Bagdasaryan, Eugenia Kuzina

Use ADCM to add hosts to your ADH cluster, as explained below.

> **NOTE**
>
> To manage an ADH cluster, install Arenadata Cluster Manager (ADCM) and ADH as described in the following tutorial: Get started with Arenadata Hyperwave.

Follow these steps to add a host to your cluster:

1. Select the ***Hosts*** item in the left navigation menu and click ***Create host***.

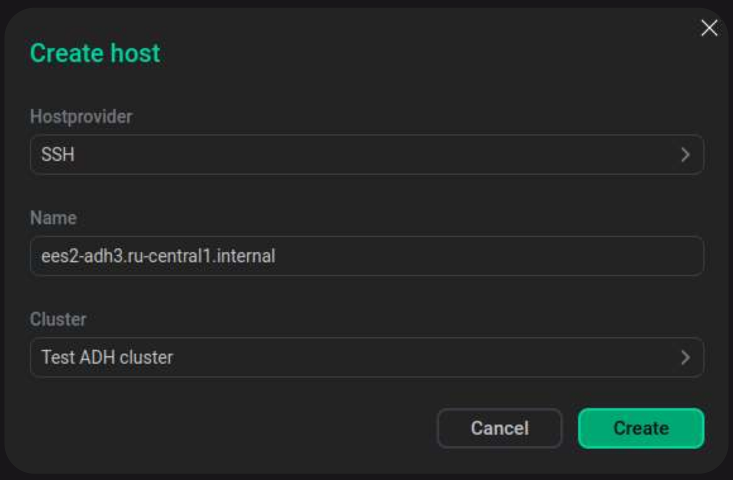

ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Hosts
Hosts
24 Nov 2023 08:44:10 UTC
Search hostname
Hostprovider
All
Cluster
All
Create host
Show
10 per page
1

Name ↓	State	Hostprovider	Cluster	Concerns	Actions
ees-adcm2.ru-central1.internal	running	SSH	Test monitoring cluster		
ees3-adh1.ru-central1.internal	running	SSH	Test ADH cluster		
ees3-adh2.ru-central1.internal	running	SSH	Test ADH cluster		
ees3-adh3.ru-central1.internal	running	SSH	Test ADH cluster		

Add a host

2. In the opened dialog:

   - Select a hostprovider from a drop-down list in the ***Hostprovider*** field. Hostproviders are displayed with the names that you assigned to them.
   - Enter a fully qualified domain name (FQDN) of the host in the ***Name*** field. FQDN should contain both the host name and the domain name.
   - Select a cluster from a drop-down list in the ***Cluster*** field.
   - Click ***Create***.


Create host
Cancel
Create

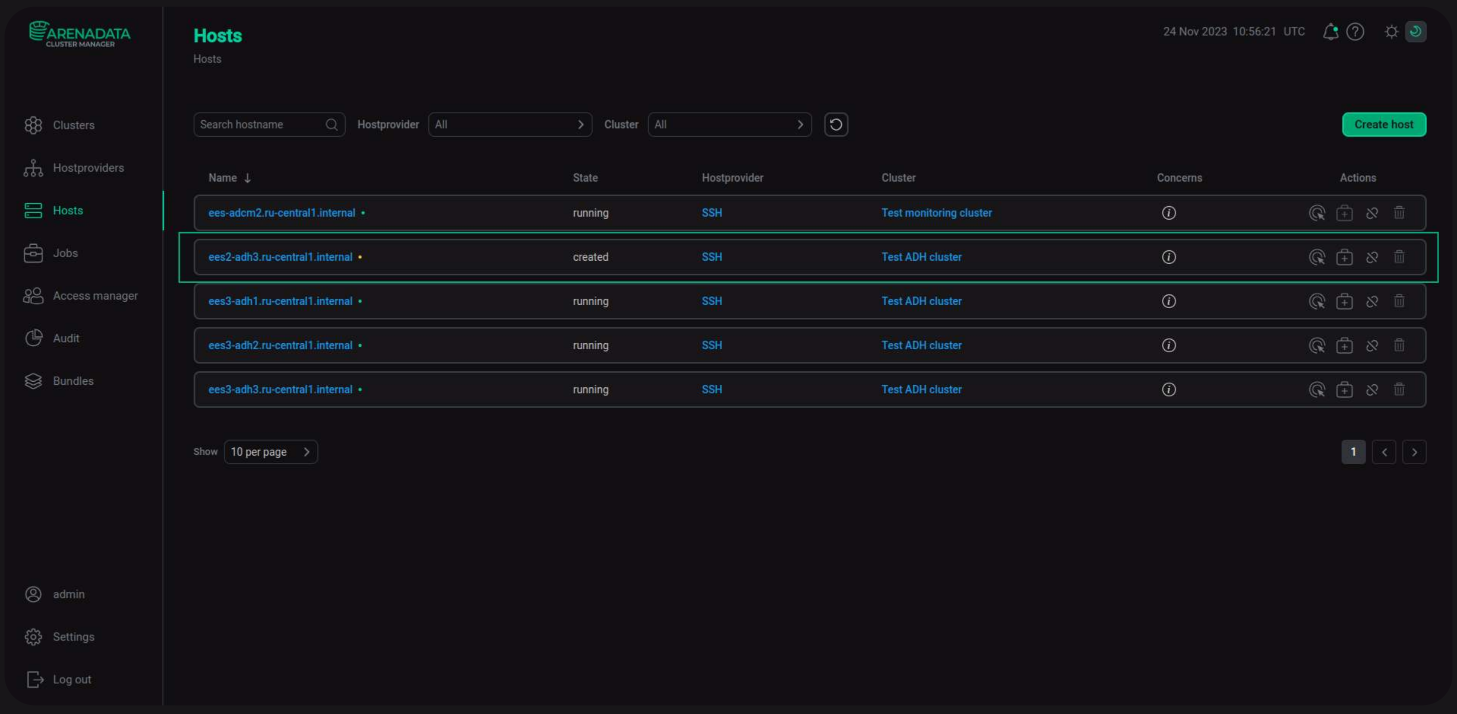
Hostprovider: SSH

Name: ees2-adh3.ru-central1.internal

Cluster: Test ADH cluster

Specify host parameters

The new host is displayed on the ***Hosts*** page.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Hosts
Hosts
24 Nov 2023 10:56:21 UTC
Search hostname
Hostprovider
All
Cluster
All
Create host
Show
10 per page
1

Name ↓	State	Hostprovider	Cluster	Concerns	Actions
ees-adcm2.ru-central1.internal	running	SSH	Test monitoring cluster		
ees2-adh3.ru-central1.internal	created	SSH	Test ADH cluster		
ees3-adh1.ru-central1.internal	running	SSH	Test ADH cluster		
ees3-adh2.ru-central1.internal	running	SSH	Test ADH cluster		
ees3-adh3.ru-central1.internal	running	SSH	Test ADH cluster		

The "Hosts" page with the new host

View host status

Albert Bagdasaryan, Eugenia Kuzina

Contents

To Table of Contents

Check host status using the Hosts page

Check the host status in its profile

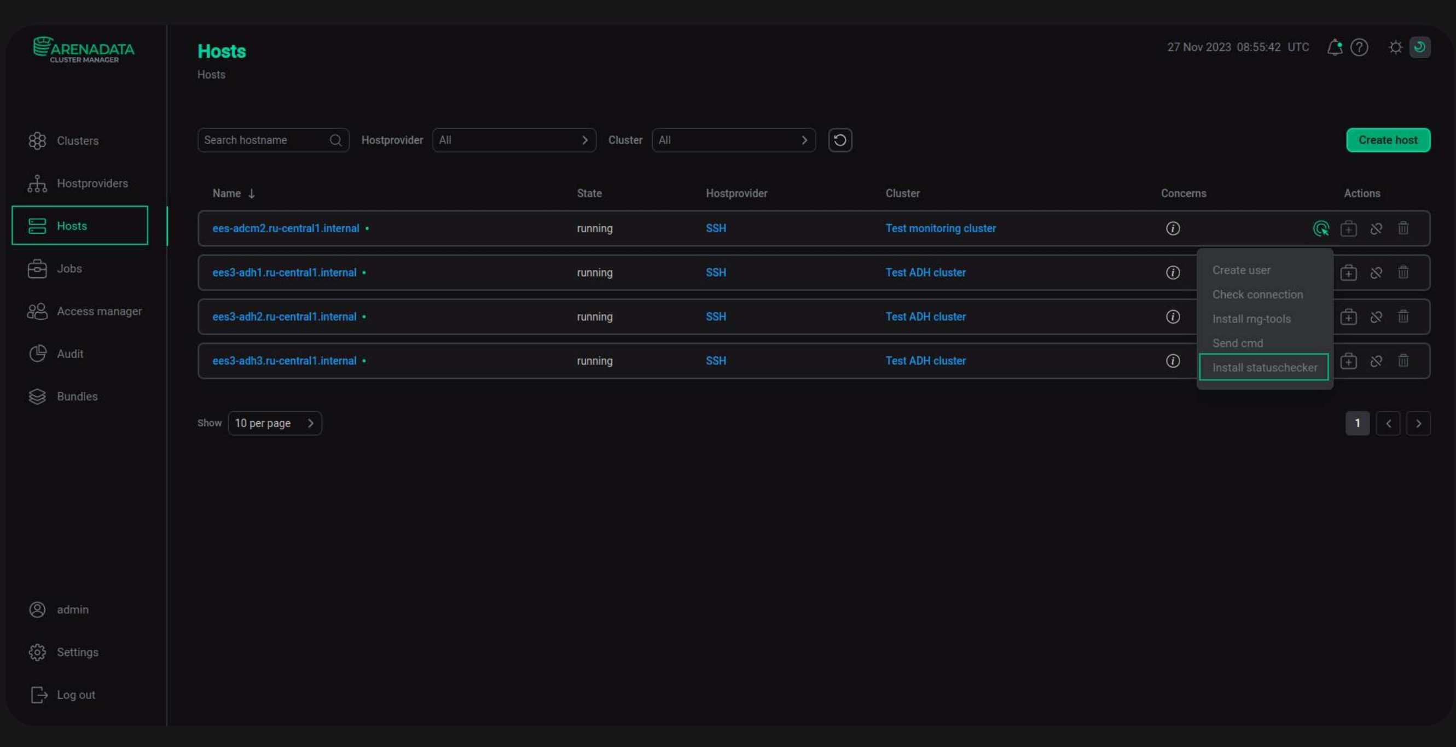
You can check a host status using the following methods provided by ADCM:

- Using the main ***Hosts*** page.
- Using the host profile.

Check host status using the Hosts page

Follow the steps below to view the host status:

1. In ADCM, select the ***Hosts*** item in the left navigation menu.
2. If you did not install statuschecker before, click in the ***Actions*** column and select the ***Install statuschecker*** action in the drop-down list.

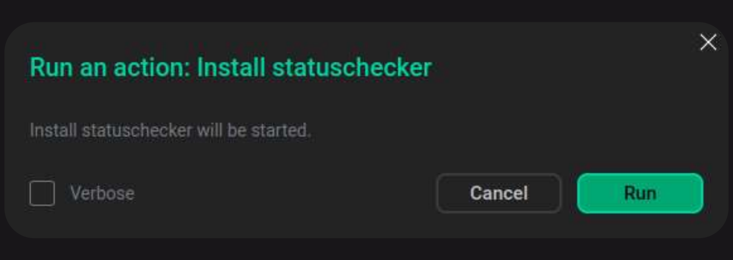


ARENADATA CLUSTER MANAGER
Hosts
Hosts
27 Nov 2023 08:55:42 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Cluster
All
Create host
Create user
Check connection
Install mg-tools
Send cmd
Install statuschecker
Show
10 per page
1

Name	State	Hostprovider	Cluster	Concerns	Actions
ees-adcm2.ru-central1.internal •	running	SSH	Test monitoring cluster		
ees3-adh1.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh2.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh3.ru-central1.internal •	running	SSH	Test ADH cluster		

Start statuschecker installation

3. Click ***Run*** to confirm the statuschecker installation. You can track the task status on the ***Jobs*** page.

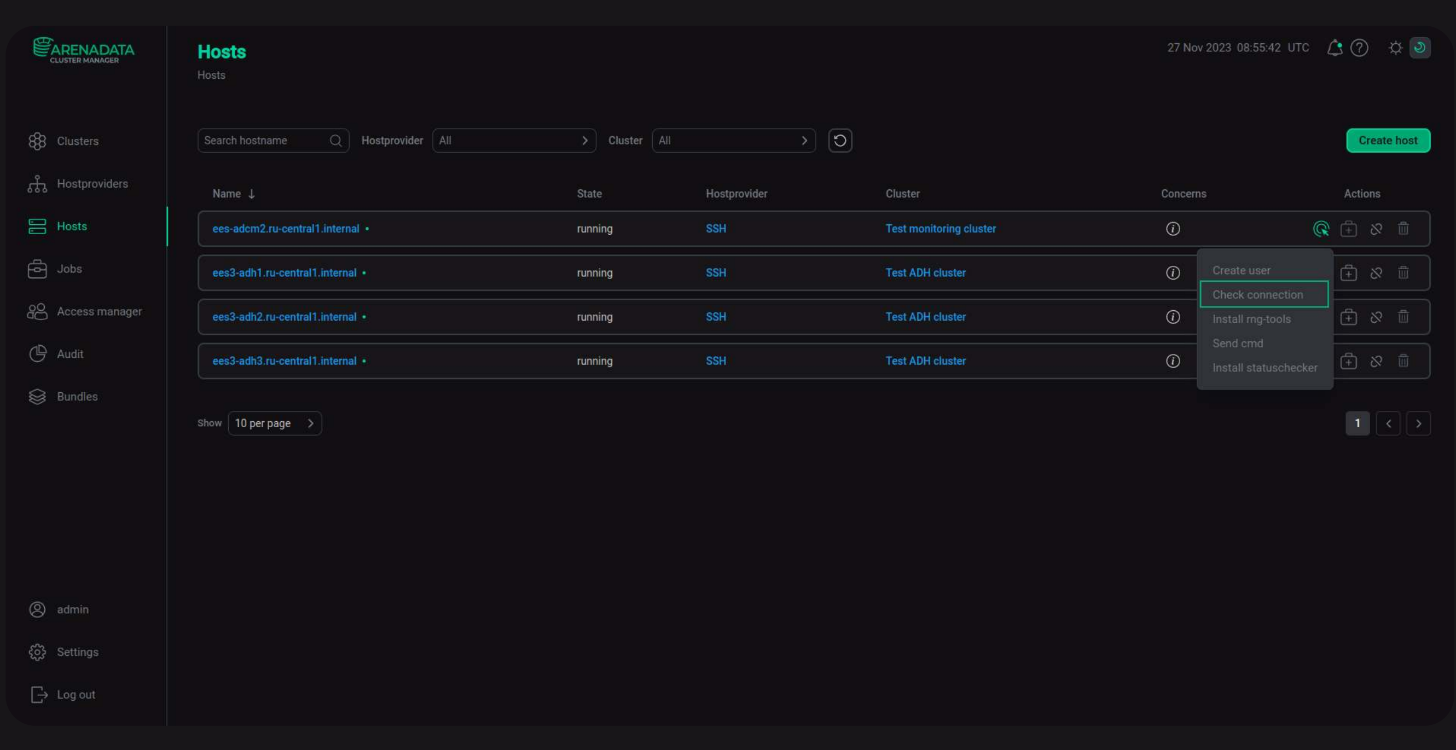


Run an action: Install statuschecker
☐ Verbose
Cancel
Run

Install statuschecker will be started.

Confirm the statuschecker installation

4. Click again and choose the ***Check connection*** action.

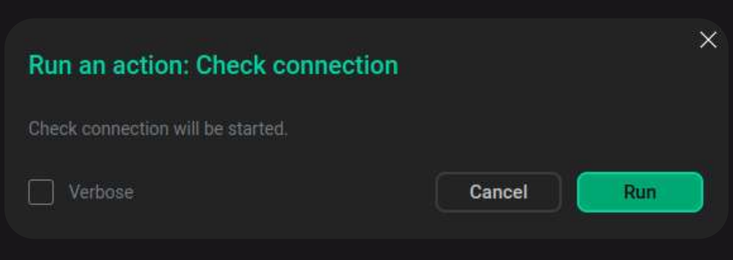


ARENADATA CLUSTER MANAGER
Hosts
Hosts
27 Nov 2023 08:55:42 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Cluster
All
Create host
Create user
Check connection
Install mg-tools
Send cmd
Install statuschecker
Show
10 per page
1

Name	State	Hostprovider	Cluster	Concerns	Actions
ees-adcm2.ru-central1.internal •	running	SSH	Test monitoring cluster		
ees3-adh1.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh2.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh3.ru-central1.internal •	running	SSH	Test ADH cluster		

Start "Check connection" action

5. In the opened window, click ***Run*** to confirm the action.

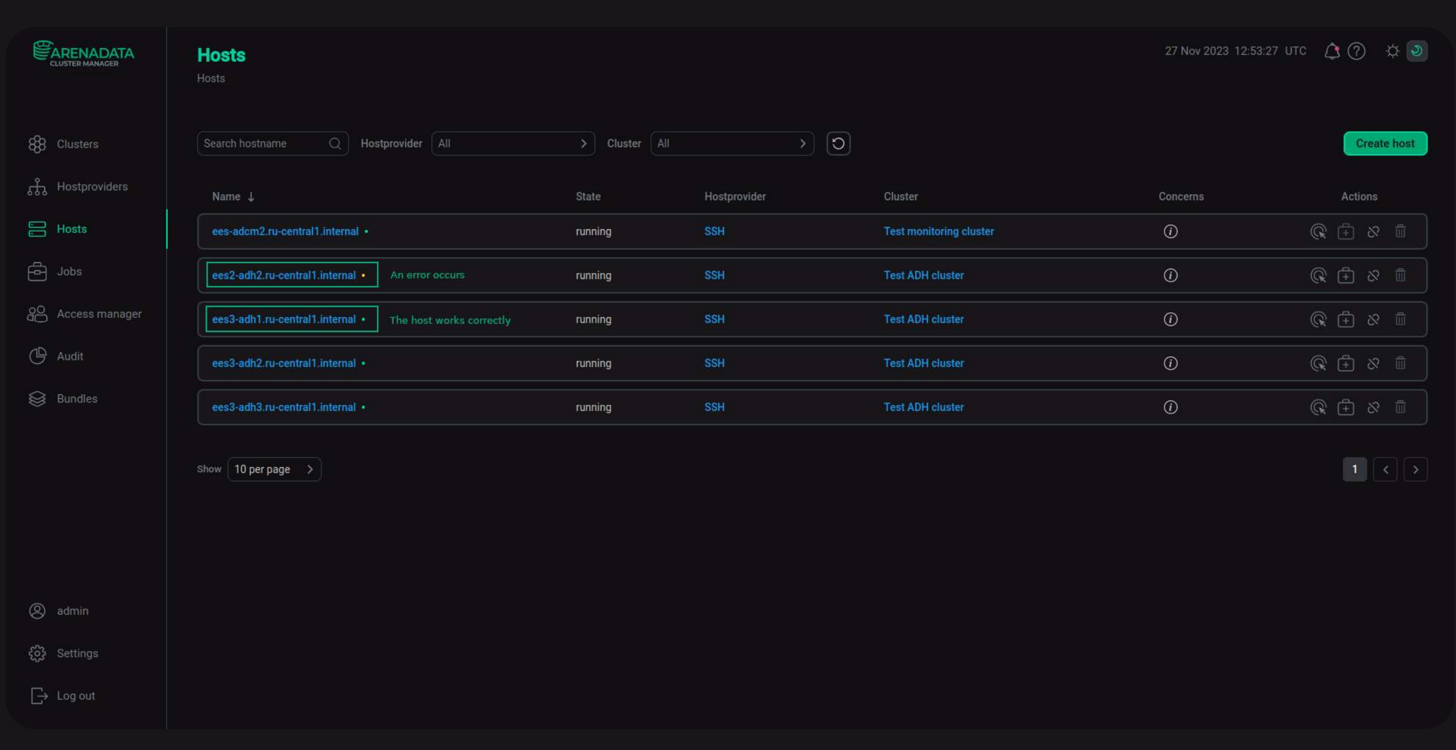


Run an action: Check connection
☐ Verbose
Cancel
Run

Check connection will be started.

Confirm the "Check connection" action

6. Check the host status. If the host is available, a green dot is displayed next to the host name. If a yellow dot is shown in the ***Name*** column, there are some errors. You can find details on the ***Jobs*** page.



ARENADATA CLUSTER MANAGER
Hosts
Hosts
27 Nov 2023 12:53:27 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Cluster
All
Create host
An error occurs
The host works correctly
Show
10 per page
1

Name	State	Hostprovider	Cluster	Concerns	Actions
ees-adcm2.ru-central1.internal •	running	SSH	Test monitoring cluster		
ees2-adh2.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh1.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh2.ru-central1.internal •	running	SSH	Test ADH cluster		
ees3-adh3.ru-central1.internal •	running	SSH	Test ADH cluster		

Determine whether a host works correctly

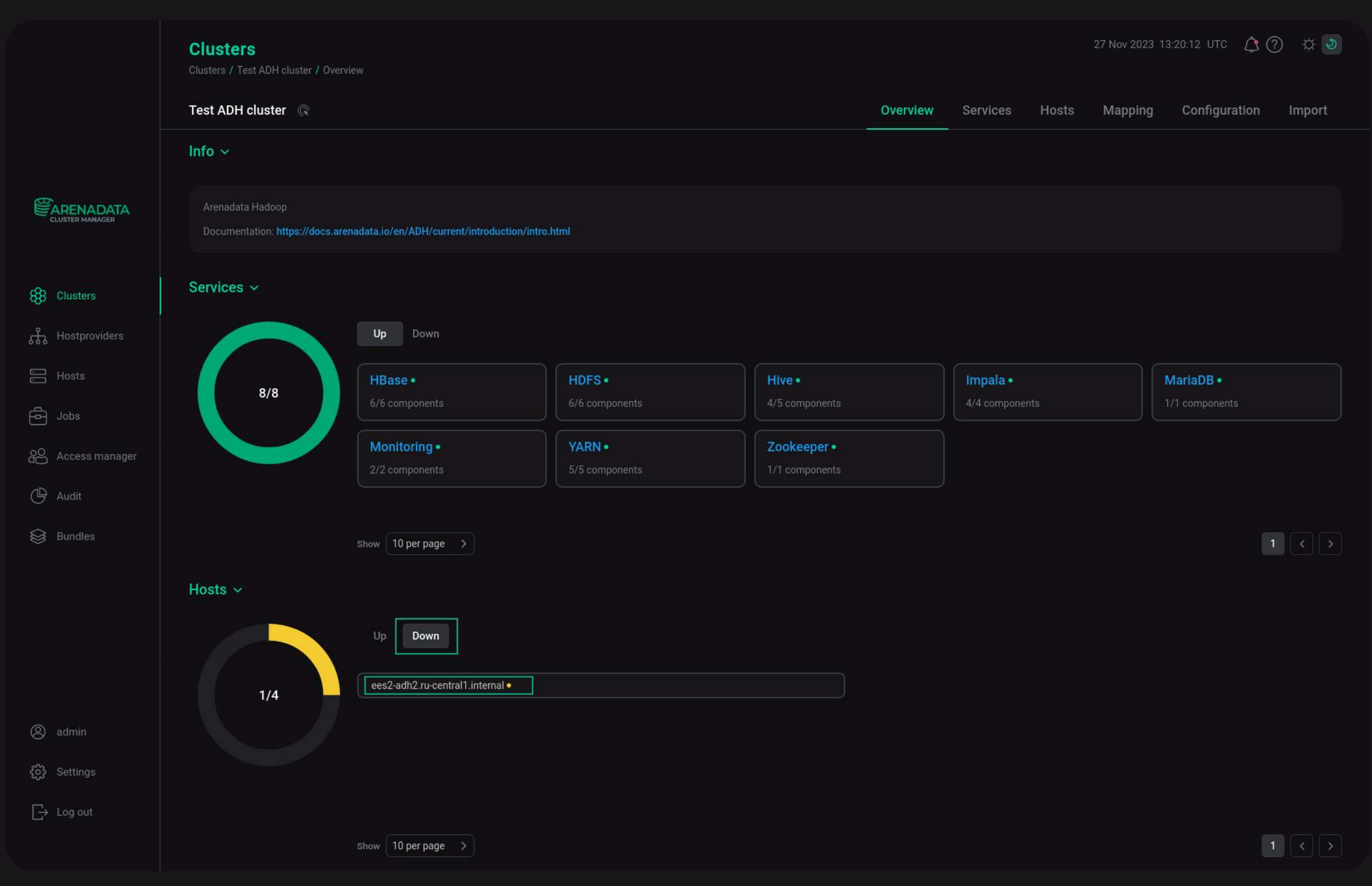
Check the host status in its profile

Follow these steps to view the host status:

1. Click the required host near the ***Hosts*** diagram on the ***Overview*** cluster tab. Cluster hosts are distributed on two tabs:

   - ***Up*** — hosts that operate correctly;
   - ***Down*** — unavailable hosts.

   To view hosts that do not work correctly, switch to the ***Down*** tab.



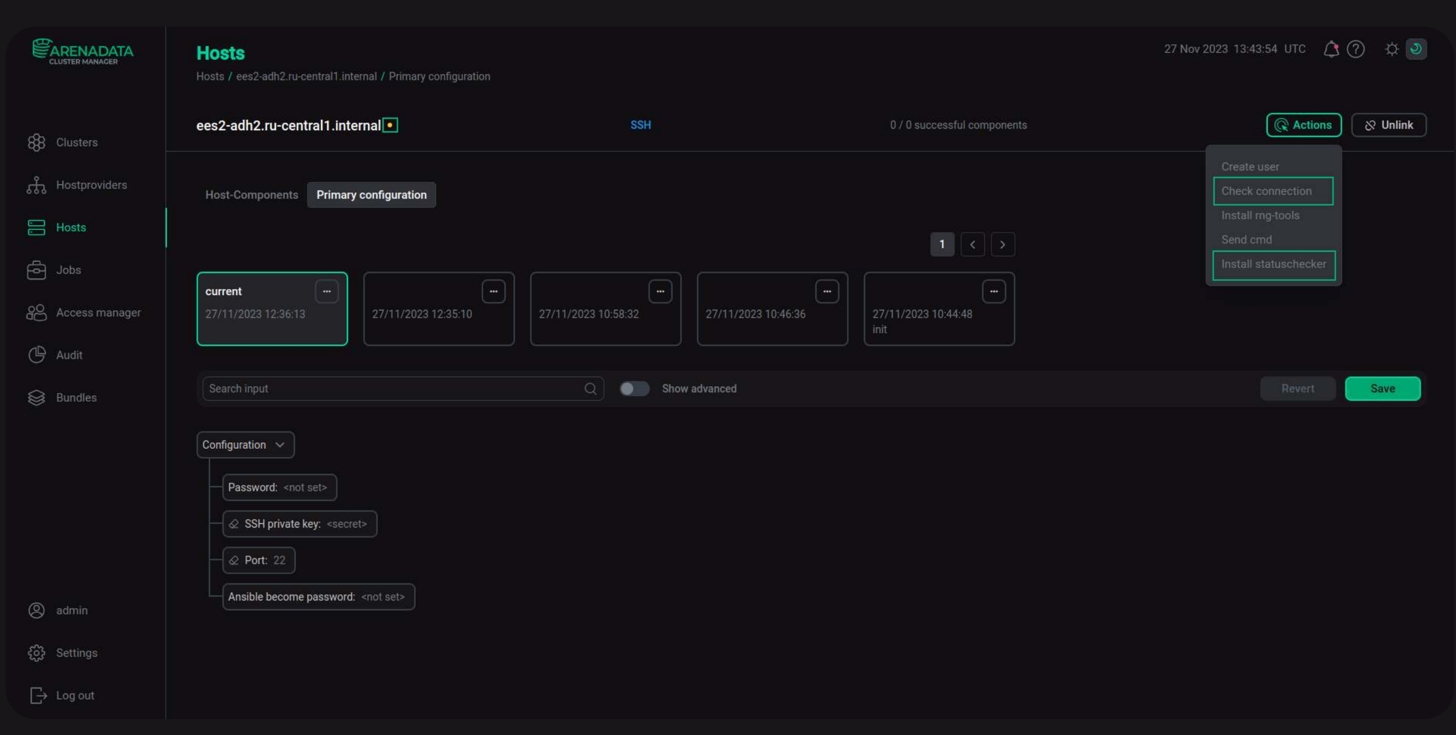
ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADH cluster / Overview
27 Nov 2023 13:20:12 UTC
Test ADH cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Info
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Services
8/8
Up
Down
HBase •
6/6 components
HDFS •
6/6 components
Hive •
4/5 components
Impala •
4/4 components
MariaDB •
1/1 components
Monitoring •
2/2 components
YARN •
5/5 components
Zookeeper •
1/1 components
Show
10 per page
1
Hosts
1/4
Up
Down
ees2-adh2.ru-central1.internal •

Arenadata Hadoop
Documentation: https://docs.arenadata.io/en/ADH/current/introduction/intro.html

Open host settings

   The page with host settings opens.

2. If you did not install statuschecker before, click ***Actions*** in the upper right corner. The displayed drop-down list contains the same actions that you can find in the host ***Actions*** column. Install the statuschecker and check connection as described above.



ARENADATA CLUSTER MANAGER
Hosts
Hosts / ees2-adh2.ru-central1.internal / Primary configuration
27 Nov 2023 13:43:54 UTC
ees2-adh2.ru-central1.internal
SSH
0 / 0 successful components
Actions
Unlink
Create user
Check connection
Install mg-tools
Send cmd
Install statuschecker
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Host-Components
Primary configuration
1
current
27/11/2023 12:36:13
27/11/2023 12:35:10
27/11/2023 10:58:32
27/11/2023 10:46:36
27/11/2023 10:44:48
init
Search input
Show advanced
Revert
Save
Configuration

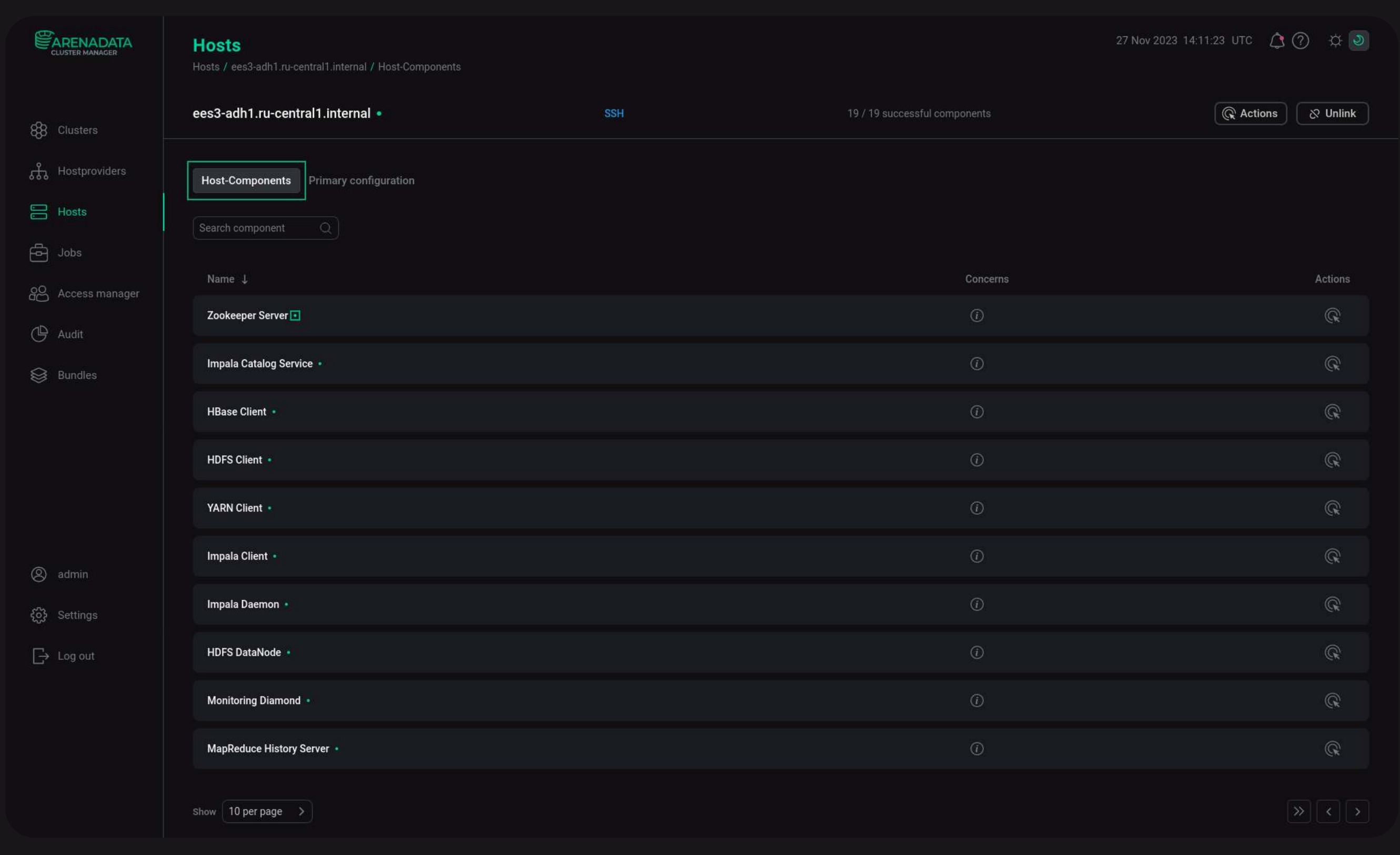
Password: <not set>
SSH private key: <secret>
Port: 22
Ansible become password: <not set>

Host settings

   The dot displayed next to the host name determines the host status. The green dot indicates that the host works correctly, the yellow one — a problem occurs. You can find additional information about errors on the ***Jobs*** page.

3. On the ***Host-Components*** tab, you can get more information about the current host:

   - All components installed on the host.
   - Status of each component.



ARENADATA CLUSTER MANAGER
Hosts
Hosts / ees3-adh1.ru-central1.internal / Host-Components
27 Nov 2023 14:11:23 UTC
ees3-adh1.ru-central1.internal •
SSH
19 / 19 successful components
Actions
Unlink
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Host-Components
Primary configuration
Search component
Show
10 per page

Name	Concerns	Actions
Zookeeper Server •		
Impala Catalog Service •		
HBase Client •		
HDFS Client •		
YARN Client •		
Impala Client •		
Impala Daemon •		
HDFS DataNode •		
Monitoring Diamond •		
MapReduce History Server •		

Host components and their statuses

   The green dot in the ***Name*** column shows that a component operates correctly, the yellow one — a problem occurs.

Remove a host from a cluster

Sergei Tikhomirov

Contents

To Table of Contents

Step 1. Prepare the host

Step 2. Remove components from the host

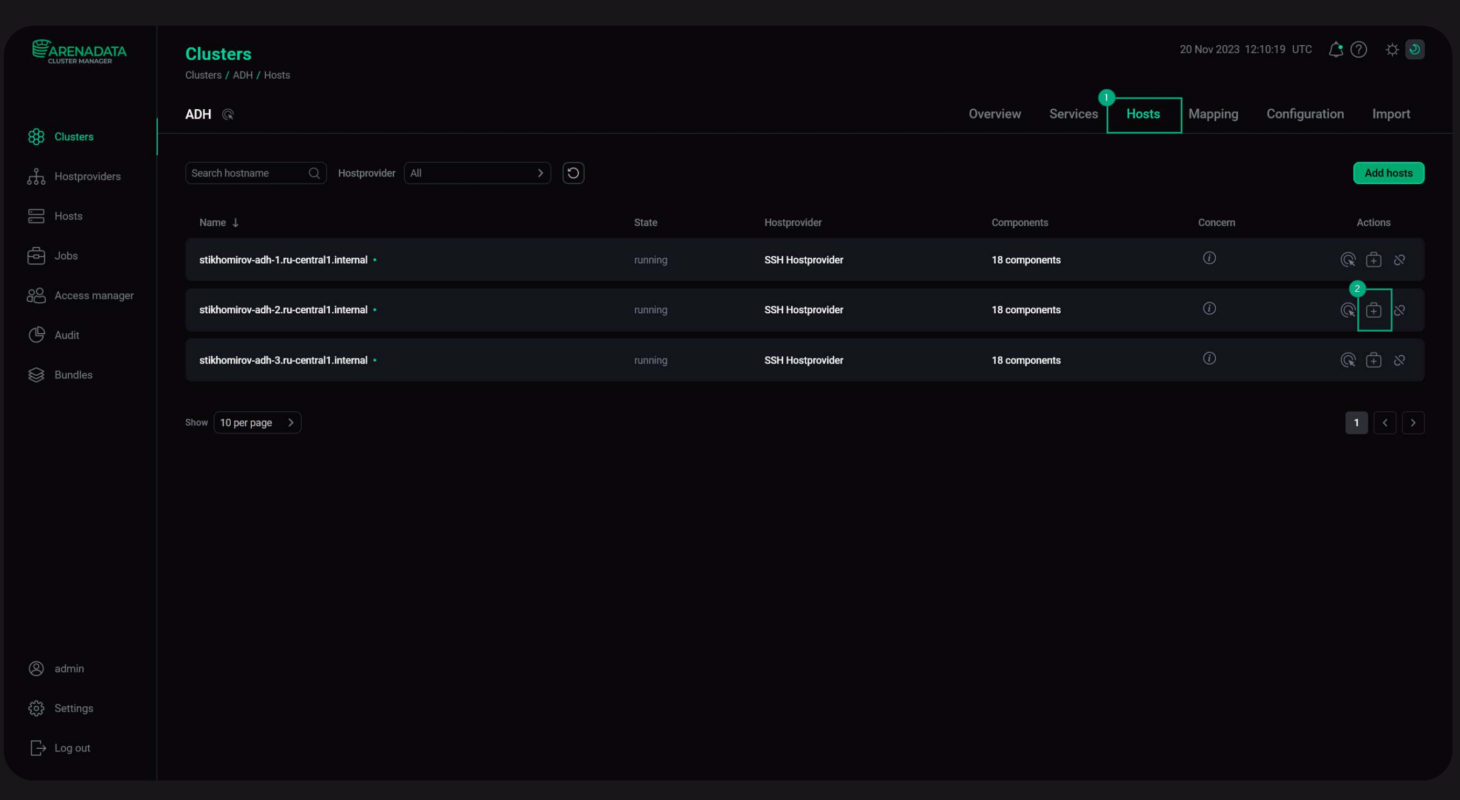
Step 3. Remove the host

Removing a host from a cluster is a tricky operation that may be required in case some node becomes unavailable. Normally, when performing the ***Remove***, ***Move***, or ***Decommiss*** actions on unavailable hosts you get unavailability errors. To ignore such errors the maintenance mode was introduced.

To remove a host from a cluster using maintenance mode, follow these steps.

Step 1. Prepare the host

1. Turn on the maintenance mode on the host you want to remove. You can do that on the ***Hosts*** page or on the ***Hosts*** tab on the chosen cluster page by clicking the ⊞ icon.



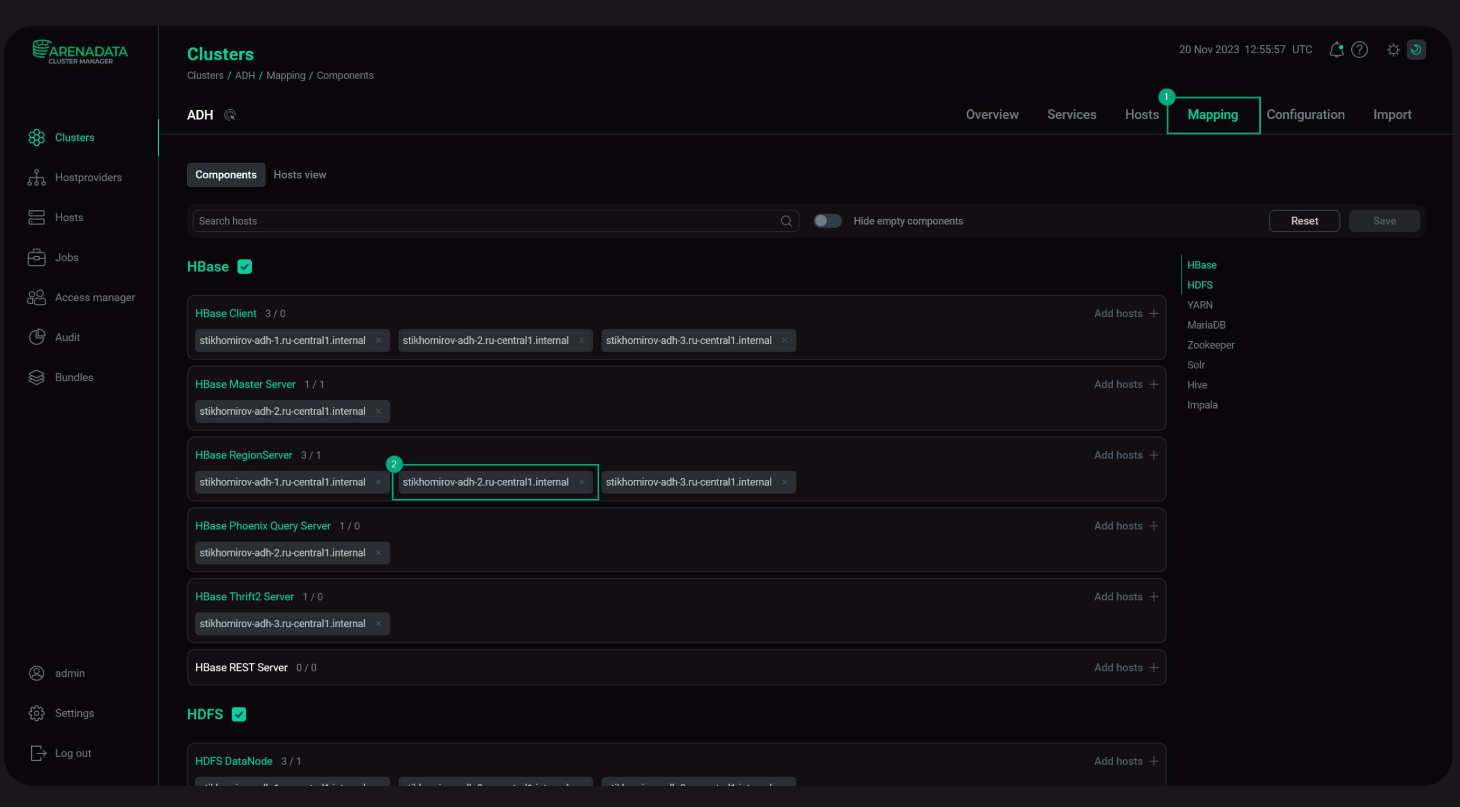
Maintenance mode

2. Check that the replication factor value in HDFS is not smaller than a number of live DataNodes. Otherwise, add a new DataNode or reduce the `dfs.replication` parameter value. It has to be done because insufficient number of replicas leads to moving the NameNode to safe mode that makes DataNodes and dependant components (HBase RegionServer, Tez, etc.) go down.

Step 2. Remove components from the host

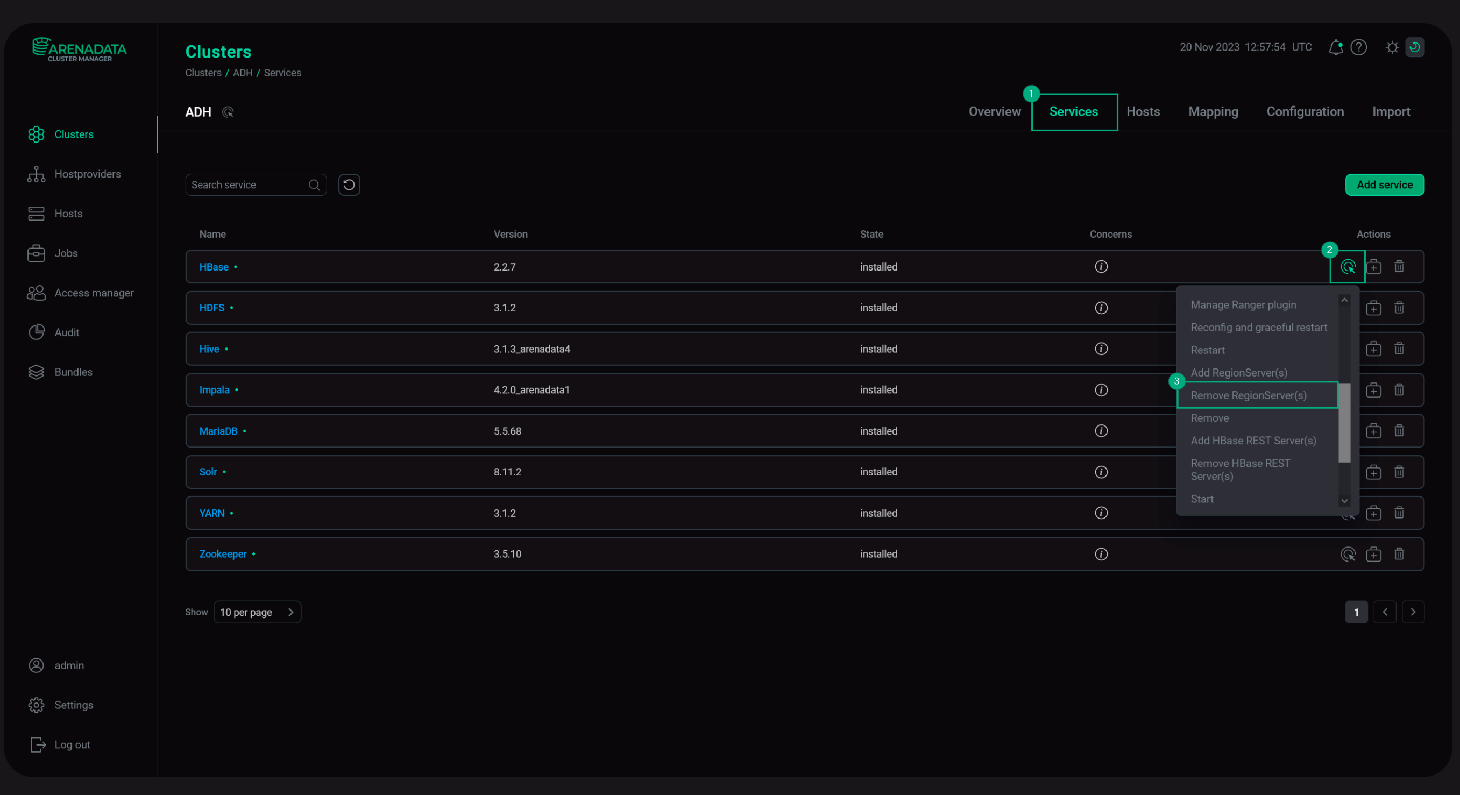
To remove all components from a host, repeat the following steps for each component:

1. Check if a component is present on the host using the ***Mapping*** tab.



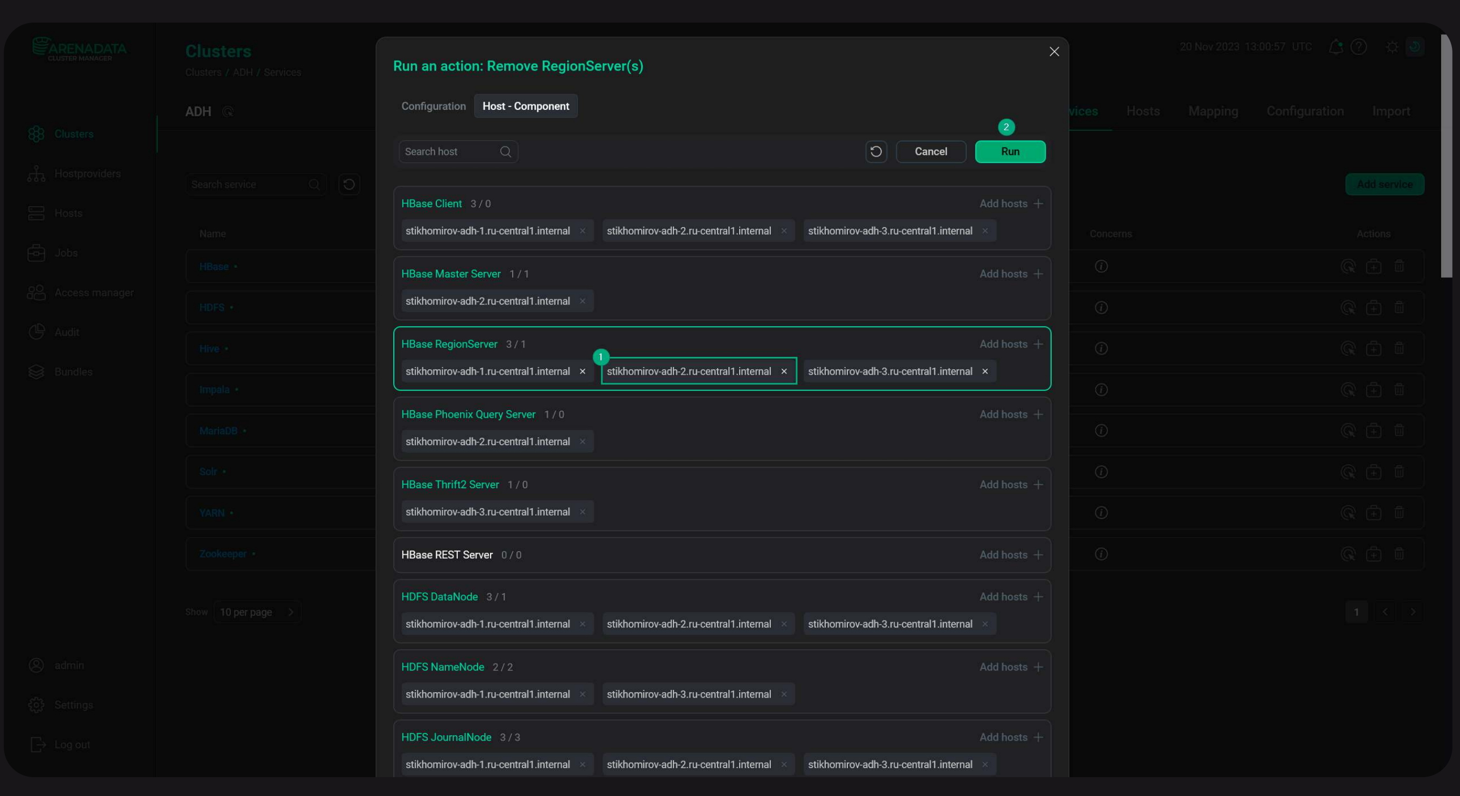
Mapping tab

2. If it is present on the unavailable host, go to the ***Services*** tab, find the corresponding service, and run the action that removes that component.



Service action menu

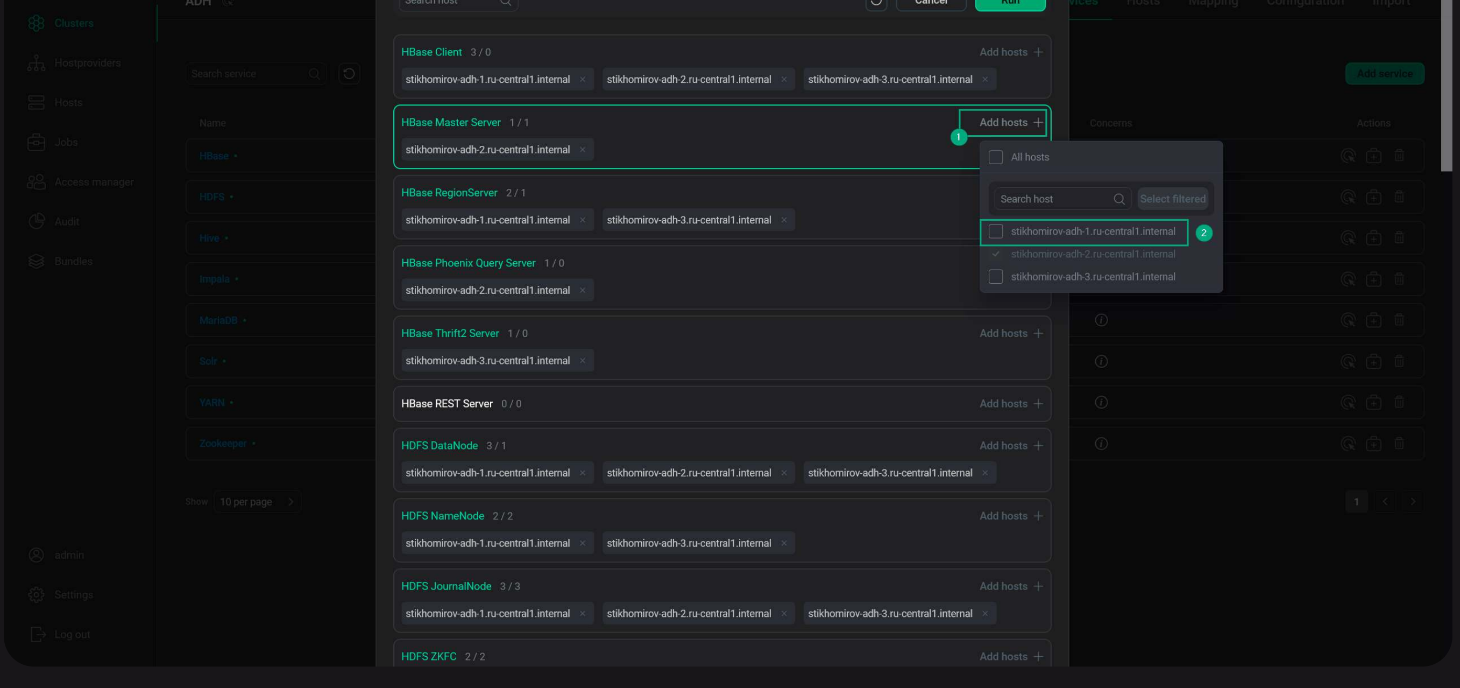
3. In the pop-up window, click × next to the unavailable host. After that, click ***Run*** and wait for the job to complete.



Component removal

Special cases

- Some components cannot be removed from a host until they were added to another host (e.g. HBase Master Server). In that case, run the ***Add*** action for them in the corresponding service action menu and select an additional host for that component prior to removing them from the unavailable host.

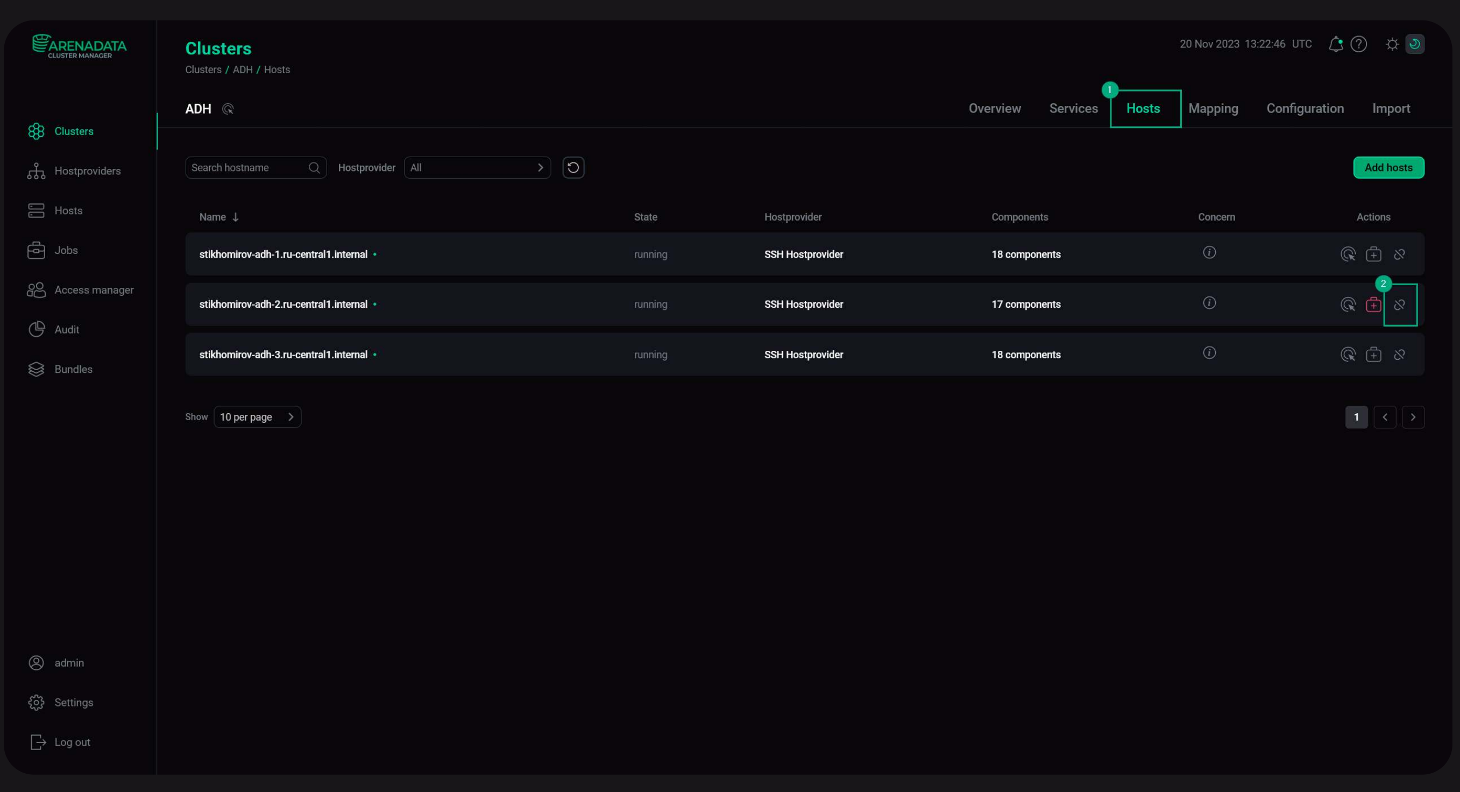


Component addition

- Some components can have only one instance in the cluster (e.g. MapReduce History Server), so they need to be moved to another host using the ***Move*** action of the corresponding service. Keep in mind that while a host is in maintenance mode, it's only possible to remove components from it, so for this operation you need to turn the maintenance mode off and turn it back on once the moving is done.
- If a component is an HDFS DataNode or a YARN NodeManager, it's required to decommission it before removing. You can do that by running the ***Maintenance/Decommission DataNodes*** and ***Decommission/Recommission NodeManagers*** actions for HDFS and YARN, respectively.
- Some services need to be restarted after certain components were moved. Thus, you need to restart Hive and Spark after moving/removing YARN components; HBase after moving HDFS NameNodes; Spark after moving Hive Metastore; all HA services (Hive, YARN, Flink) after moving/removing/adding a Zookeeper Server.

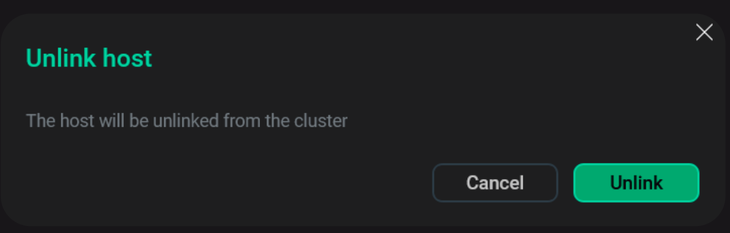
Step 3. Remove the host

Once all the components have been removed from the unavailable host, you can remove it from the cluster by clicking the ⛓ icon.



Host removal

Next, confirm the action by clicking ***Unlink*** in the pop-up window.



Unlink host

The host will be unlinked from the cluster

Action confirmation

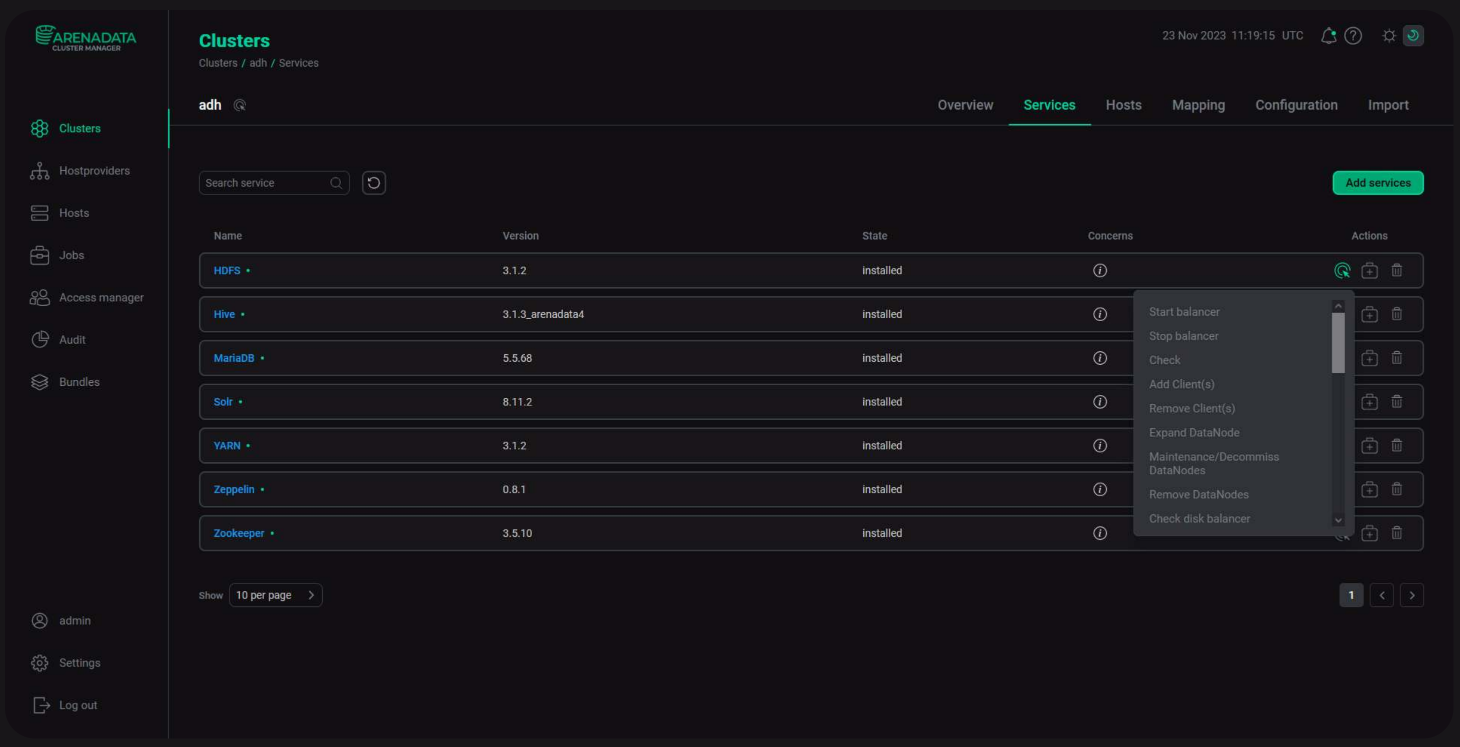
ADH service actions

Konstantin Alpashkin

To Table of Contents

ADCM UI provides a convenient way to interact with your ADH cluster, hosts, and services via actions. To perform an action on a service, do the following:

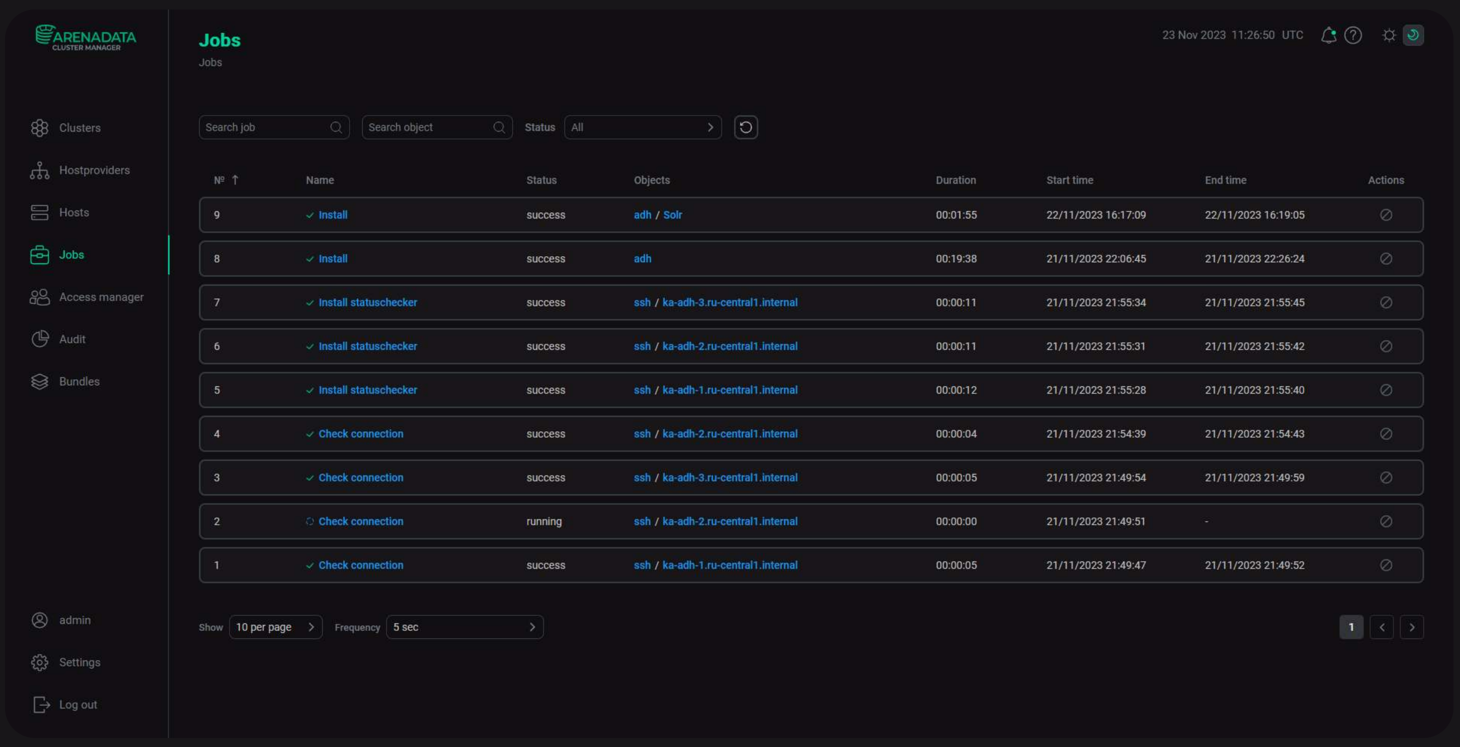
1. On the ***Clusters*** page, select the ADH cluster you want to work with.
2. On the cluster page that opens, go to ***Services***.
3. Click ***Actions*** for the service you need. The list of actions available for the given service is displayed in the drop-down.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / adh / Services
23 Nov 2023 11:19:15 UTC
adh
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Start balancer
Stop balancer
Check
Add Client(s)
Remove Client(s)
Expand DataNode
Maintenance/Decommiss DataNodes
Remove DataNodes
Check disk balancer
Show
10 per page
1

Name	Version	State	Concerns	Actions
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
MariaDB	5.5.68	installed		
Solr	8.11.2	installed		
YARN	3.1.2	installed		
Zeppelin	0.8.1	installed		
Zookeeper	3.5.10	installed		

Running service actions

To view the progress of an action or the action outcome, use the ***Jobs*** page that is shown below.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Jobs
Jobs
23 Nov 2023 11:26:50 UTC
Search job
Search object
Status
All
Show
10 per page
Frequency
5 sec
1

№	Name	Status	Objects	Duration	Start time	End time	Actions
9	✓ Install	success	adh / Solr	00:01:55	22/11/2023 16:17:09	22/11/2023 16:19:05	
8	✓ Install	success	adh	00:19:38	21/11/2023 22:06:45	21/11/2023 22:26:24	
7	✓ Install statuschecker	success	ssh / ka-adh-3.ru-central1.internal	00:00:11	21/11/2023 21:55:34	21/11/2023 21:55:45	
6	✓ Install statuschecker	success	ssh / ka-adh-2.ru-central1.internal	00:00:11	21/11/2023 21:55:31	21/11/2023 21:55:42	
5	✓ Install statuschecker	success	ssh / ka-adh-1.ru-central1.internal	00:00:12	21/11/2023 21:55:28	21/11/2023 21:55:40	
4	✓ Check connection	success	ssh / ka-adh-2.ru-central1.internal	00:00:04	21/11/2023 21:54:39	21/11/2023 21:54:43	
3	✓ Check connection	success	ssh / ka-adh-3.ru-central1.internal	00:00:05	21/11/2023 21:49:54	21/11/2023 21:49:59	
2	Check connection	running	ssh / ka-adh-2.ru-central1.internal	00:00:00	21/11/2023 21:49:51	-	
1	✓ Check connection	success	ssh / ka-adh-1.ru-central1.internal	00:00:05	21/11/2023 21:49:47	21/11/2023 21:49:52	

Jobs page

On this page you can view action execution details, including verbose Ansible STDOUT/STDERR outputs. To view the execution log of an action, click the corresponding action name on the ***Jobs*** page. Action logs are stored for a configurable time period that can be changed in ADCM settings (Data Retention Policy).

TIP

Some actions have the ***Verbose*** flag that allows generating more detailed action execution logs.

For more information on actions for each ADH service, refer to the following links:

- Airflow
- Flink
- HBase
- HDFS
- Hive
- Impala
- Ozone
- Solr
- Spark3
- Trino
- YARN
- Zeppelin
- ZooKeeper

Arenadata Platform Security overview

Daria Barysheva

Arenadata Platform Security (ADPS) is a security framework that is based on Apache Ranger and Apache Knox. You can use ADPS to configure authorization and policy-based access control for ADH services.

Before ADPS installation, ensure the following requirements are met:

- The Arenadata Hyperwave cluster of version 2.1.3 or higher is installed and configured according to the Get started with Arenadata Hyperwave guide.
- All ADPS prerequisites are met:
  - Hardware requirements
  - Network requirements
  - Software requirements
- ADPS services should not use ADH cluster hosts. It is recommended to install ADPS on a separate virtual machine or a dedicated server.
- It is recommended to store audit data in both HDFS and Solr. The default configuration for Ranger Audits in Solr uses a shared Solr instance.
- If an external DBMS is used to store metadata (PostgreSQL/MySQL), it should be started and available. The Ranger installer creates two new users (default names: `rangeradmin` and `rangerlogger`) and two new databases (default names: `ranger` and `ranger_audit`).
- One of the following authentication types should be configured:
  - MIT Kerberos
  - MS Active Directory
  - FreeIPA

NOTE

For more information on ADPS installation, see Get started with Arenadata Platform Security.

After ADPS installation and integration with ADH (on the ***Import*** tab in the ADH cluster menu) you can start working with Ranger plugins available for some ADH services: HDFS, HBase, Hive, etc.

Ranger plugins for ADH

Sergei Tikhomirov

Contents

To Table of Contents

Overview

Access Ranger Web UI

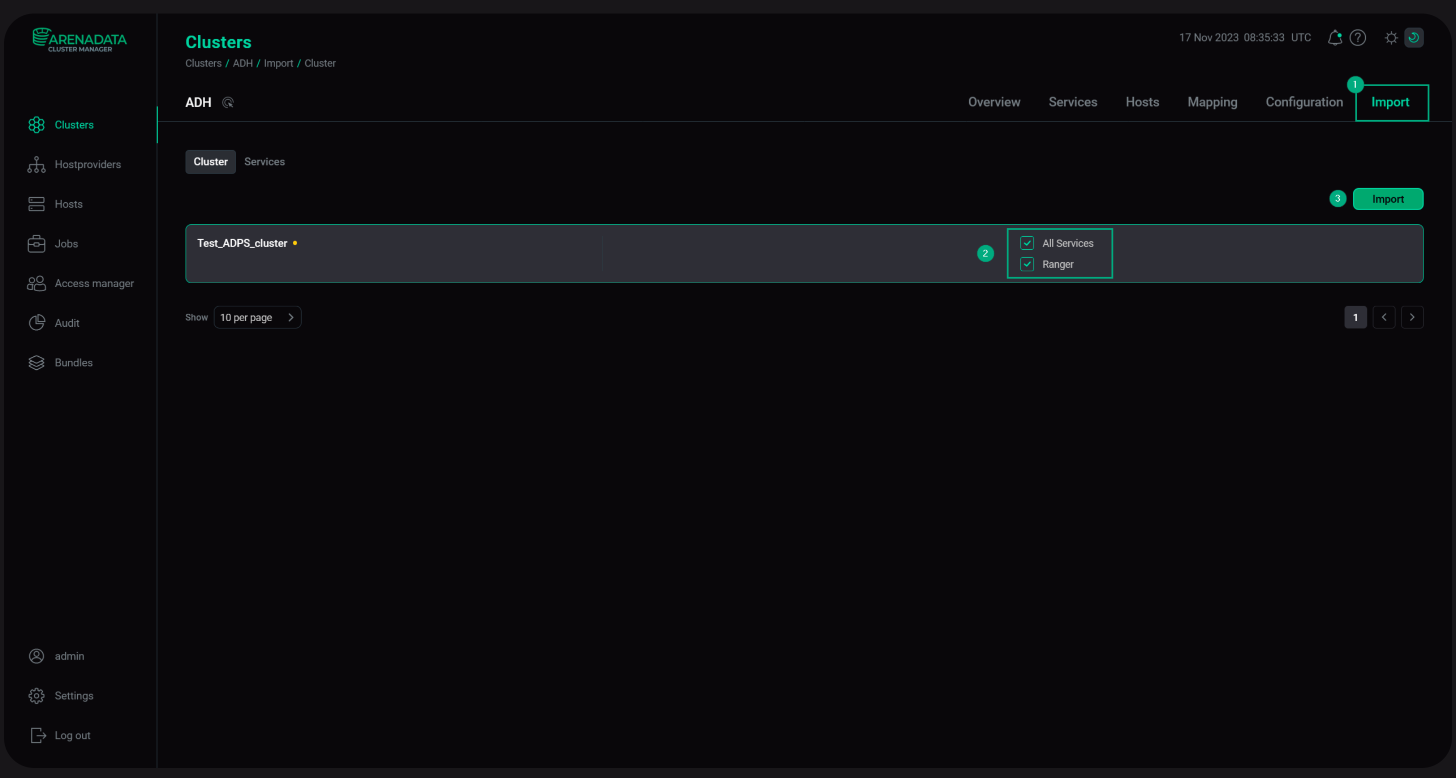
Overview

Ranger plugins are available for the following ADH services:

- HBase
- HDFS
- Hive
- Impala
- Ozone
- Solr
- Spark3
- Trino
- YARN

Before enabling plugins, you need to integrate your ADPS cluster with the ADH cluster.

1. On the ***Clusters*** page find your ADH cluster and click on its name. In the cluster menu, go to the ***Import*** tab.
2. On the ***Import*** tab, select the ***<ADPS_cluster_name>/Ranger*** checkbox to add your ADPS cluster and click ***Import***.

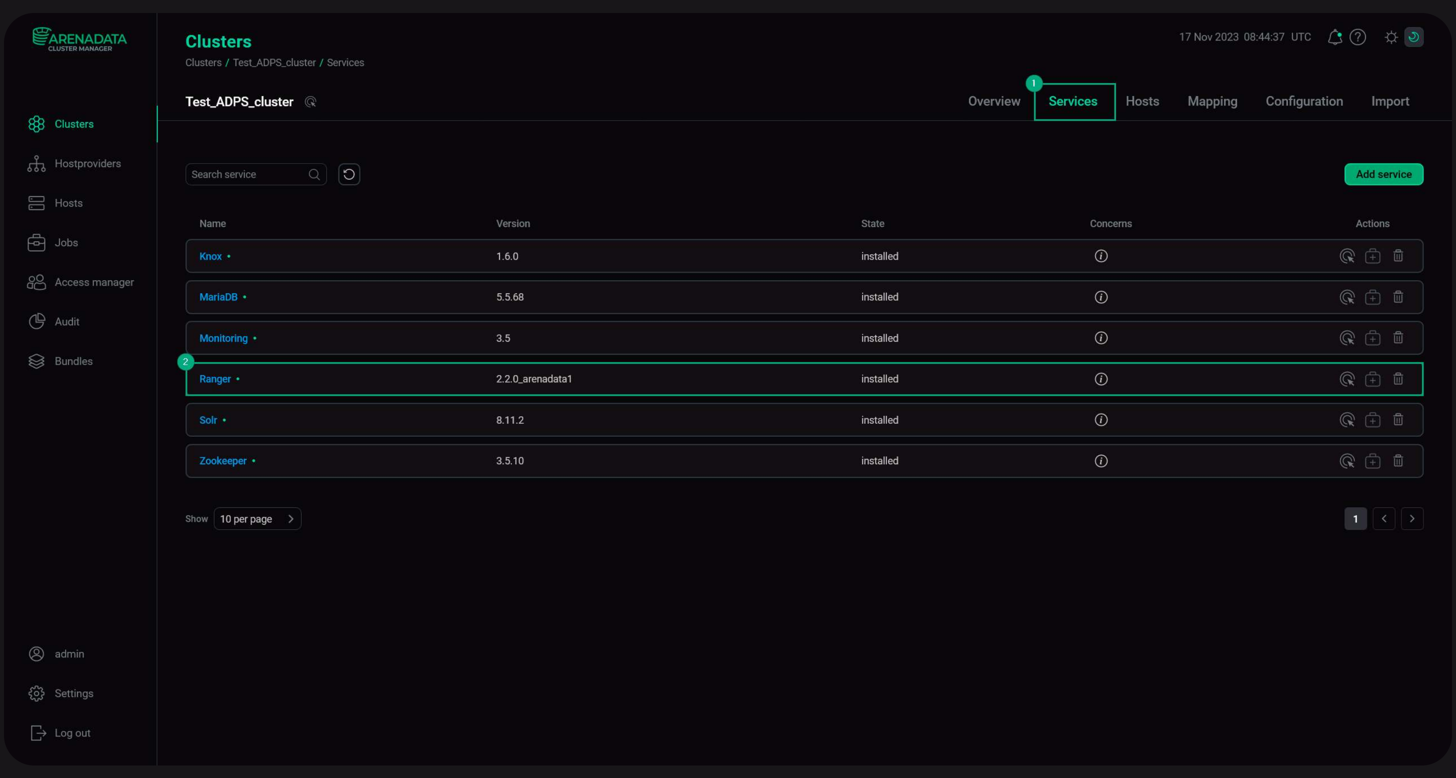


Ranger import

Access Ranger Web UI

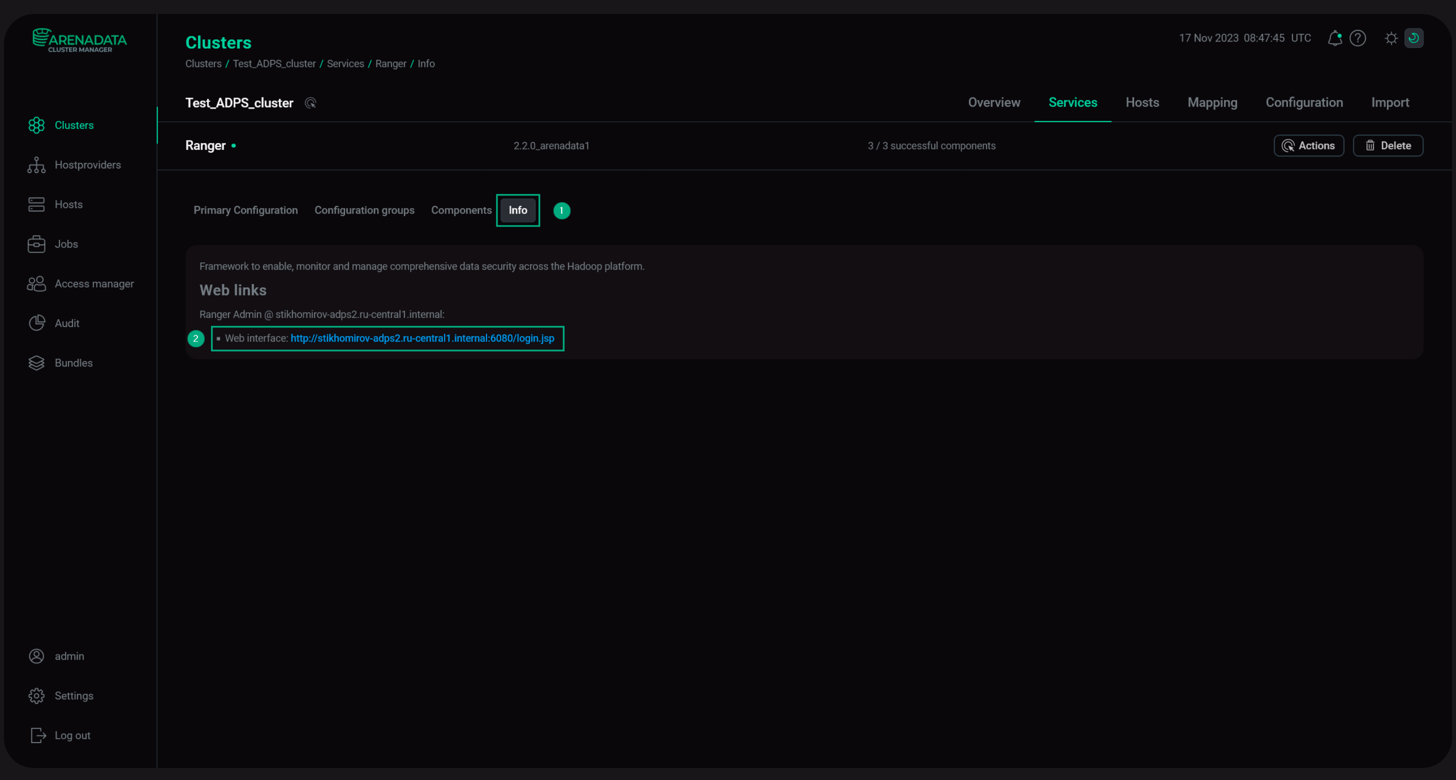
Ranger provides a web UI which allows you to manage policies and permissions. Follow the steps below to access it.

1. On the ***Clusters*** page, find the ADPS cluster associated with your ADH cluster and click on it. Open the ***Services*** tab and click on ***Ranger***.



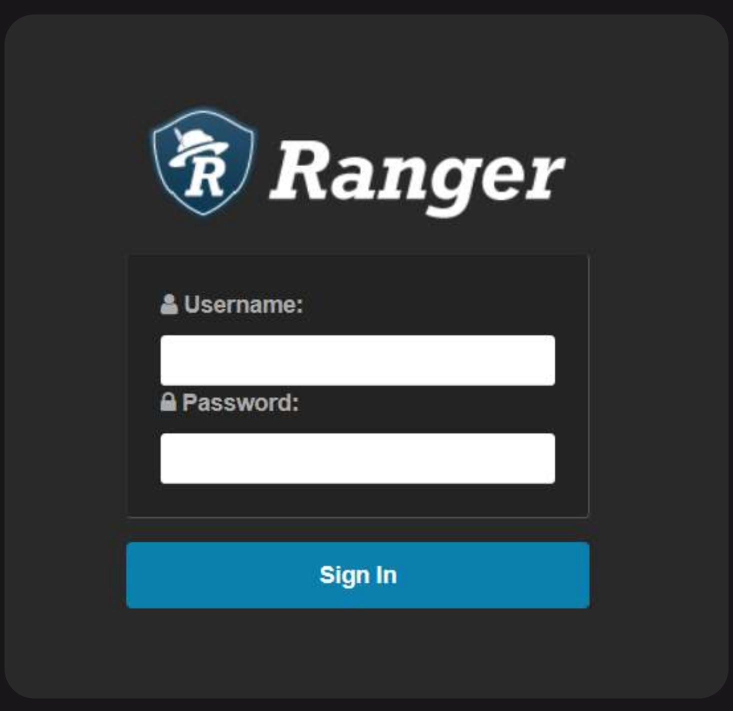
Ranger service

2. On the opened page, click on the link labeled as ***Web interface***.



Web interface link

3. Enter the admin credentials and click ***Sign in*** to authorize as Ranger admin.



Ranger sign-in

Passwords for accessing the Ranger web interface are set before Ranger installation. For more information, see the ***Credentials*** section in the Ranger configuration parameters.

Ranger HBase plugin

Sergei Tikhomirov

Contents

To Table of Contents

Enable HBase plugin

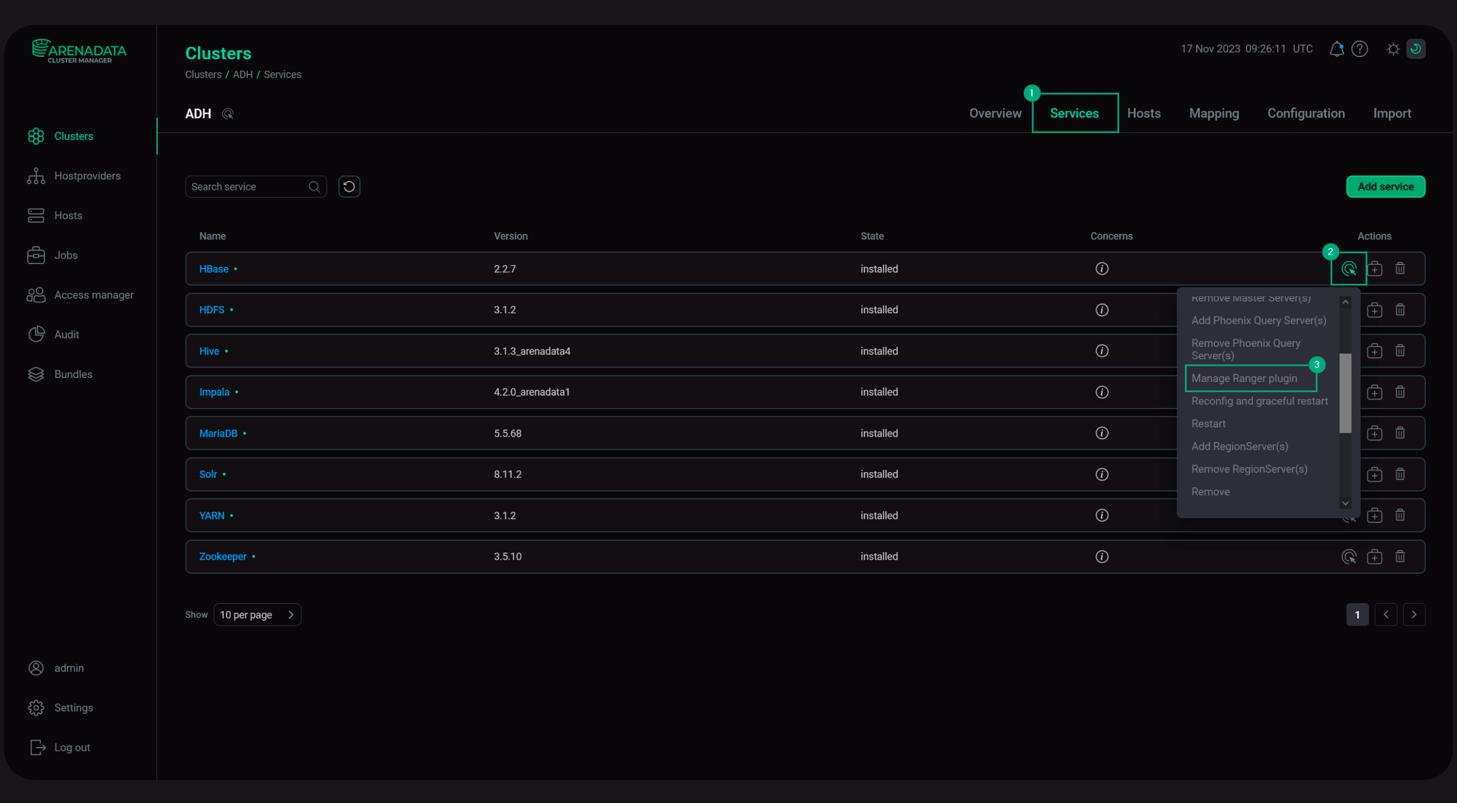
Add a new policy in Ranger

Enable HBase plugin

To enable the Ranger HBase plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.

2. Find HBase and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.

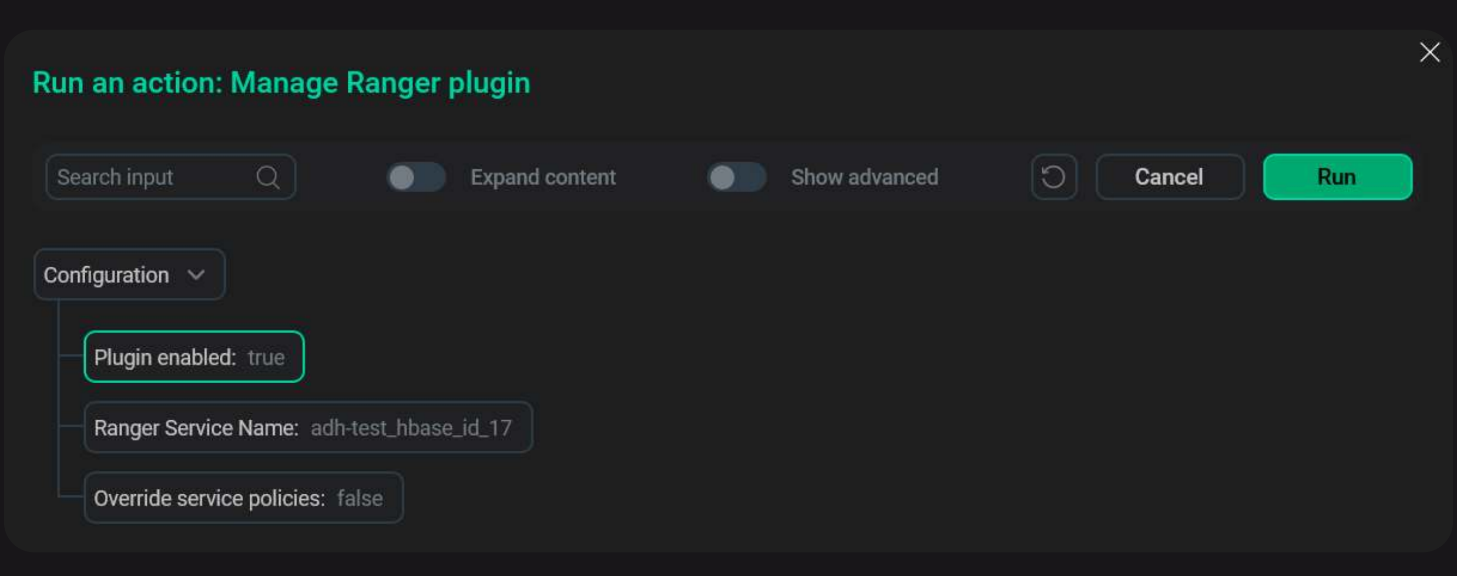


Name	Version	State
HBase	2.2.7	installed
HDFS	3.1.2	installed
Hive	3.1.3_arenadata4	installed
Impala	4.2.0_arenadata1	installed
MariaDB	5.5.68	installed
Solr	8.11.2	installed
YARN	3.1.2	installed
Zookeeper	3.5.10	installed

ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH / Services
17 Nov 2023 09:26:11 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
Concerns
Actions
Add Phoenix Query Server(s)
Remove Phoenix Query Server(s)
Manage Ranger plugin
Reconfig and graceful restart
Restart
Add RegionServer(s)
Remove RegionServer(s)
Remove
Show 10 per page
1
2
3

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.



Run an action: Manage Ranger plugin

Configuration

Plugin enabled: true

Ranger Service Name: adh-test_hbase_id_17

Override service policies: false

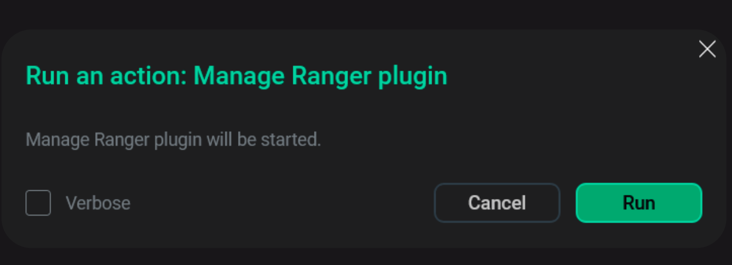
Search input
Expand content
Show advanced
Cancel
Run

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin

Manage Ranger plugin will be started.

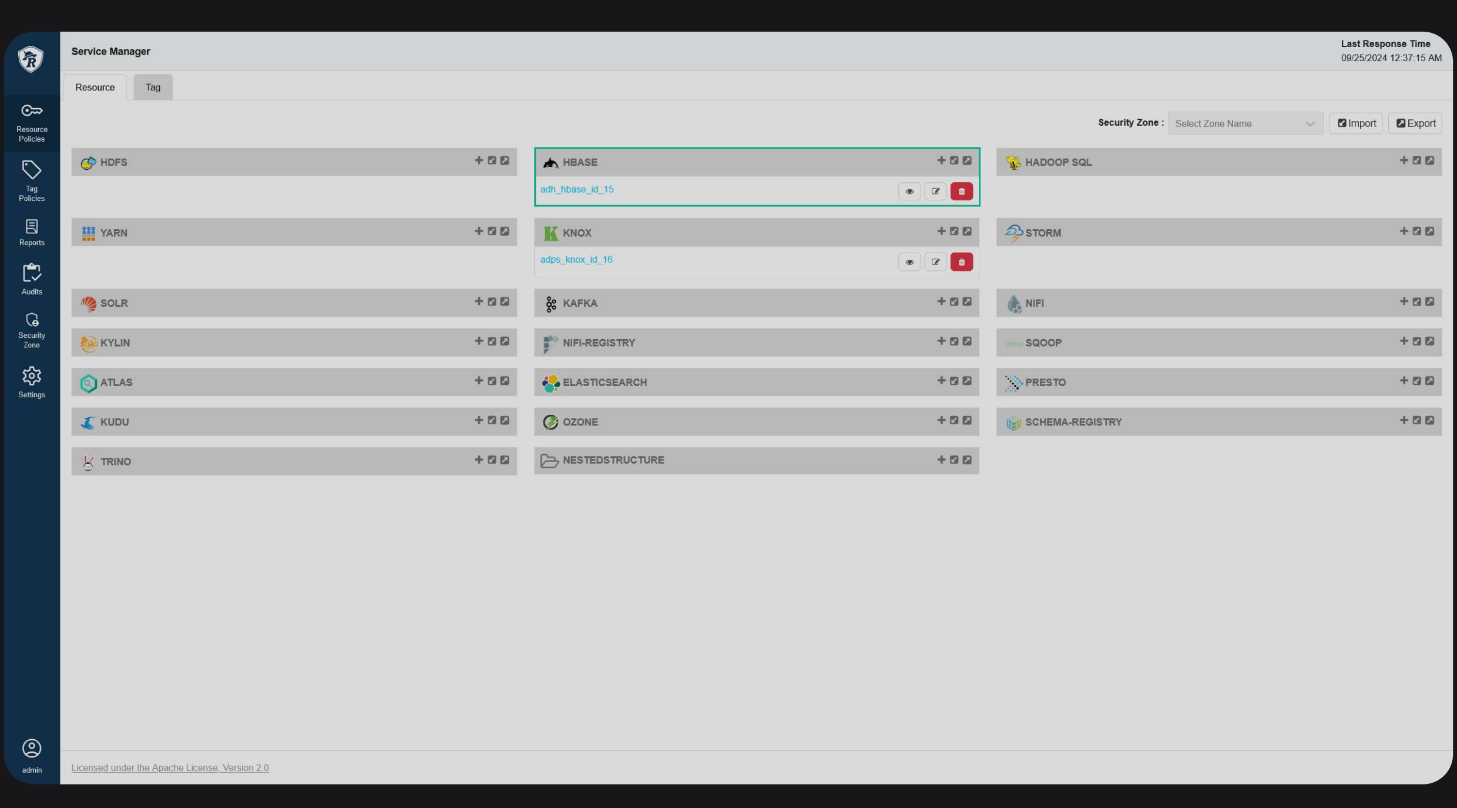
☐ Verbose
Cancel
Run

Action confirmation

Add a new policy in Ranger

To add a new policy to an existing HBase service, you should perform the following actions:

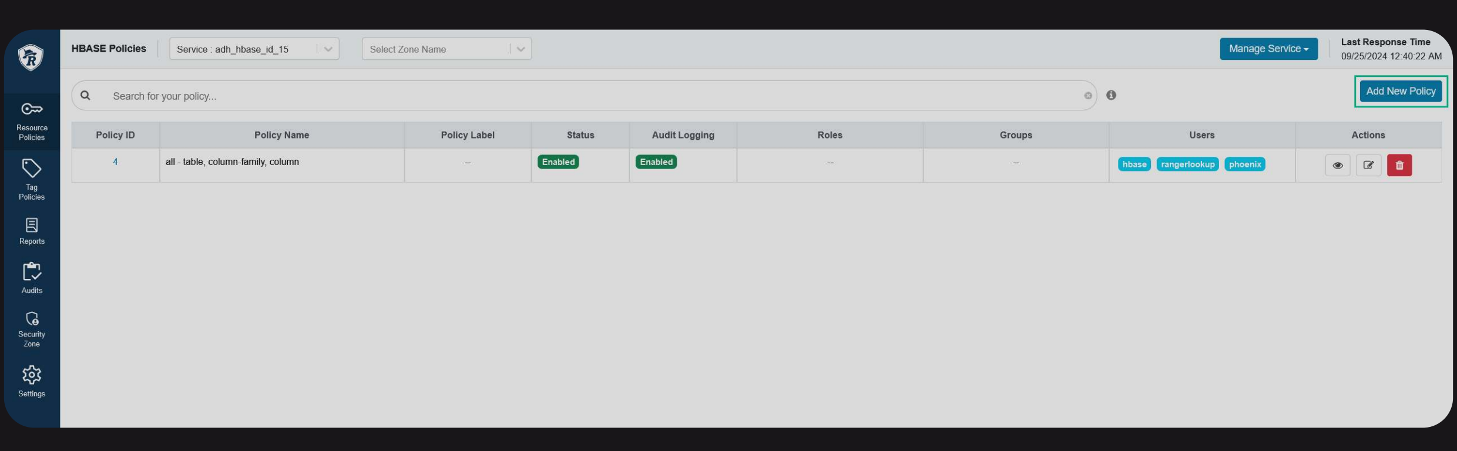
1. On the ***Service Manager*** page, click an existing service in the HBase pane.



Service Manager
Resource
Tag
Security Zone :
Import
Export
HDFS
HBASE
HADOOP SQL

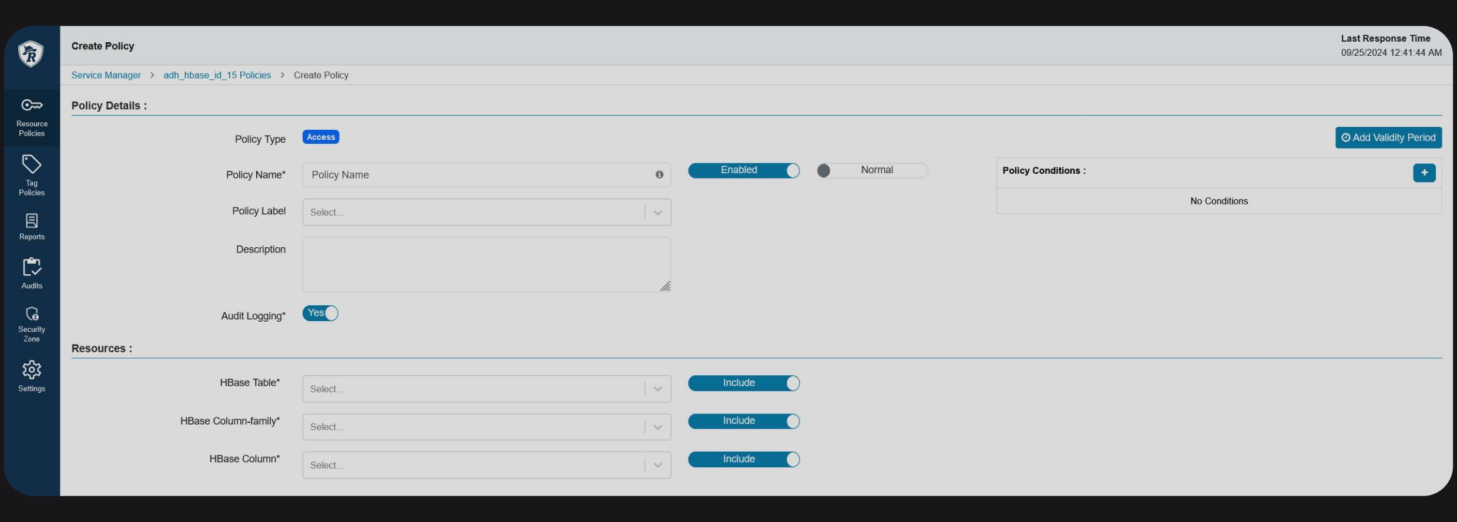
Service Manager

2. On the HBase policy page, click ***Add New Policy***.



Add new policy

3. On the opened ***Create Policy*** page, fill in the required details.



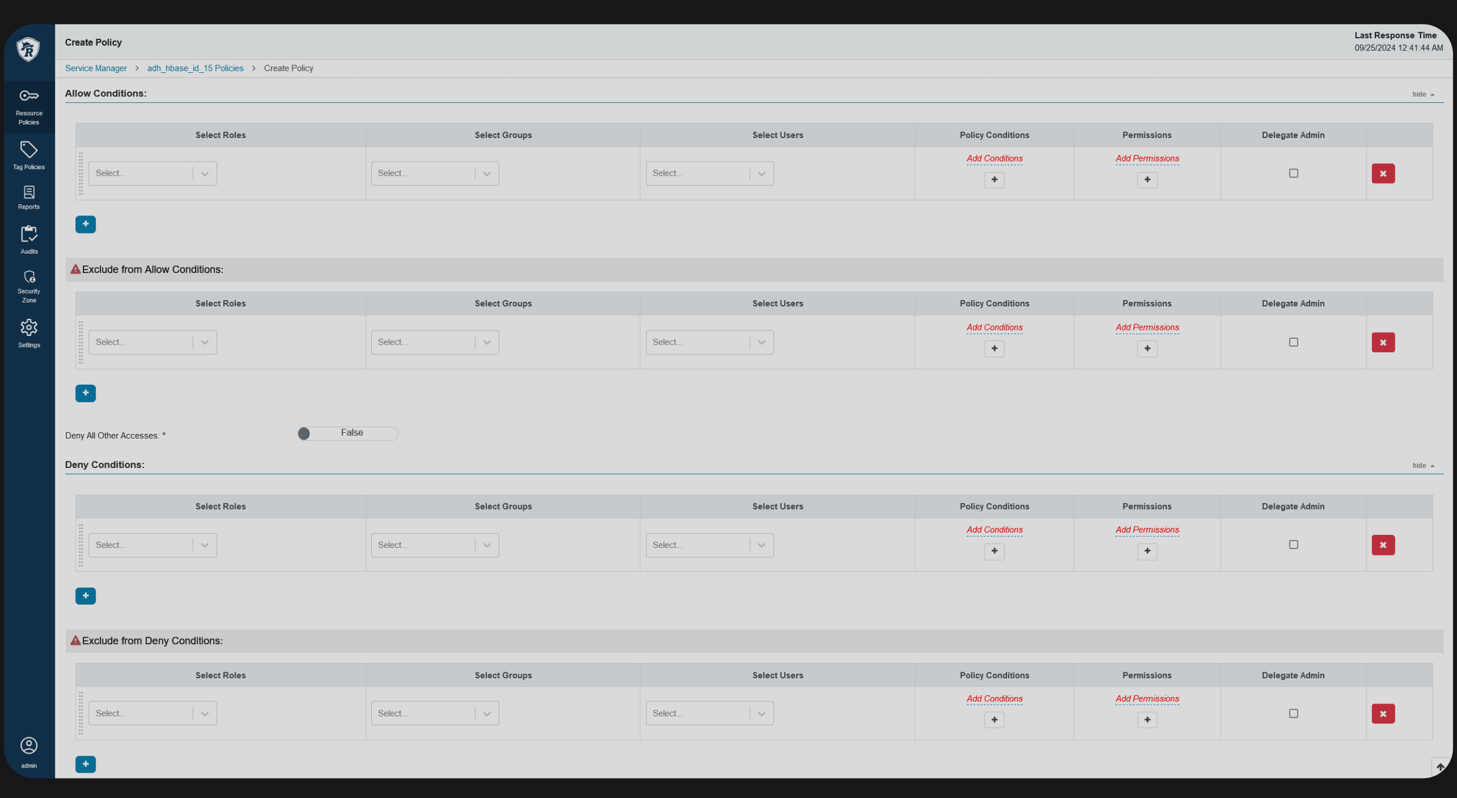
Create Policy
Service Manager > adh_hbase_id_15 Policies > Create Policy
Last Response Time 09/25/2024 12:41:44 AM
Policy Details :
Policy Type
Access
Add Validity Period
Policy Name*
Enabled
Normal
Policy Conditions :
No Conditions
Policy Label
Description
Audit Logging*
Yes
Resources :
HBase Table*
HBase Column-family*
HBase Column*
Include
Resource Policies
Tag Policies
Reports
Audits
Security Zone
Settings

HBase policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
HBase Table	Name of an HBase table
HBase Column-family	Name of an HBase column family
HBase Column	Name of an HBase column
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click +. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Create Policy
Service Manager > adh_hbase_id_15 Policies > Create Policy
Last Response Time 09/25/2024 12:41:44 AM
Allow Conditions:
Exclude from Allow Conditions:
Deny All Other Accesses: *
False
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Add Conditions
Add Permissions

HBase allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger HDFS plugin

Sergei Tikhomirov

Contents

To Table of Contents

Overview

Enable HDFS plugin

Add a new policy in Ranger

Overview

Apache Ranger configuration allows you to apply both Ranger policies and HDFS permissions. When a NameNode receives a user request, the Ranger plugin checks the policies set via the Ranger Service Manager, and if there are none, checks the permissions set in HDFS.

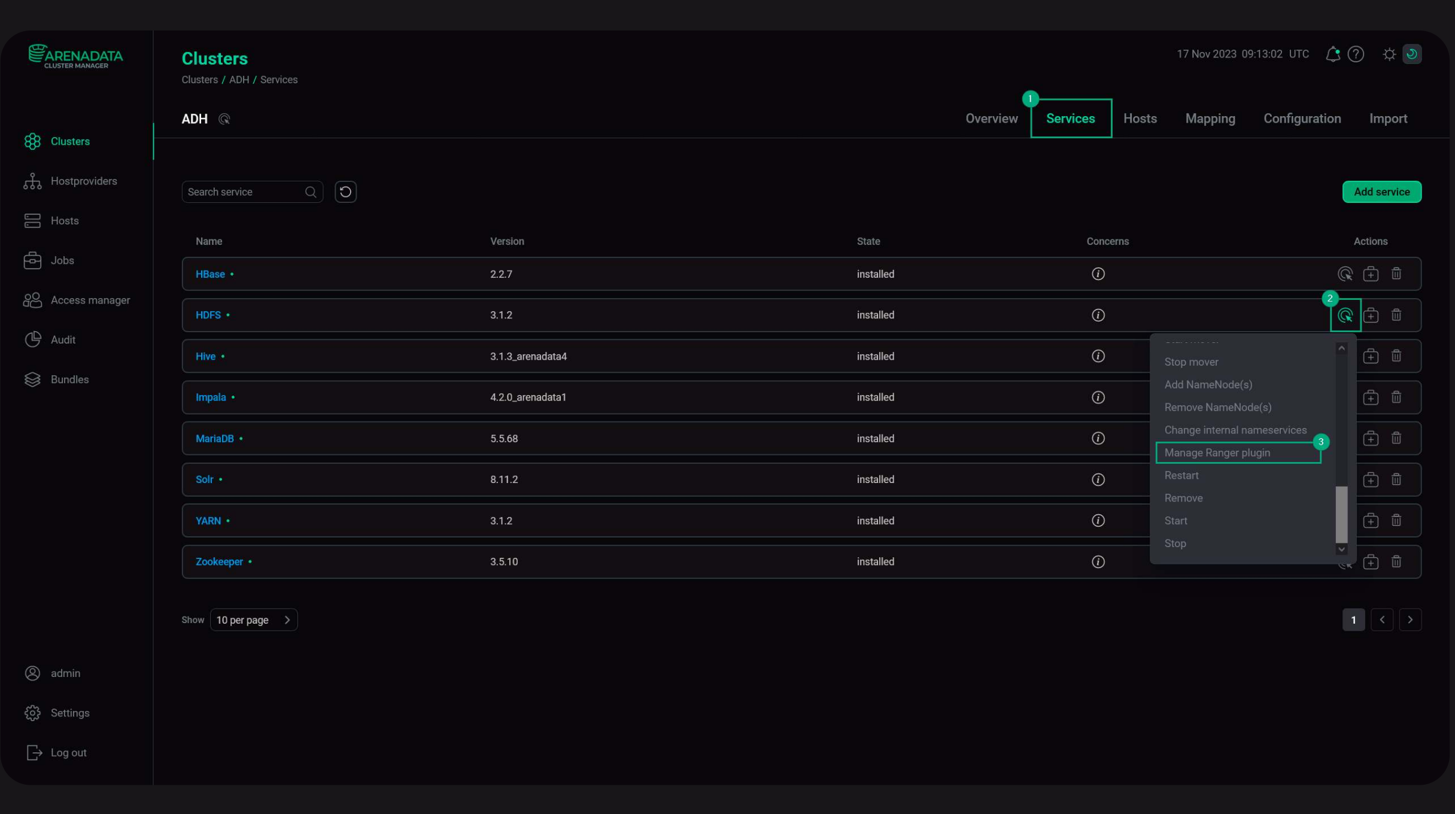
NOTE

It's recommended to create permissions in Ranger Service Manager and have limited permissions at the HDFS level.

Enable HDFS plugin

To enable the Ranger HDFS plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.
2. Find HDFS and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.

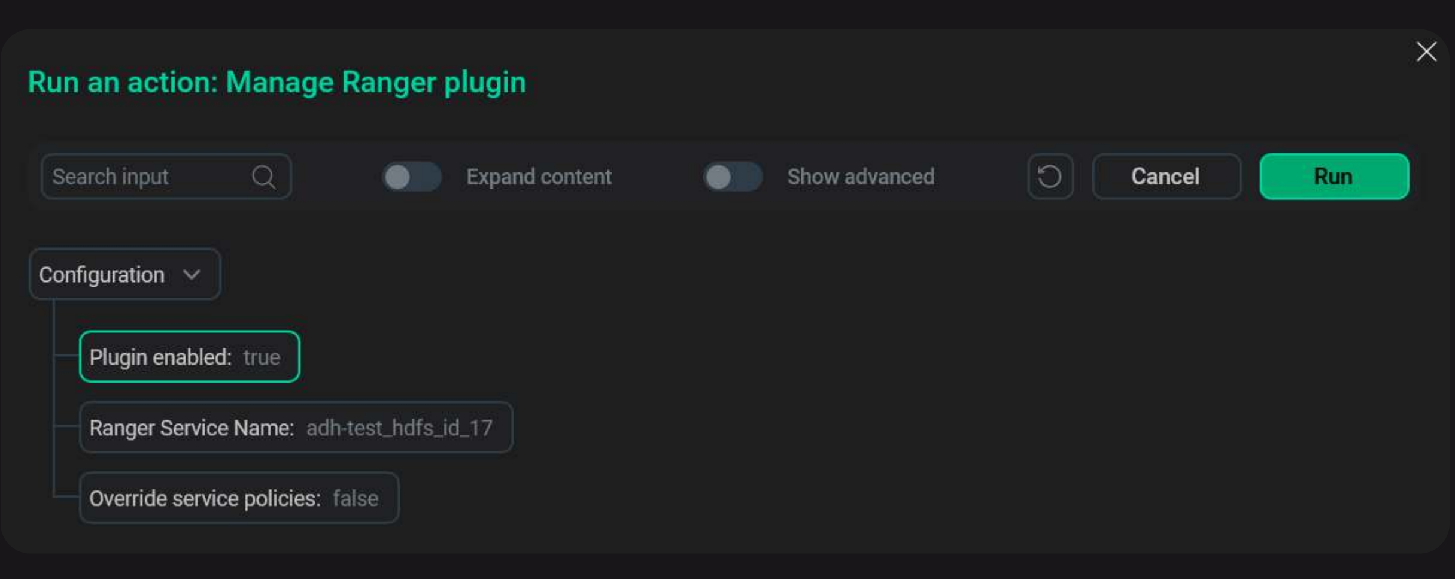


Name	Version	State
HBase	2.2.7	installed
HDFS	3.1.2	installed
Hive	3.1.3_arenadata4	installed
Impala	4.2.0_arenadata1	installed
MariaDB	5.5.68	installed
Solr	8.11.2	installed
YARN	3.1.2	installed
Zookeeper	3.5.10	installed

Clusters
Clusters / ADH / Services
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Add service
Stop mover
Add NameNode(s)
Remove NameNode(s)
Change internal nameservices
Manage Ranger plugin
Restart
Remove
Start
Stop

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.



Run an action: Manage Ranger plugin

Plugin enabled: true

Ranger Service Name: adh-test_hdfs_id_17

Override service policies: false

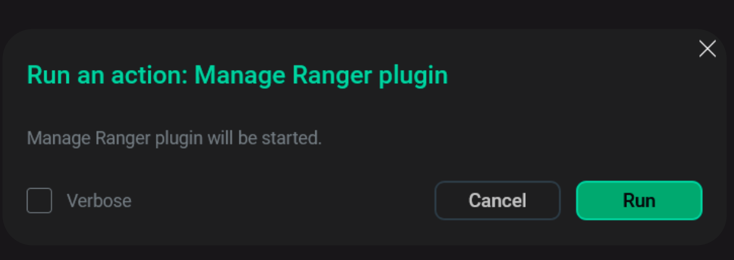
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin

Manage Ranger plugin will be started.

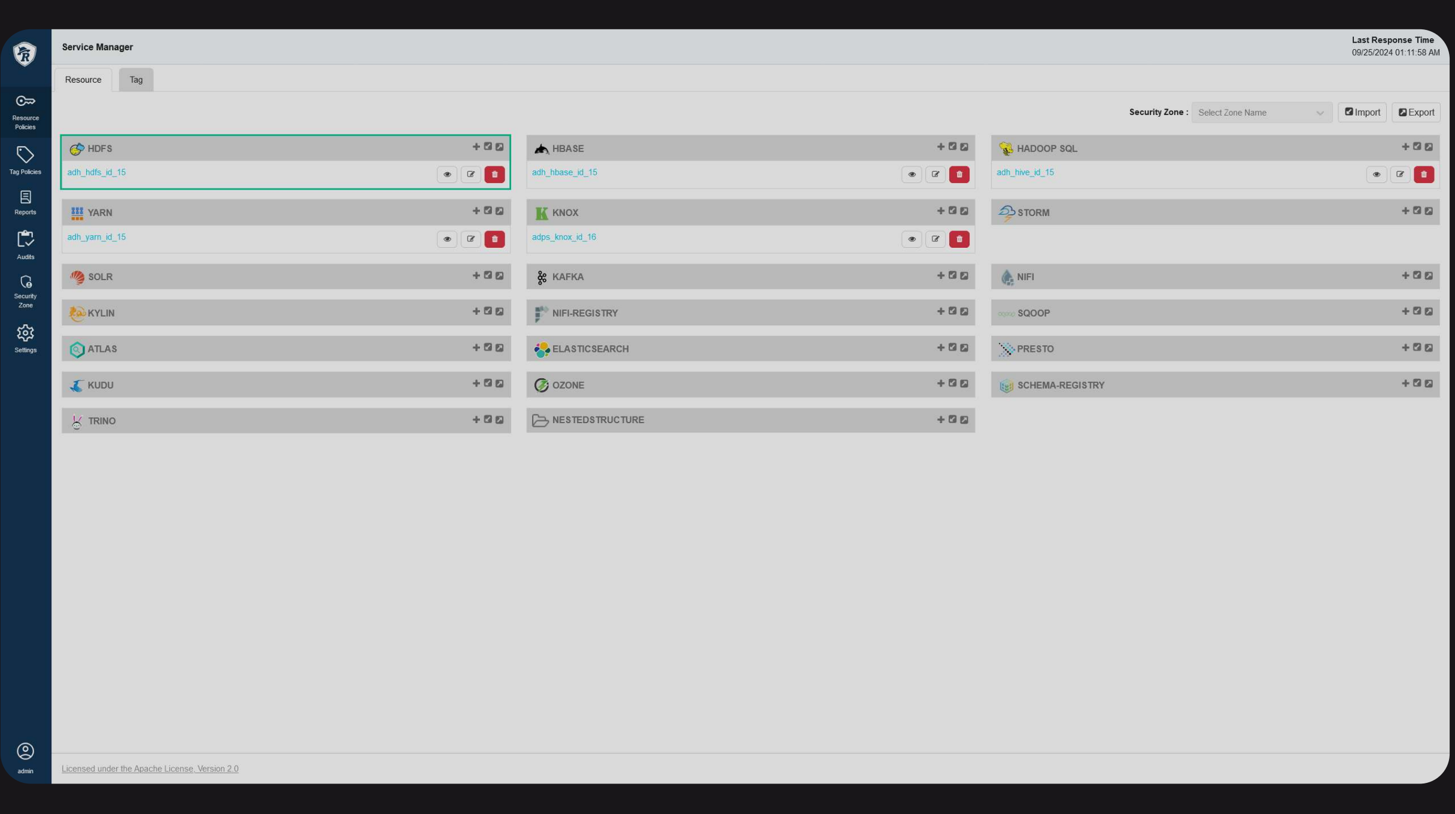
Verbose
Cancel
Run

Action confirmation

Add a new policy in Ranger

To add a new policy to an existing HDFS service, you should perform the following actions:

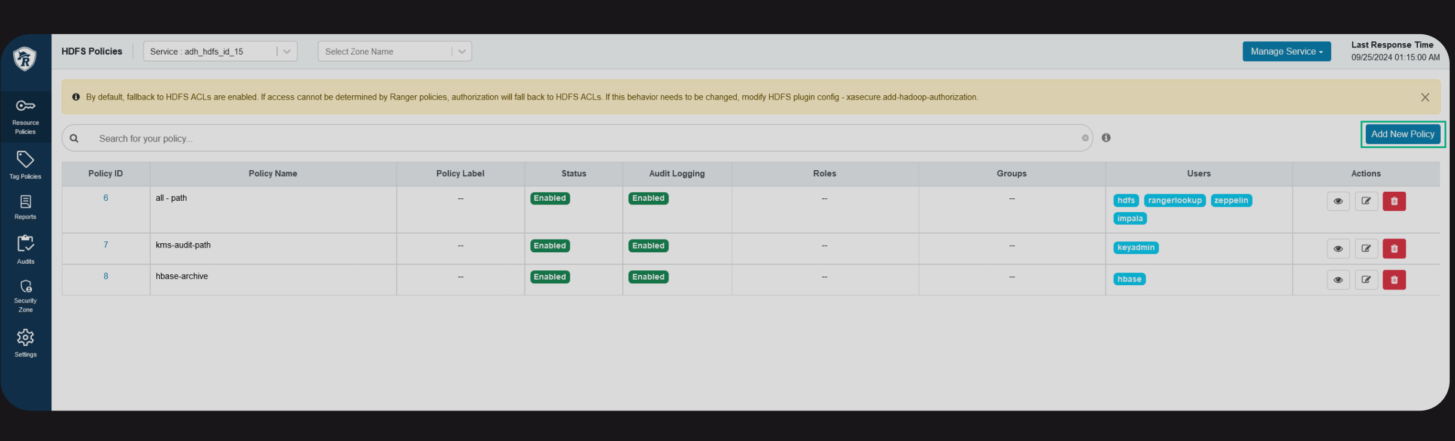
1. On the ***Service Manager*** page, click an existing service in the HDFS pane.



Service Manager
Resource
Tag
HDFS
HBASE
HADOOP SQL
YARN
KNOX
STORM
SOLR
KAFKA
NIFI
KYLIN
NIFI-REGISTRY
SQOOP
ATLAS
ELASTICSEARCH
PRESTO
KUDU
OZONE
SCHEMA-REGISTRY
TRINO
NESTEDSTRUCTURE

The Service Manager page

2. On the HDFS policy page, click ***Add New Policy***.



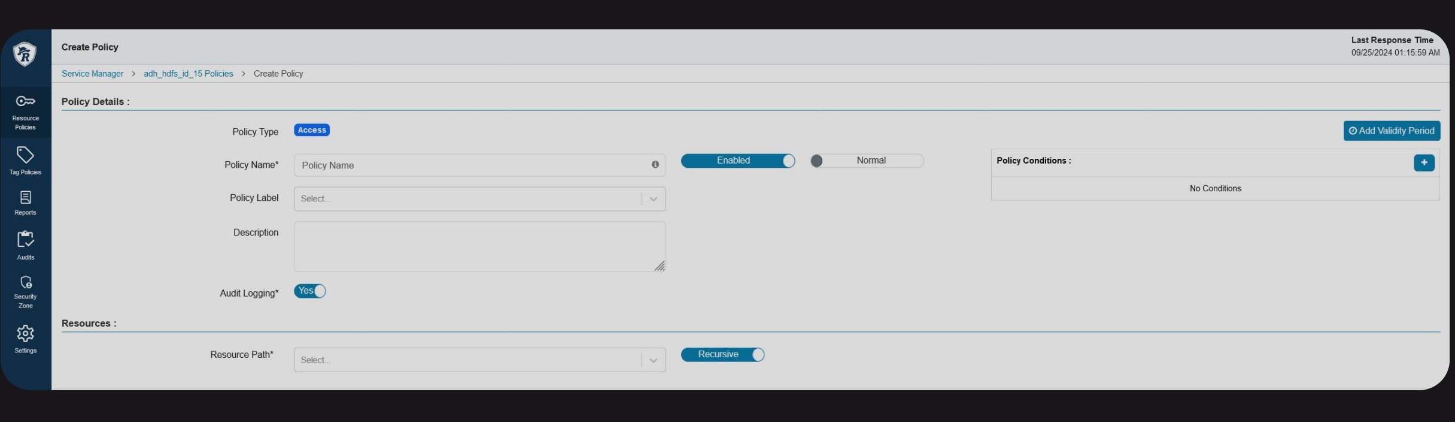
By default, fallback to HDFS ACLs are enabled. If access cannot be determined by Ranger policies, authorization will fall back to HDFS ACLs. If this behavior needs to be changed, modify HDFS plugin config - xasecure.add-hadoop-authorization

Policy ID	Policy Name	Policy Label	Status	Audit Logging	Roles	Groups	Users
6	all - path	--	Enabled	Enabled	--	--	hdfs rangerlookup zeppelin impala
7	kms-audit-path	--	Enabled	Enabled	--	--	keyadmin
8	hbase-archive	--	Enabled	Enabled	--	--	hbase

HDFS Policies
Service : adh_hdfs_id_15
Manage Service
Add New Policy

Add new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.



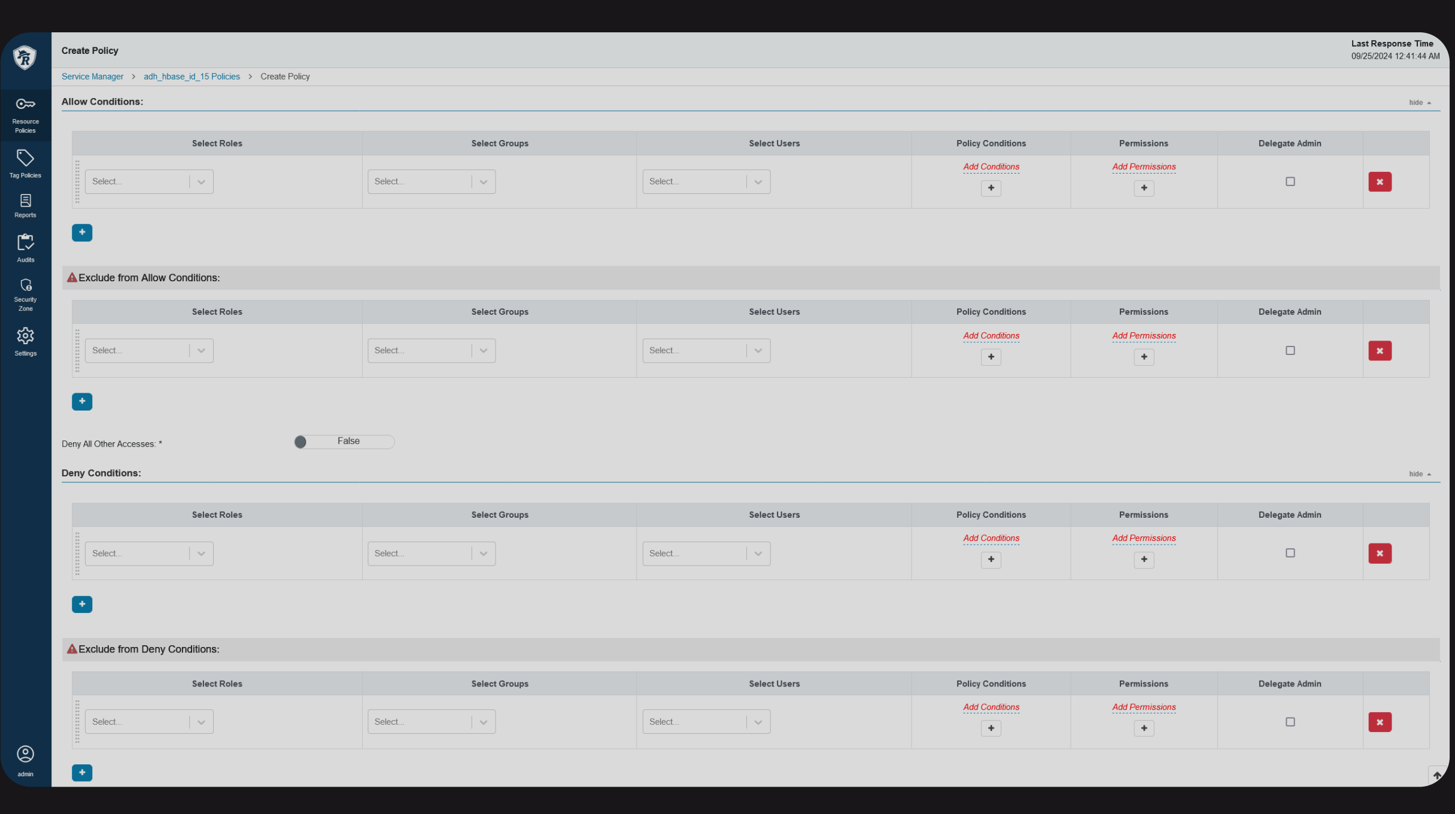
Create Policy
Policy Details :
Policy Type
Access
Policy Name*
Enabled
Normal
Policy Label
Description
Audit Logging*
Add Validity Period
Policy Conditions :
No Conditions
Resources :
Resource Path*
Recursive

HDFS policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
Resource Path	Defines the resource path for the policy folder/file. To avoid the need to specify the full path or include a policy for all subfolders or files, you can fill in this field using wildcards (for example, `/home`) or specify that the policy should be recursive. Wildcards can be included in the resource path, database name, table, or column: `\*` — indicates zero or more characters; `?` — indicates one character
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click +. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Deny All Other Accesses: * False

Create Policy
Allow Conditions:
Exclude from Allow Conditions:
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Add Conditions
Add Permissions

HDFS allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger Hive plugin

Sergei Tikhomirov

Contents

To Table of Contents

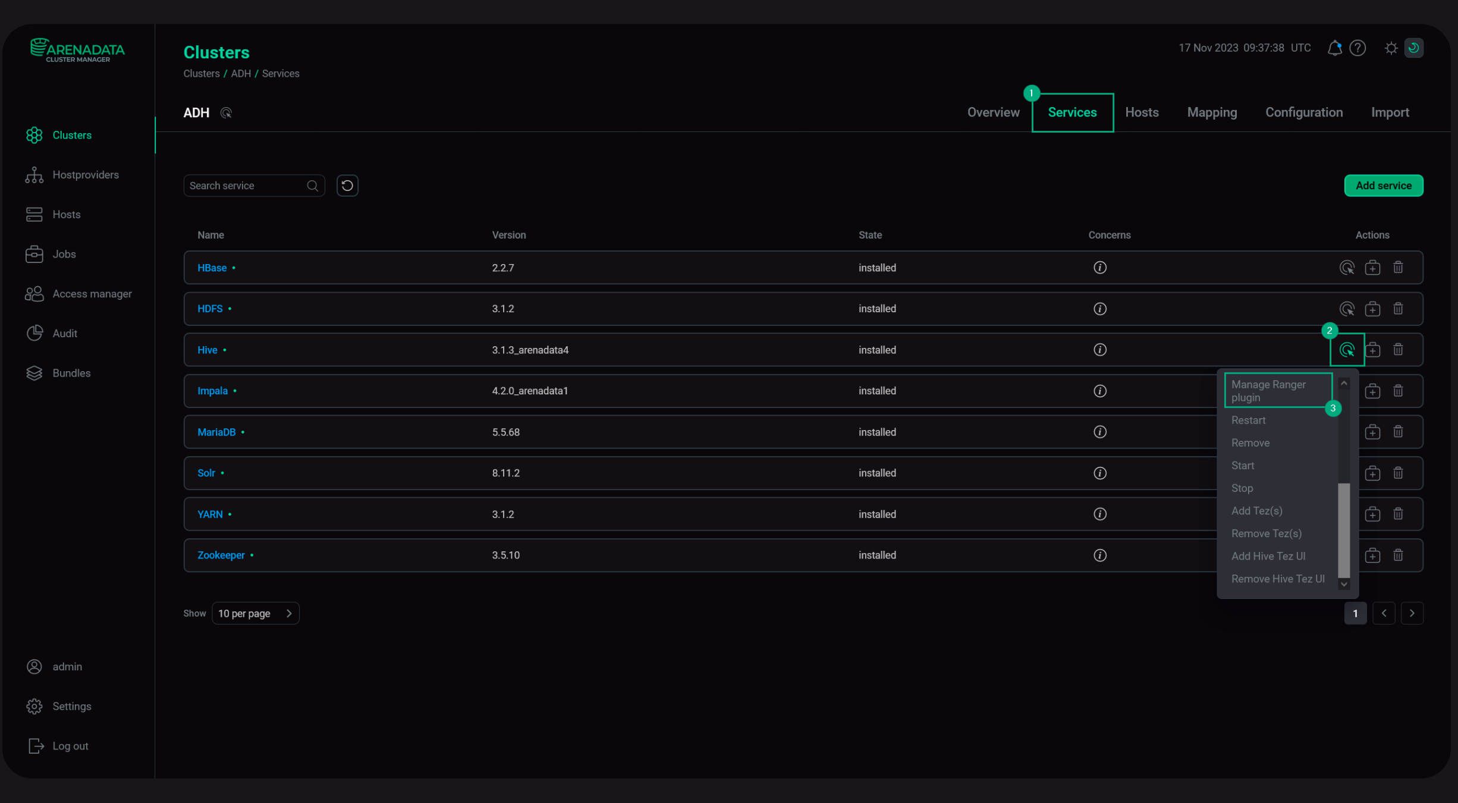
Enable Hive plugin

Add a new policy in Ranger

Enable Hive plugin

To enable the Ranger Hive plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.
2. Find Hive and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.



Clusters

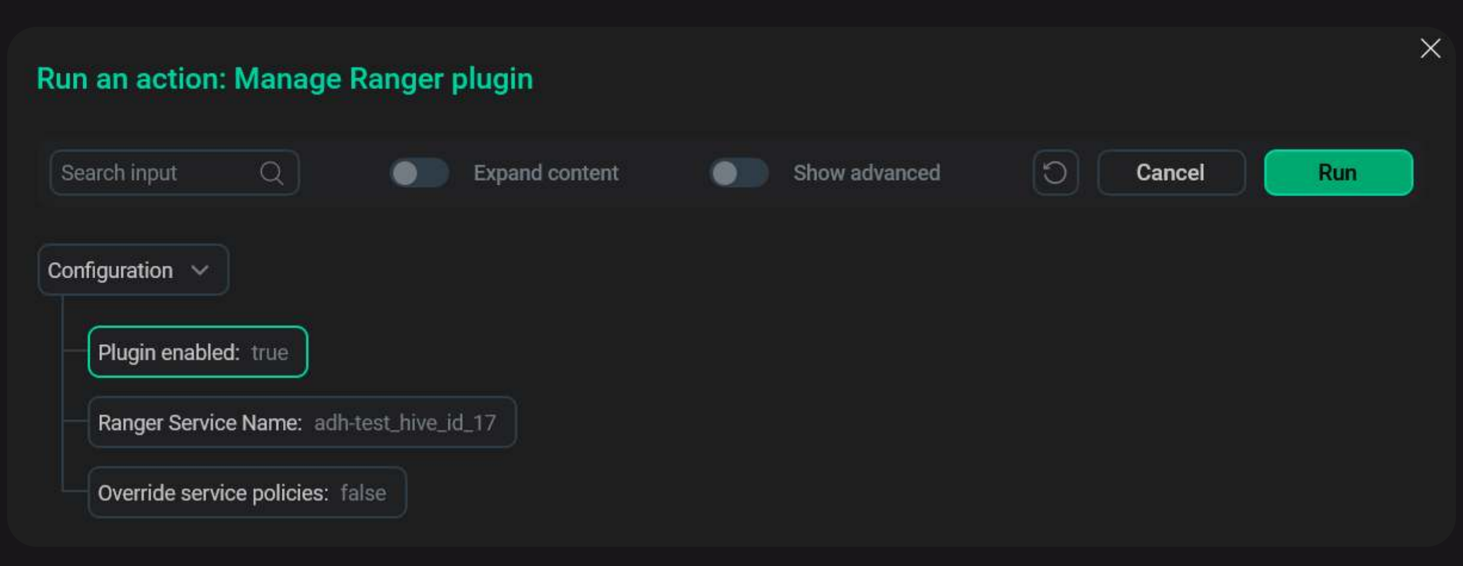
Clusters / ADH / Services

Name	Version	State
HBase	2.2.7	installed
HDFS	3.1.2	installed
Hive	3.1.3_arenadata4	installed
Impala	4.2.0_arenadata1	installed
MariaDB	5.5.68	installed
Solr	8.11.2	installed
YARN	3.1.2	installed
Zookeeper	3.5.10	installed

ARENADATA CLUSTER MANAGER
17 Nov 2023 09:37:38 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add service
Concerns
Actions
Manage Ranger plugin
Restart
Remove
Start
Stop
Add Tez(s)
Remove Tez(s)
Add Hive Tez UI
Remove Hive Tez UI
Show 10 per page
admin
Settings
Log out
1
2
3

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.



Run an action: Manage Ranger plugin

Plugin enabled: true

Ranger Service Name: adh-test_hive_id_17

Override service policies: false

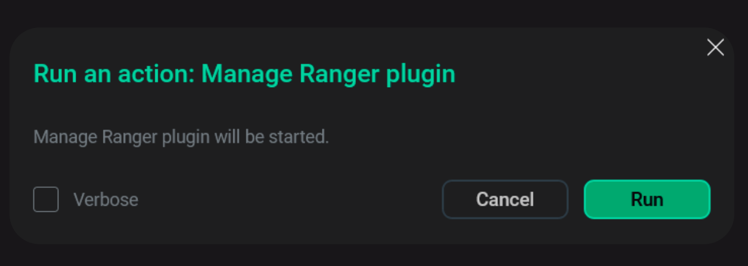
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin

Manage Ranger plugin will be started.

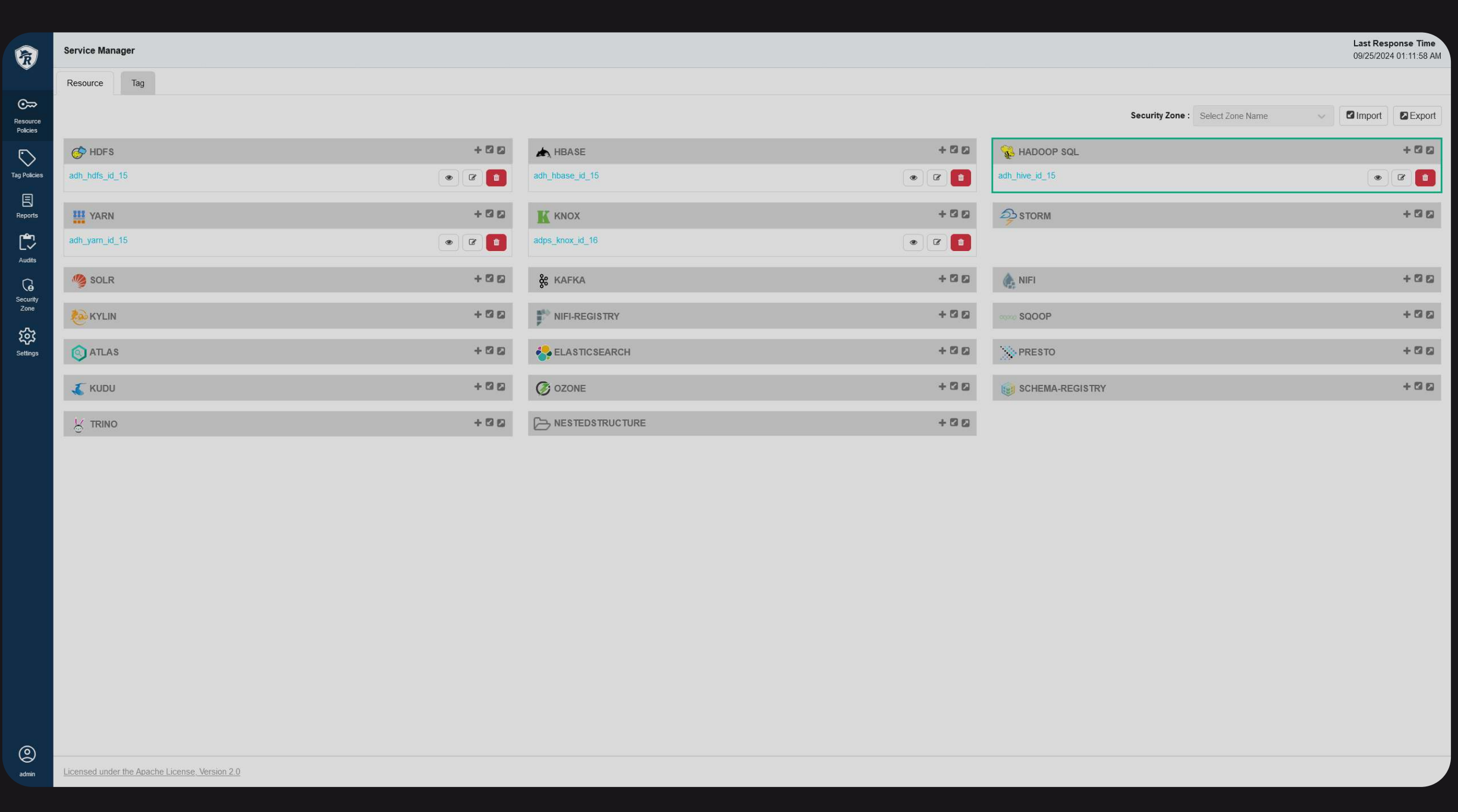
☐ Verbose
Cancel
Run

Action confirmation

Add a new policy in Ranger

To add a new policy to an existing Hive service, you should perform the following actions:

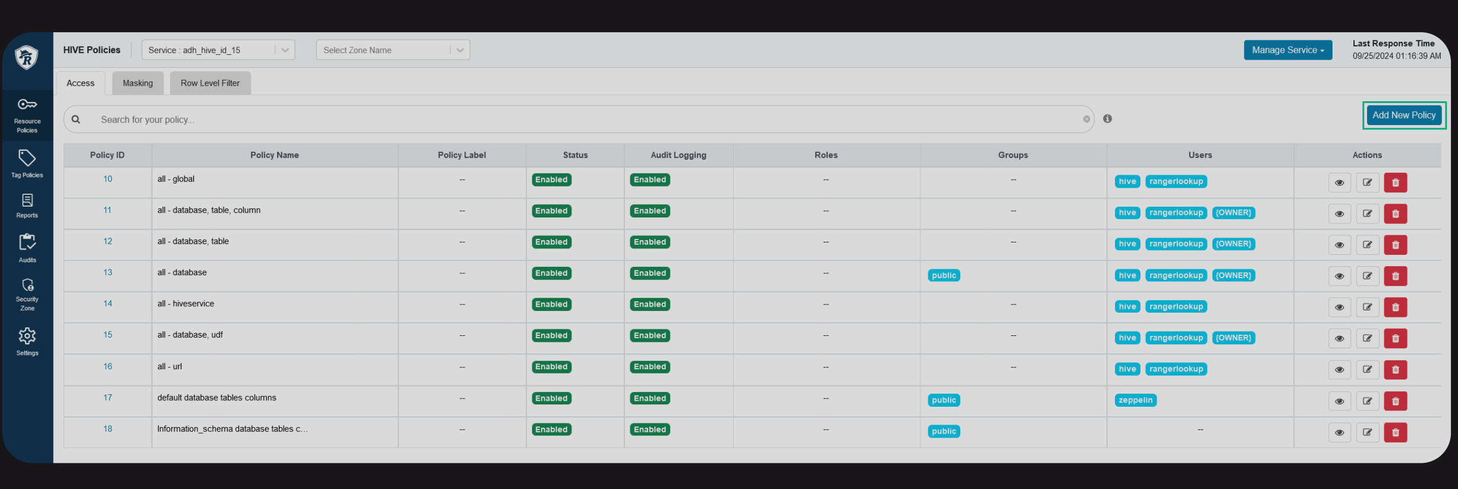
1. On the ***Service Manager*** page, click an existing Hive service in the Hadoop SQL pane.



Service Manager
Resource
Tag
Security Zone :
Import
Export
HDFS
HBASE
HADOOP SQL
YARN
KNOX
STORM
ATLAS
ELASTICSEARCH
PRESTO
KUDU
OZONE
SCHEMA-REGISTRY
TRINO
NESTEDSTRUCTURE
Licensed under the Apache License, Version 2.0

Service Manager

2. On the Hive policy page, click ***Add New Policy***.



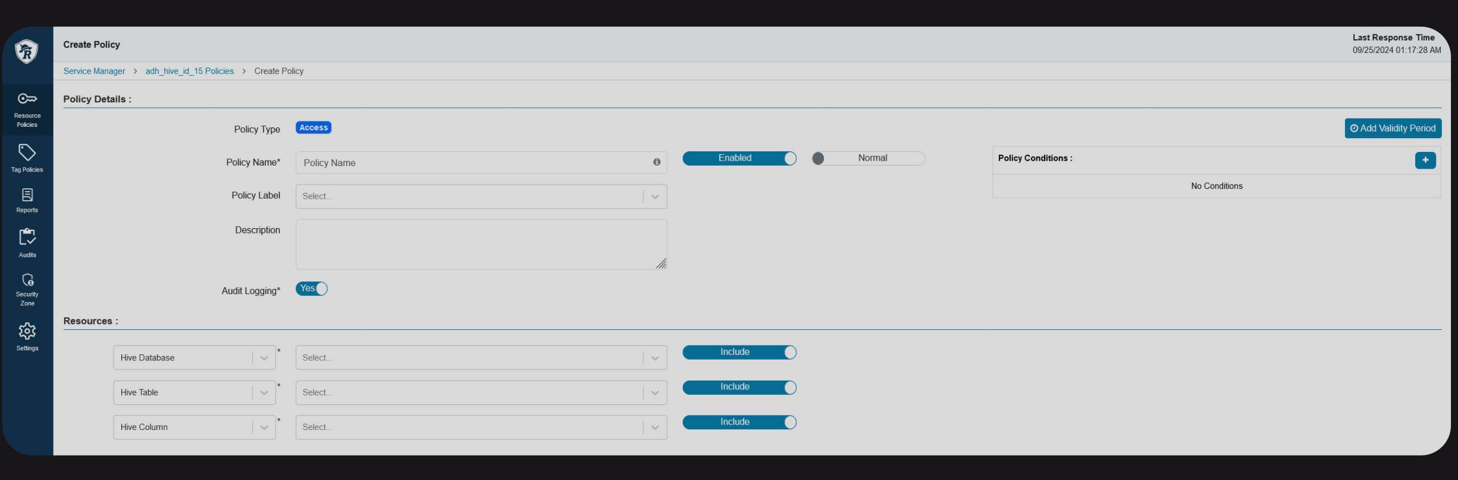
HIVE Policies Service : adh_hive_id_15

Policy ID	Policy Name	Policy Label	Status	Audit Logging	Roles	Groups	Users
10	all - global	--	Enabled	Enabled	--	--	hive rangerlookup
11	all - database, table, column	--	Enabled	Enabled	--	--	hive rangerlookup {OWNER}
12	all - database, table	--	Enabled	Enabled	--	--	hive rangerlookup {OWNER}
13	all - database	--	Enabled	Enabled	--	public	hive rangerlookup {OWNER}
14	all - hiveservice	--	Enabled	Enabled	--	--	hive rangerlookup
15	all - database, udf	--	Enabled	Enabled	--	--	hive rangerlookup {OWNER}
16	all - url	--	Enabled	Enabled	--	--	hive rangerlookup
17	default database tables columns	--	Enabled	Enabled	--	public	zeppelin
18	Information_schema database tables c...	--	Enabled	Enabled	--	public	--

Select Zone Name
Manage Service
Last Response Time 09/25/2024 01:16:39 AM
Access
Masking
Row Level Filter
Search for your policy...
Add New Policy
Actions

Add new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.



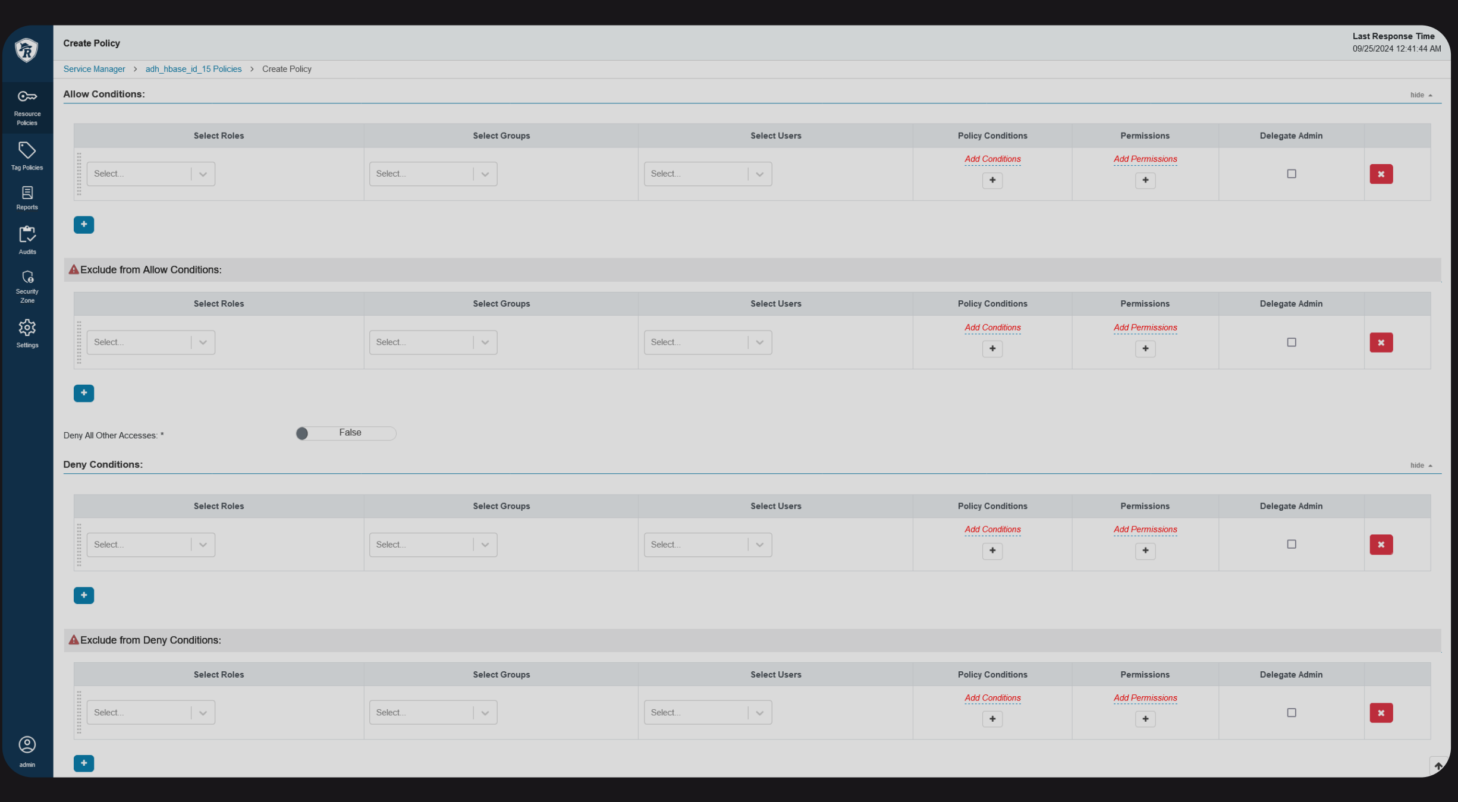
Create Policy
Last Response Time 09/25/2024 01:17:28 AM
Service Manager > adh_hive_id_15 Policies > Create Policy
Policy Details :
Policy Type Access
Add Validity Period
Policy Name*
Enabled
Normal
Policy Conditions :
No Conditions
Policy Label
Description
Audit Logging* Yes
Resources :
Hive Database
Hive Table
Hive Column
Include

Hive policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
database/url/hiveservice/global	If ***database*** is selected, specify the applicable database name. The autocomplete feature displays available databases based on the entered text. If ***url*** is selected, specify the cloud storage path (e.g., *s3a://dev-admin/demo/campaigns.txt*) where the end-user permission is needed to read/write the Hive data from/to a cloud storage path. The ***hiveservice*** option enables a user who has Service Admin permission in Ranger to run the stop query API: `kill query <queryID>`. Should be used only when Permissions=Service Admin. Select ***global*** to create global policies that use matching patterns and define the desired default access
table/udf/none	Specifies whether this policy is table-based or UDF-based
column/none	Specifies the applicable Hive column name. Failure to set this parameter will result in the error message `HiveAccessControlException`
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click +. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Create Policy
Last Response Time 09/25/2024 12:41:44 AM
Service Manager > adh_hbase_id_15 Policies > Create Policy
Allow Conditions:
Exclude from Allow Conditions:
Deny All Other Accesses: * False
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Add Conditions
Add Permissions

Hive allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger Impala plugin

Sergei Tikhomirov

Contents

To Table of Contents

Enable Impala plugin

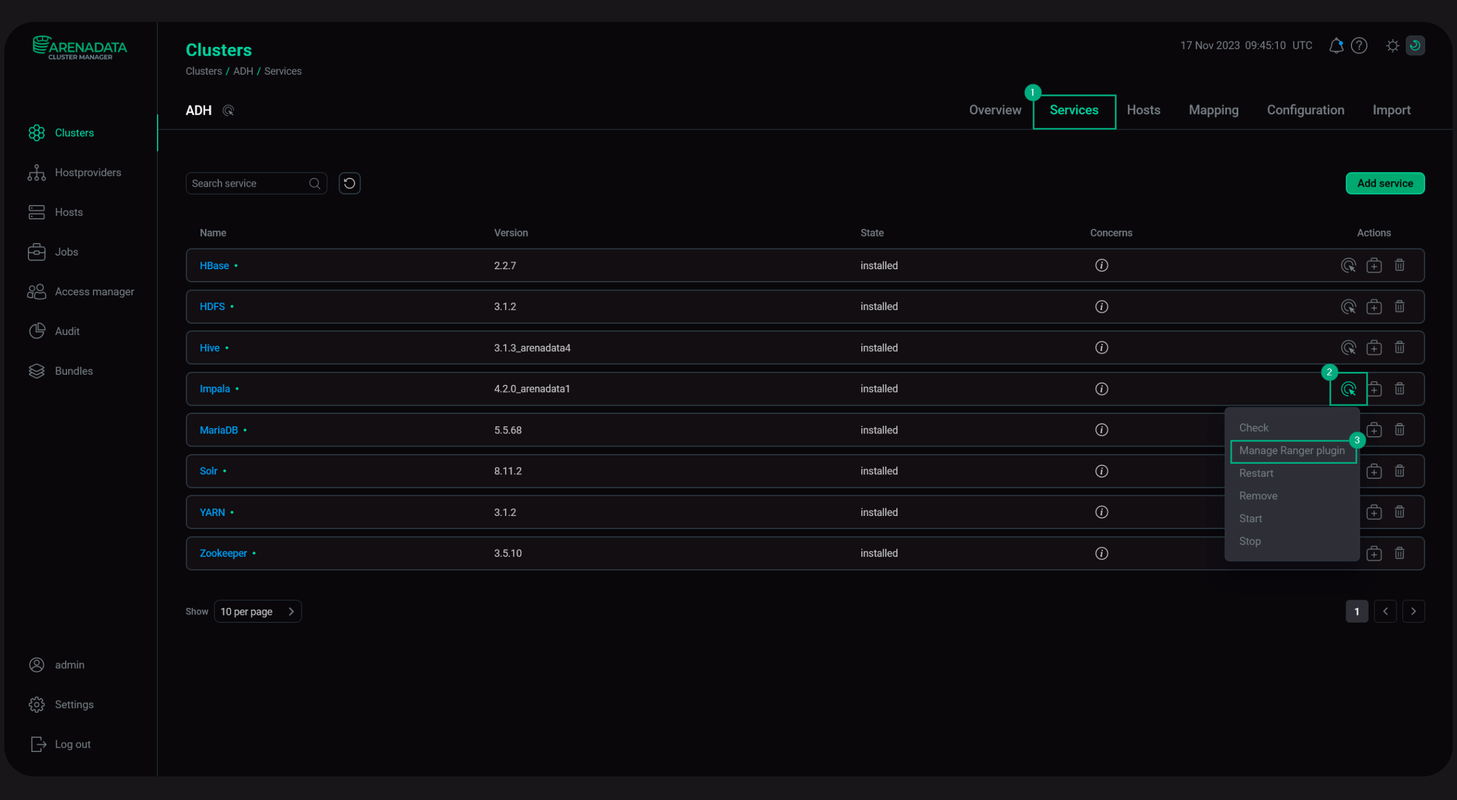
Add a new policy in Ranger

Enable Impala plugin

To enable the Ranger Impala plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.

2. Find Impala and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.

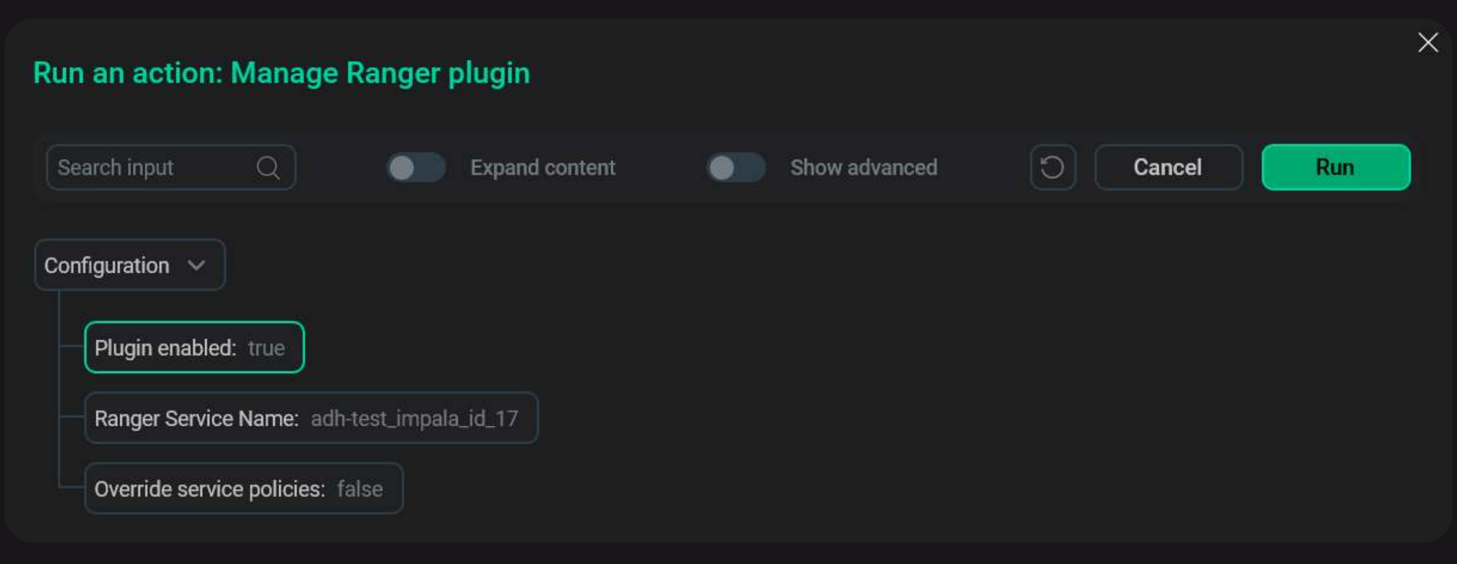


Name	Version	State
HBase	2.2.7	installed
HDFS	3.1.2	installed
Hive	3.1.3_arenadata4	installed
Impala	4.2.0_arenadata1	installed
MariaDB	5.5.68	installed
Solr	8.11.2	installed
YARN	3.1.2	installed
Zookeeper	3.5.10	installed

ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH / Services
17 Nov 2023 09:45:10 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add service
Concerns
Actions
Check
Manage Ranger plugin
Restart
Remove
Start
Stop
Show 10 per page
admin
Settings
Log out
1
2
3

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.



Run an action: Manage Ranger plugin

Plugin enabled: true

Ranger Service Name: adh-test_impala_id_17

Override service policies: false

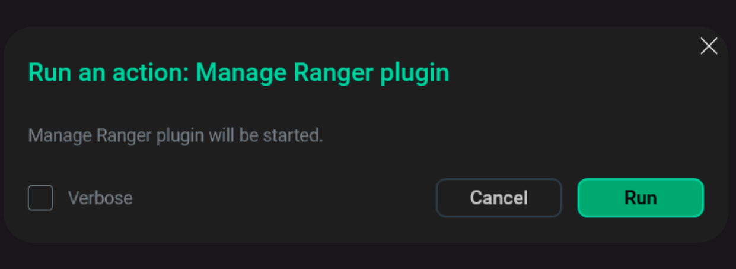
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin

Manage Ranger plugin will be started.

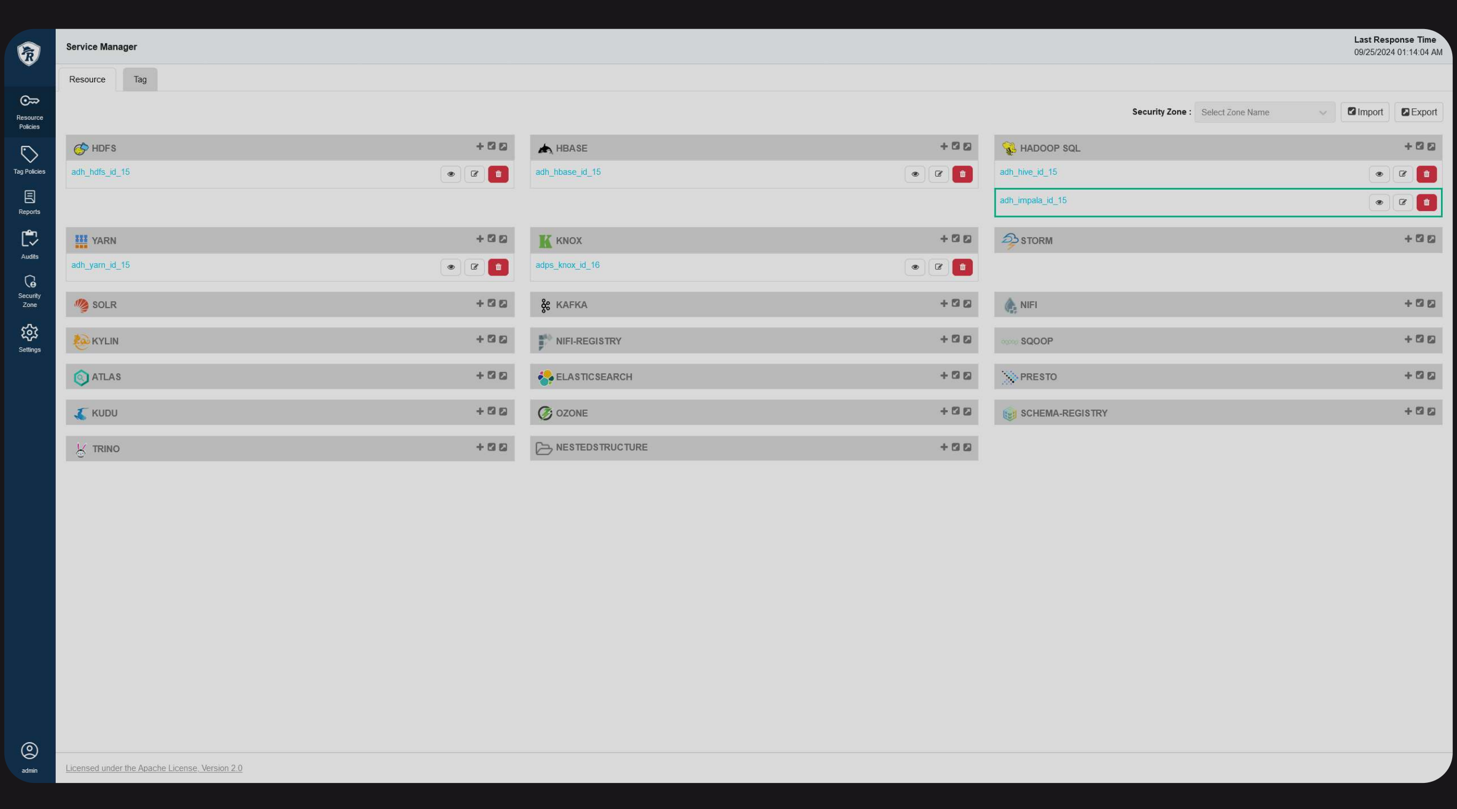
☐ Verbose
Cancel
Run

Action confirmation

Add a new policy in Ranger

To add a new policy to an existing Impala service, you should perform the following actions:

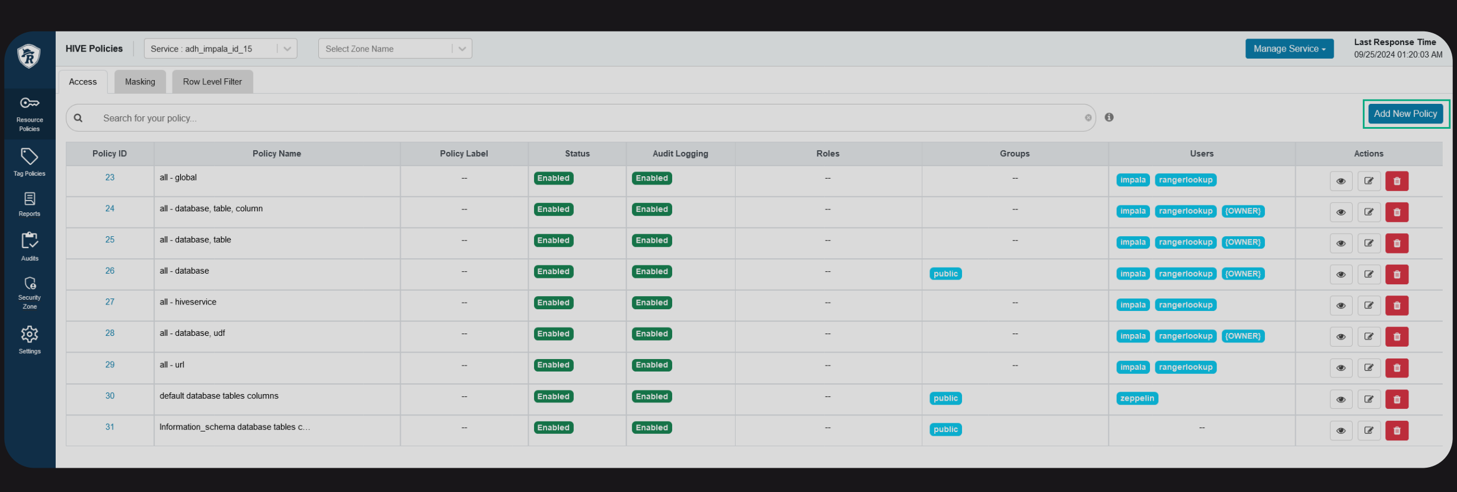
1. On the ***Service Manager*** page, click an existing Impala service in the Hadoop SQL pane.



Service Manager
Last Response Time
09/25/2024 01:14:04 AM
Resource
Tag
Security Zone :
Select Zone Name
Import
Export
HDFS
adh_hdfs_id_15
HBASE
adh_hbase_id_15
HADOOP SQL
adh_hive_id_15
adh_impala_id_15
YARN
adh_yarn_id_15
KNOX
adps_knox_id_16
STORM
SOLR
KAFKA
NIFI
KYLIN
NIFI-REGISTRY
SQOOP
ATLAS
ELASTICSEARCH
PRESTO
KUDU
OZONE
SCHEMA-REGISTRY
TRINO
NESTEDSTRUCTURE
Licensed under the Apache License, Version 2.0

Service Manager

2. On the Impala policy page, click ***Add New Policy***.

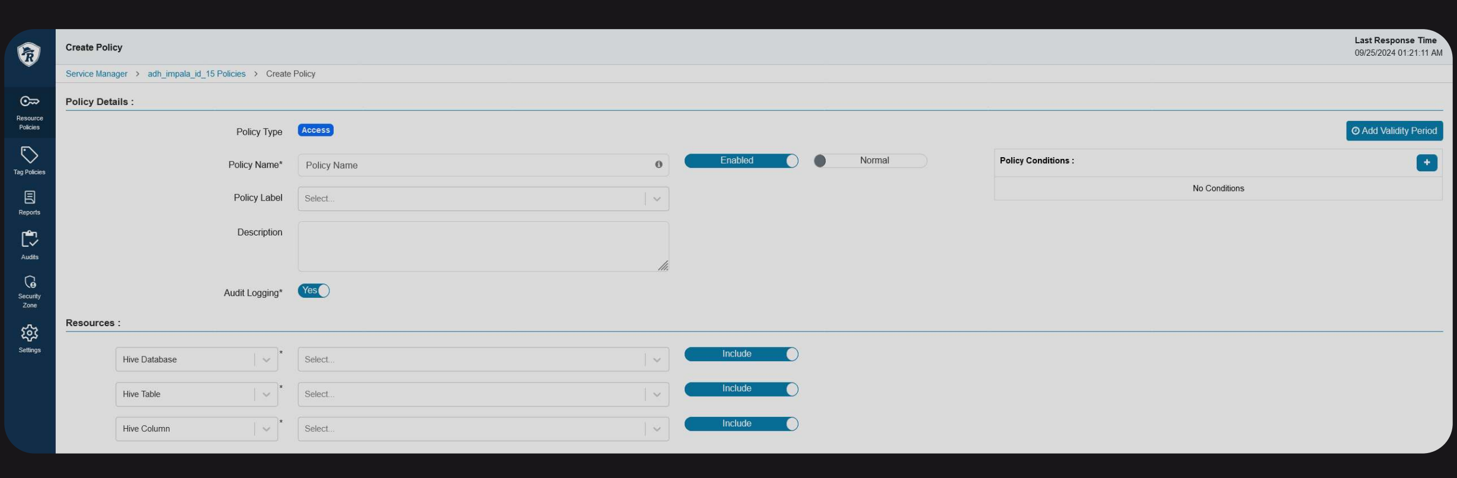


Policy ID	Policy Name	Policy Label	Status	Audit Logging	Roles	Groups	Users
23	all - global	–	Enabled	Enabled	–	–	impala rangerlookup
24	all - database, table, column	–	Enabled	Enabled	–	–	impala rangerlookup {OWNER}
25	all - database, table	–	Enabled	Enabled	–	–	impala rangerlookup {OWNER}
26	all - database	–	Enabled	Enabled	–	public	impala rangerlookup {OWNER}
27	all - hiveservice	–	Enabled	Enabled	–	–	impala rangerlookup
28	all - database, udf	–	Enabled	Enabled	–	–	impala rangerlookup {OWNER}
29	all - url	–	Enabled	Enabled	–	–	impala rangerlookup
30	default database tables columns	–	Enabled	Enabled	–	public	zeppelin
31	Information_schema database tables c...	–	Enabled	Enabled	–	public	–

HIVE Policies
Service : adh_impala_id_15
Select Zone Name
Manage Service
Last Response Time
09/25/2024 01:20:03 AM
Access
Masking
Row Level Filter
Search for your policy...
Add New Policy
Actions

Add new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.



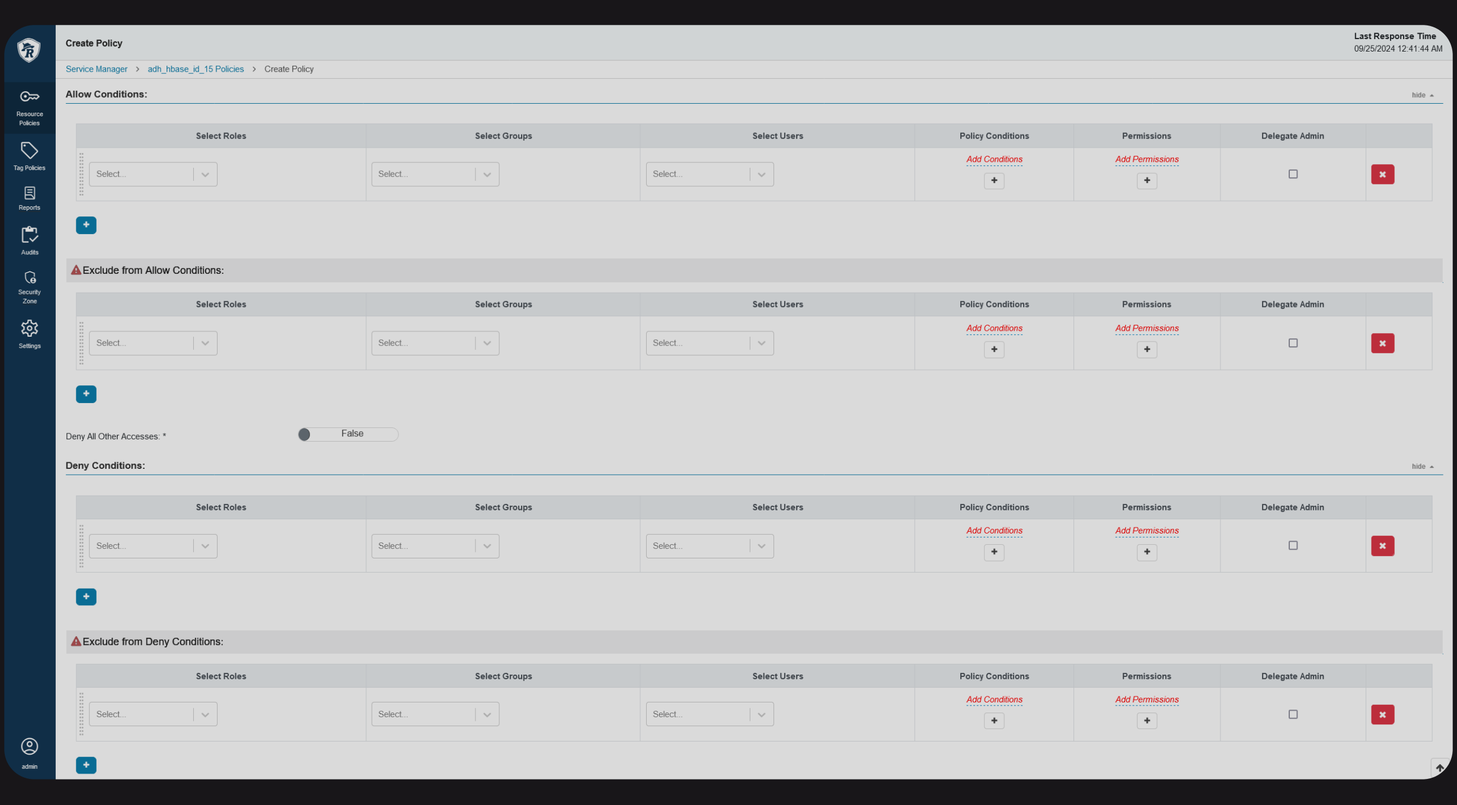
Create Policy
Last Response Time
09/25/2024 01:21:11 AM
Service Manager > adh_impala_id_15 Policies > Create Policy
Policy Details :
Policy Type
Access
Add Validity Period
Policy Name*
Enabled
Normal
Policy Conditions :
No Conditions
Policy Label
Select...
Description
Audit Logging*
Yes
Resources :
Hive Database
Hive Table
Hive Column
Include

Impala policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
database/url/hiveservice/global	If ***database*** is selected, specify the applicable database name. The autocomplete feature displays available databases based on the entered text. If ***url*** is selected, specify the cloud storage path (e.g., *s3a://dev-admin/demo/campaigns.txt*) where the end-user permission is needed to read/write the Impala data from/to a cloud storage path. The ***hiveservice*** option enables a user who has Service Admin permission in Ranger to run the stop query API: `kill query <queryID>`. Should be used only when Permissions=Service Admin. Select ***global*** to create global policies that use matching patterns and define the desired default access
table/udf/none	Specifies whether this policy is table-based or UDF-based
column/none	Specifies the applicable Impala column name
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click +. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Deny All Other Accesses: * False

Create Policy
Last Response Time
09/25/2024 12:41:44 AM
Service Manager > adh_hbase_id_15 Policies > Create Policy
Allow Conditions:
Exclude from Allow Conditions:
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Add Conditions
Add Permissions
Select...

Impala allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger Ozone plugin

Sergei Tikhomirov

Contents

To Table of Contents

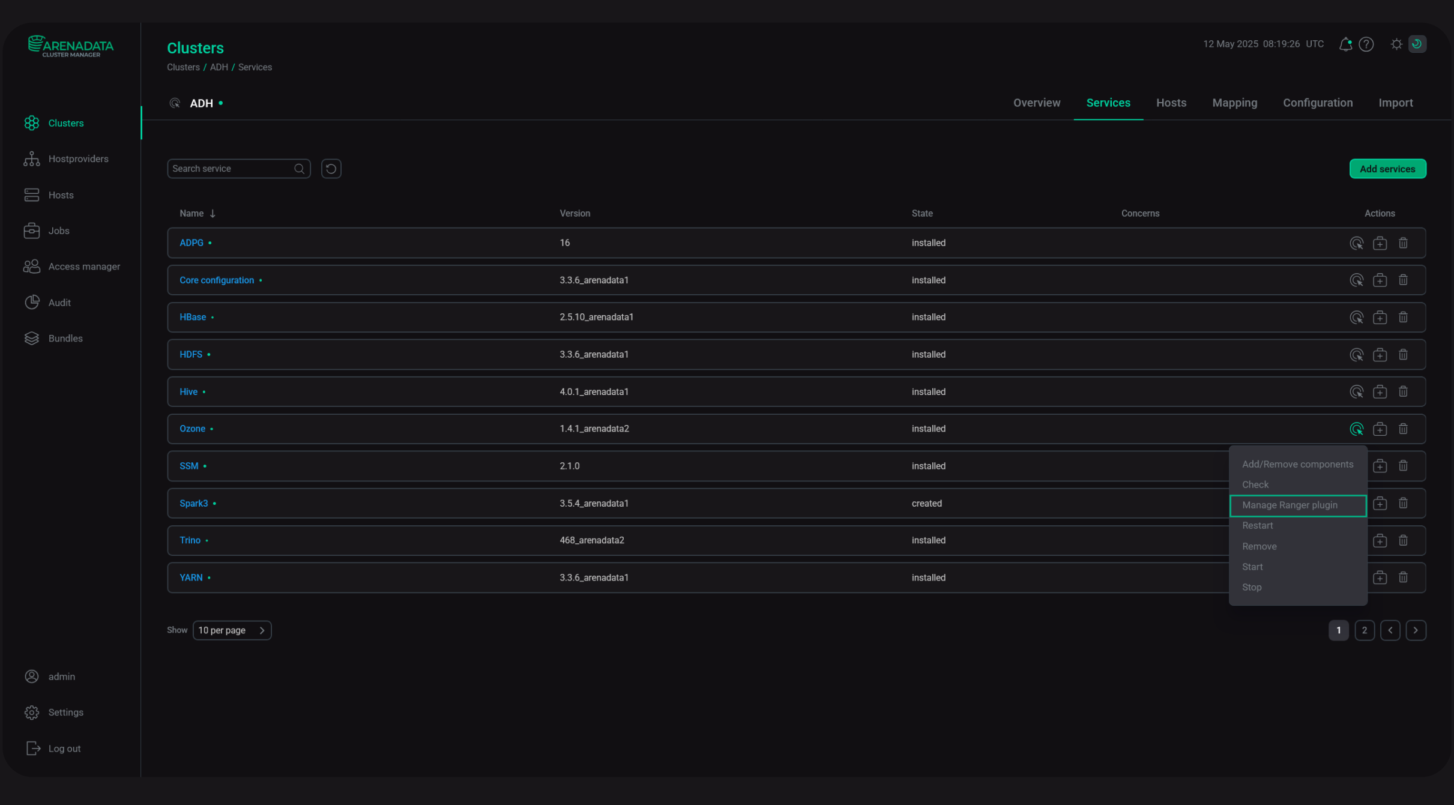
Enable Ozone plugin

Add a new policy in Ranger.

Enable Ozone plugin

To enable the Ranger Ozone plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.
2. Find Ozone and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.

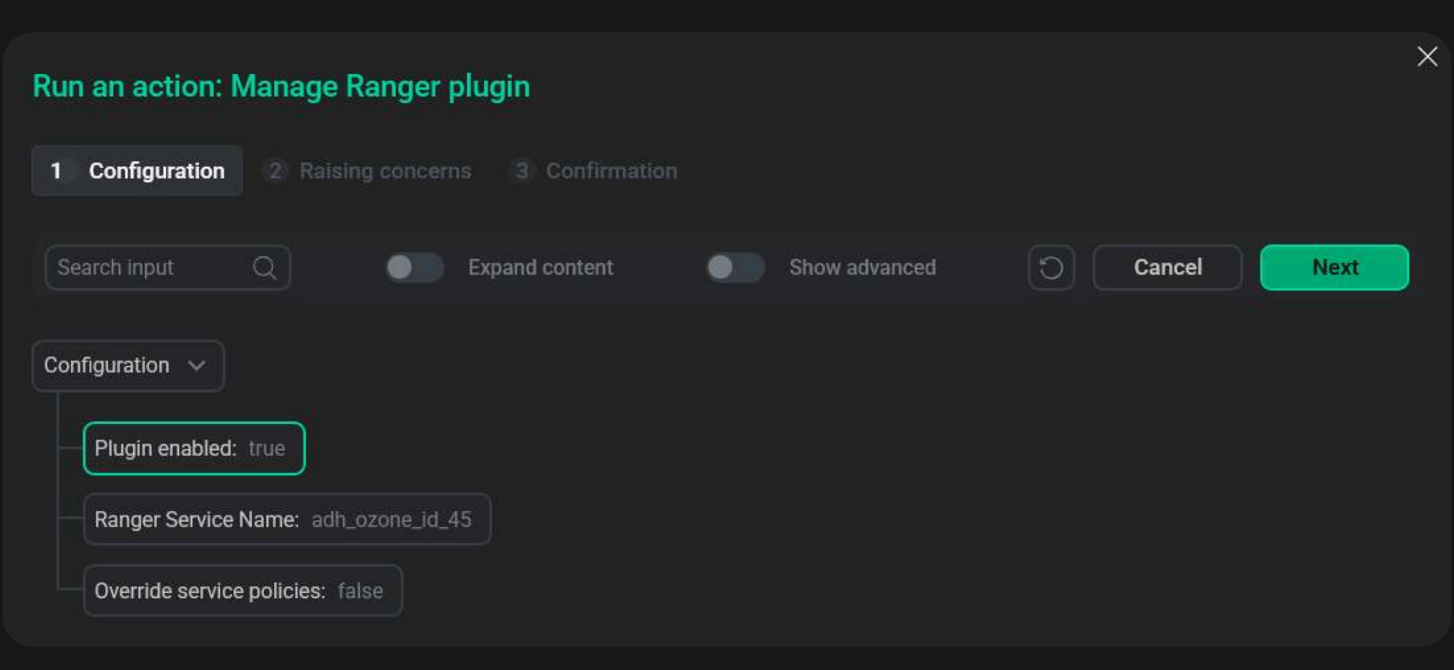


Name	Version	State	Concerns	Actions
ADPG	16	installed		
Core configuration	3.3.6_arenadata1	installed		
HBase	2.5.10_arenadata1	installed		
HDFS	3.3.6_arenadata1	installed		
Hive	4.0.1_arenadata1	installed		
Ozone	1.4.1_arenadata2	installed		
SSM	2.1.0	installed		
Spark3	3.5.4_arenadata1	created		
Trino	468_arenadata2	installed		
YARN	3.3.6_arenadata1	installed		

Clusters
Clusters / ADH / Services
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Add/Remove components
Check
Manage Ranger plugin
Restart
Remove
Start
Stop
Show 10 per page

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.



Run an action: Manage Ranger plugin

Plugin enabled: true

Ranger Service Name: adh_ozone_id_45

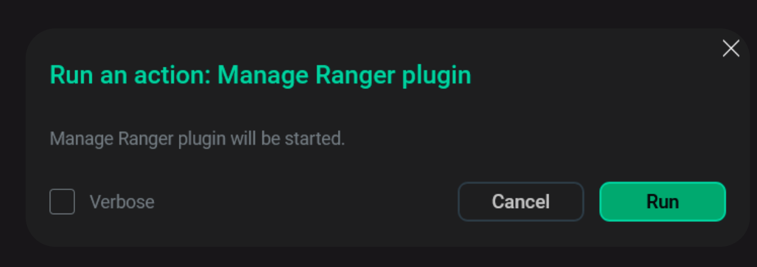
Override service policies: false

1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Plugin state

> **NOTE**
>
> On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin

Manage Ranger plugin will be started.

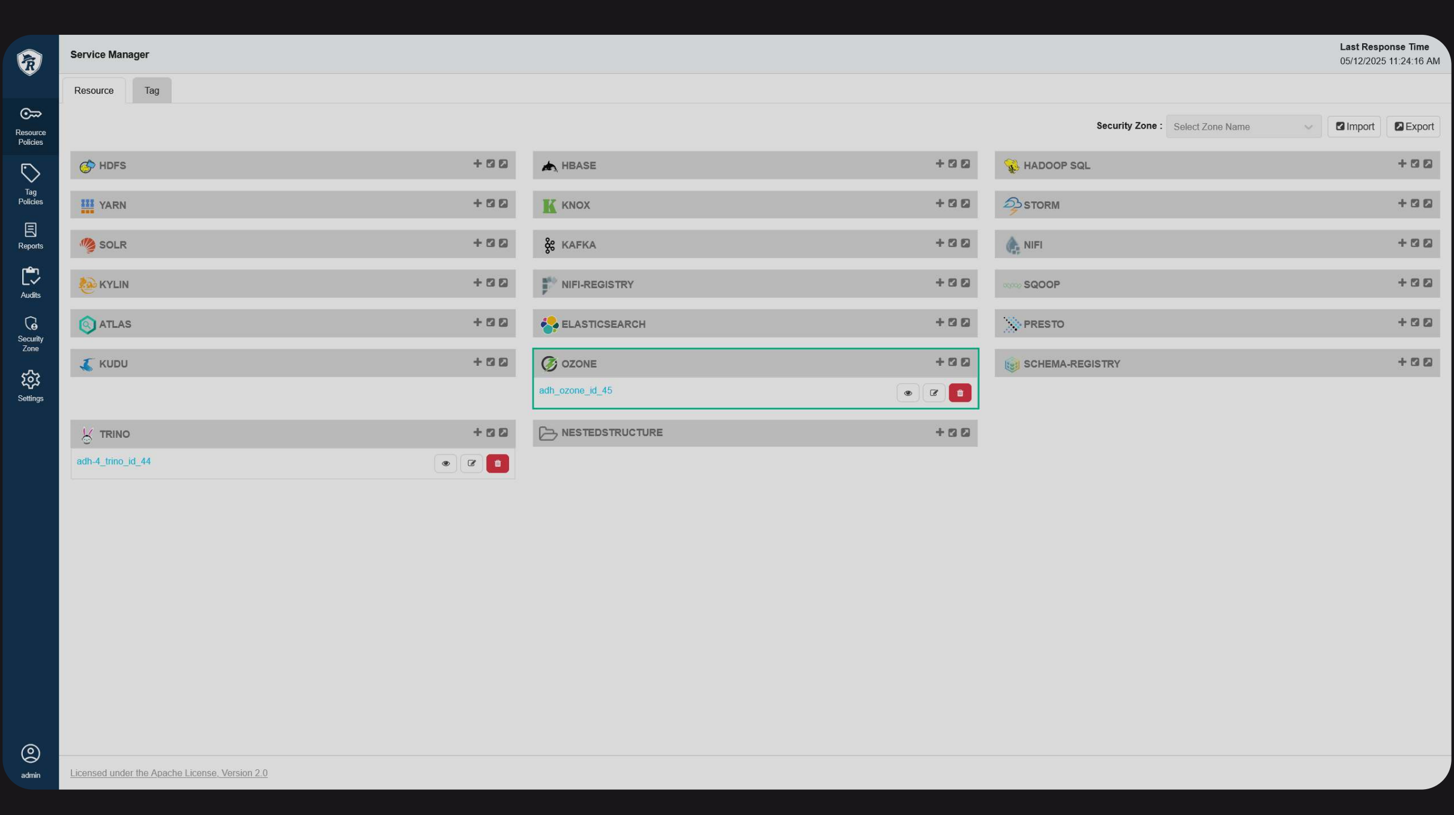
☐ Verbose
Cancel
Run

Action confirmation

Add a new policy in Ranger.

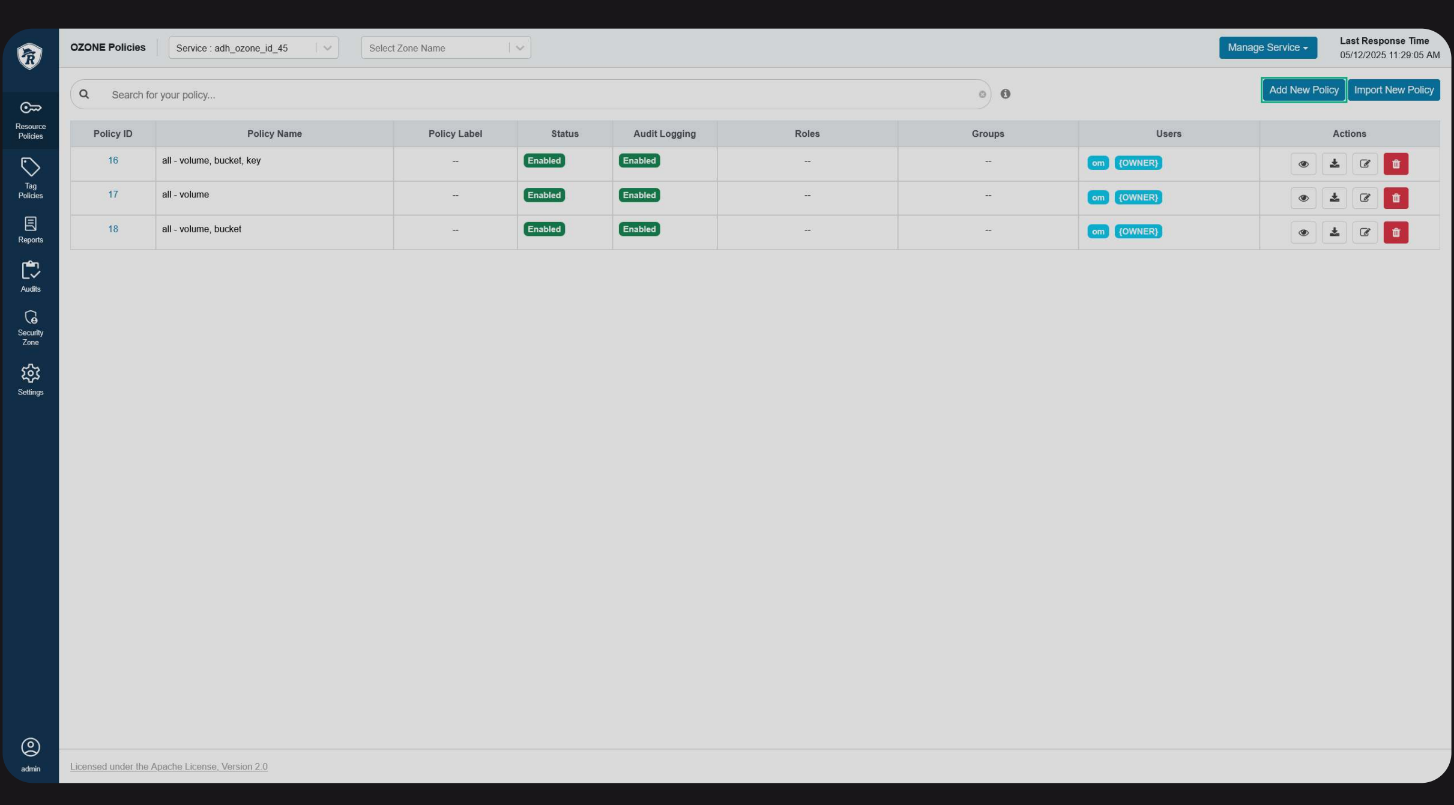
To add a new policy to an existing Ozone service, you should perform the following actions:

1. On the ***Service Manager*** page, click an existing Ozone service in the Ozone pane.



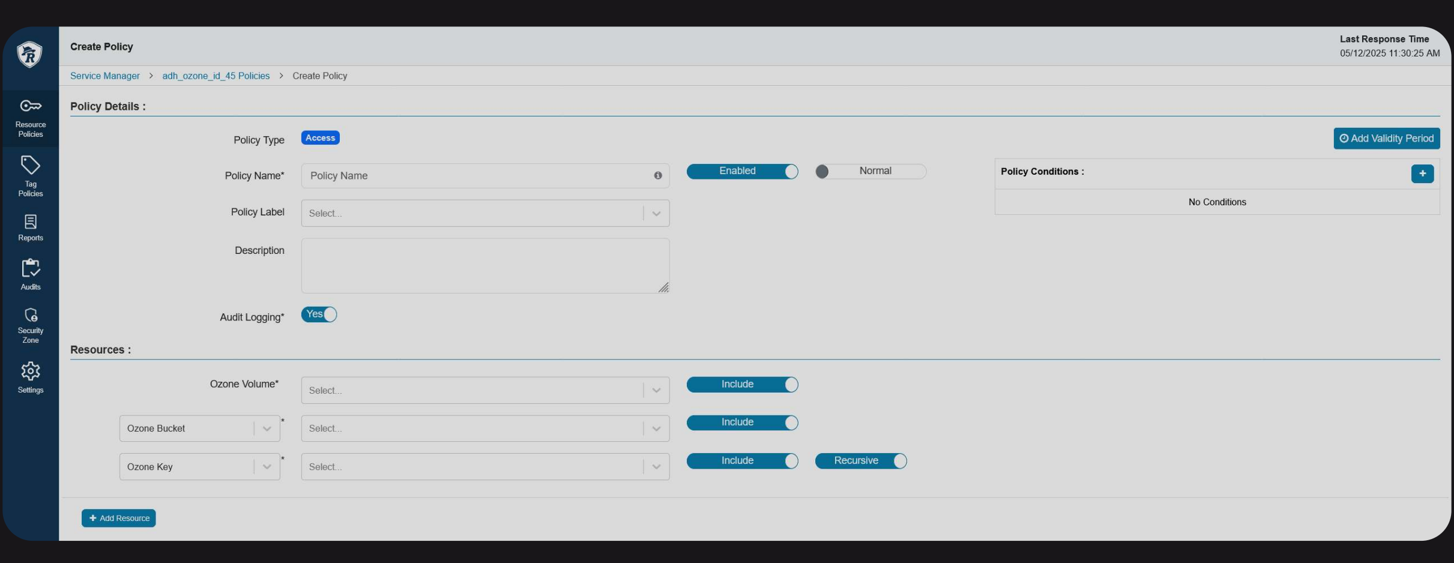
Service Manager

2. On the Ozone policy page, click ***Add New Policy***.



Add a new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.



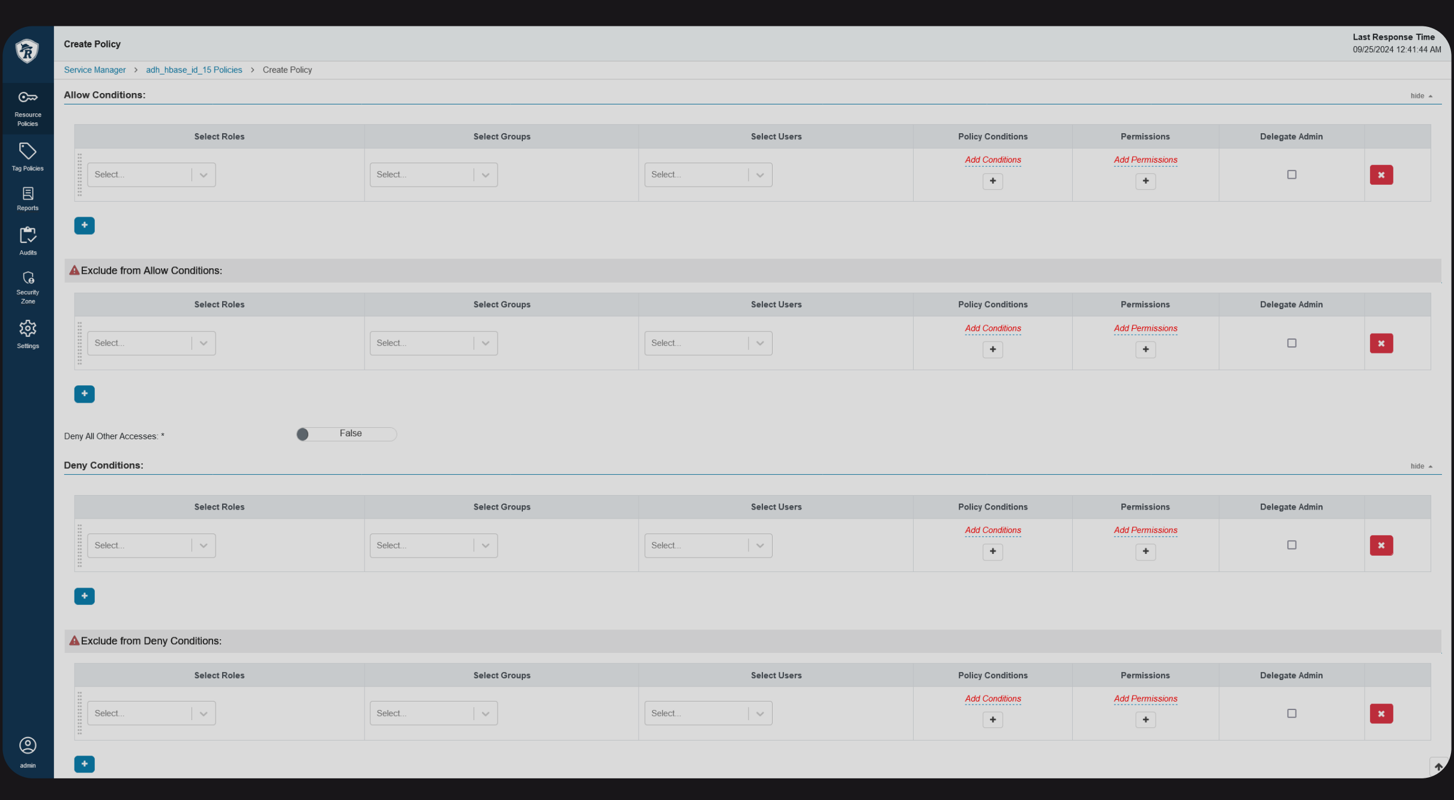
Create Policy
Service Manager > adh_ozone_id_45 Policies > Create Policy
Policy Details :
Policy Type
Access
Add Validity Period
Policy Name*
Enabled
Normal
Policy Conditions :
No Conditions
Policy Label
Description
Audit Logging*
Yes
Resources :
Ozone Volume*
Ozone Bucket
Ozone Key
Include
Recursive
Add Resource

Ozone policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy
Ozone Volume	Name of an Ozone volume
Ozone Bucket	Name of an Ozone bucket
Ozone Key	Name of an Ozone key

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click **+**. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Deny All Other Accesses: * False

Create Policy
Service Manager > adh_hbase_id_15 Policies > Create Policy
Allow Conditions:
Exclude from Allow Conditions:
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Add Conditions
Add Permissions

Ozone allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger Solr plugin

Sergei Tikhomirov

Contents

To Table of Contents

Enable Solr plugin

Add a new policy in Ranger

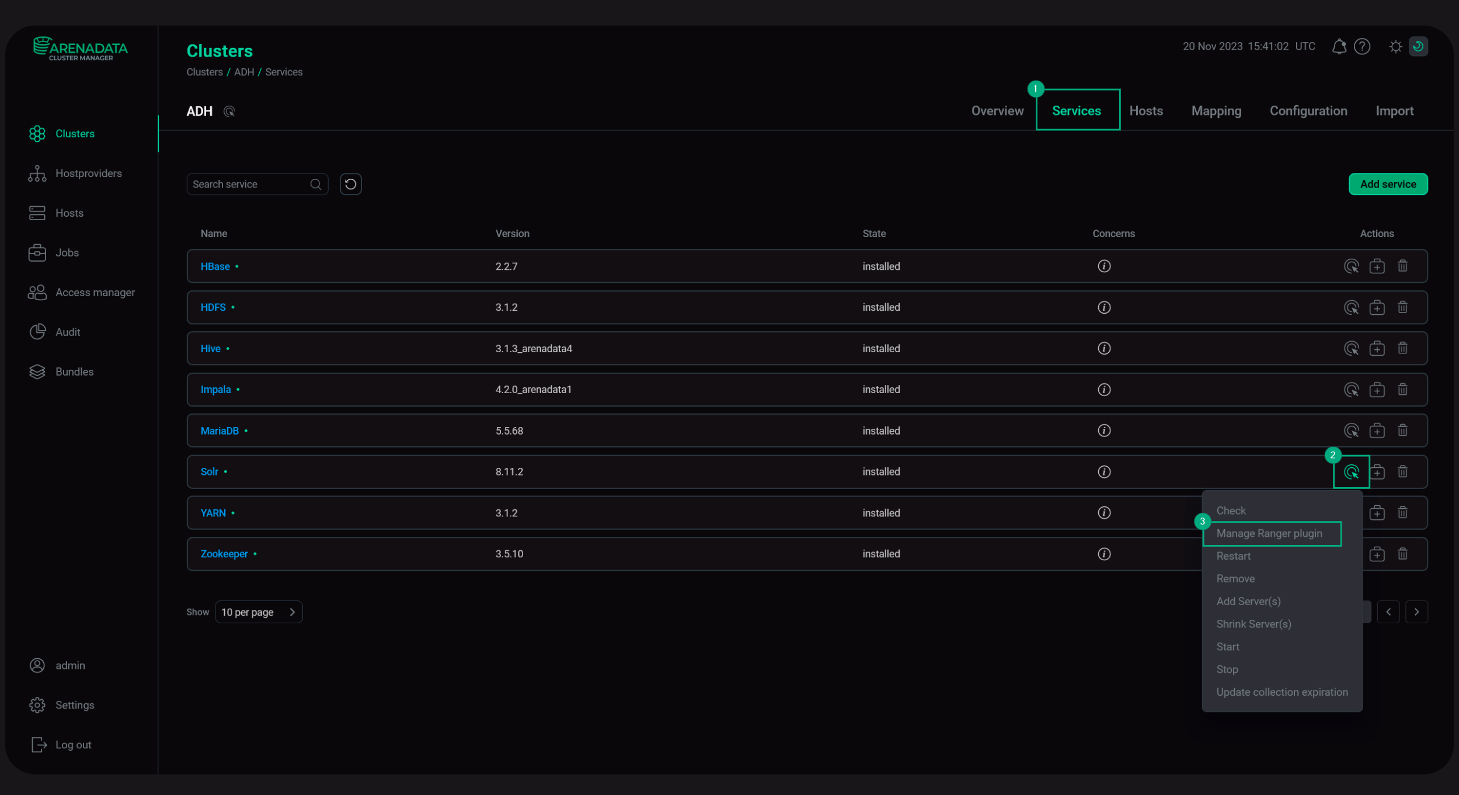
Enable Solr plugin

IMPORTANT

Solr doesn't have a basic authentication implementation, so its Ranger plugin is available only for kerberized ADH clusters.

To enable the Ranger Solr plugin, follow the steps below:

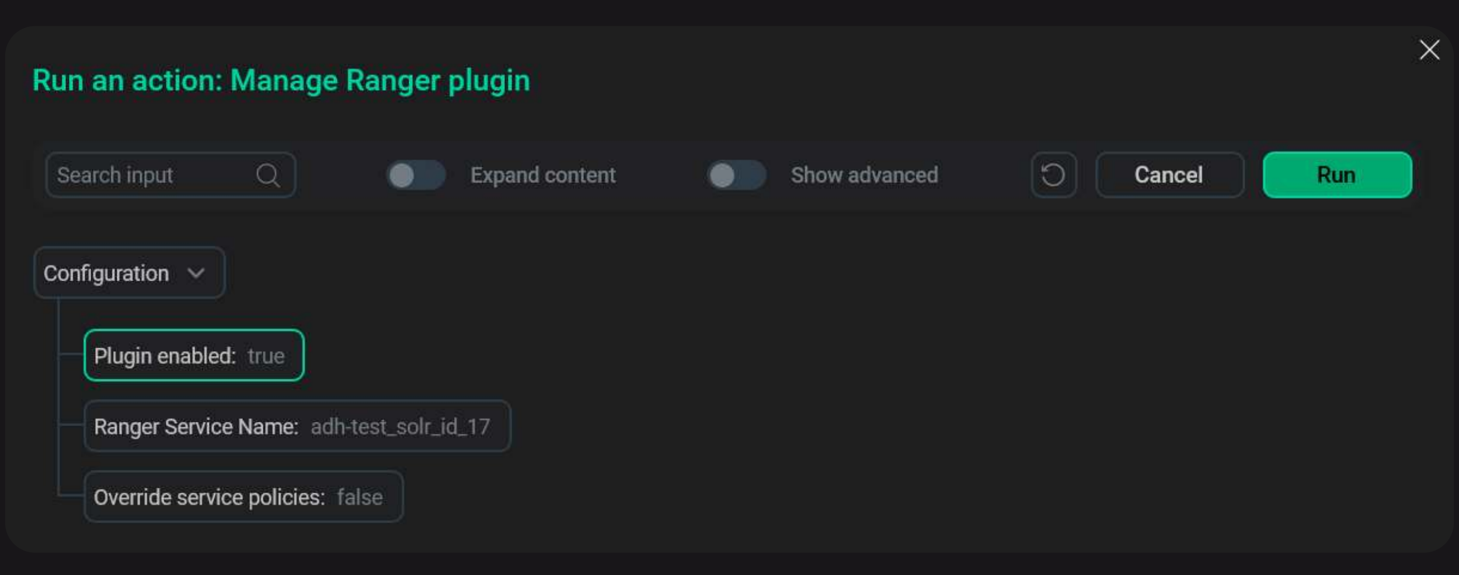
1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.
2. Find Solr and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH / Services
20 Nov 2023 15:41:02 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add service
1
2
3
Check
Manage Ranger plugin
Restart
Remove
Add Server(s)
Shrink Server(s)
Start
Stop
Update collection expiration
Show 10 per page

Name	Version	State	Concerns	Actions
HBase	2.2.7	installed		
HDFS	3.1.2	installed		
Hive	3.1.3_arenadata4	installed		
Impala	4.2.0_arenadata1	installed		
MariaDB	5.5.68	installed		
Solr	8.11.2	installed		
YARN	3.1.2	installed		
Zookeeper	3.5.10	installed		

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.


Run an action: Manage Ranger plugin
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Plugin enabled: true

Ranger Service Name: adh-test_solr_id_17

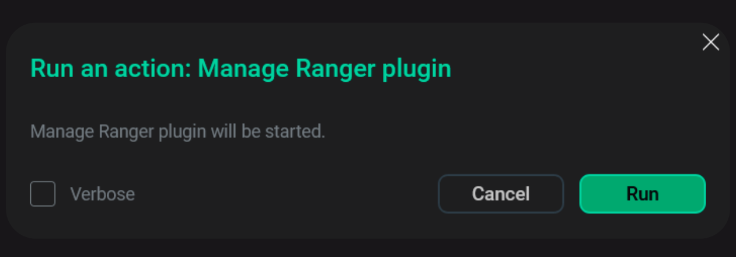
Override service policies: false

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.


Run an action: Manage Ranger plugin
☐ Verbose
Cancel
Run

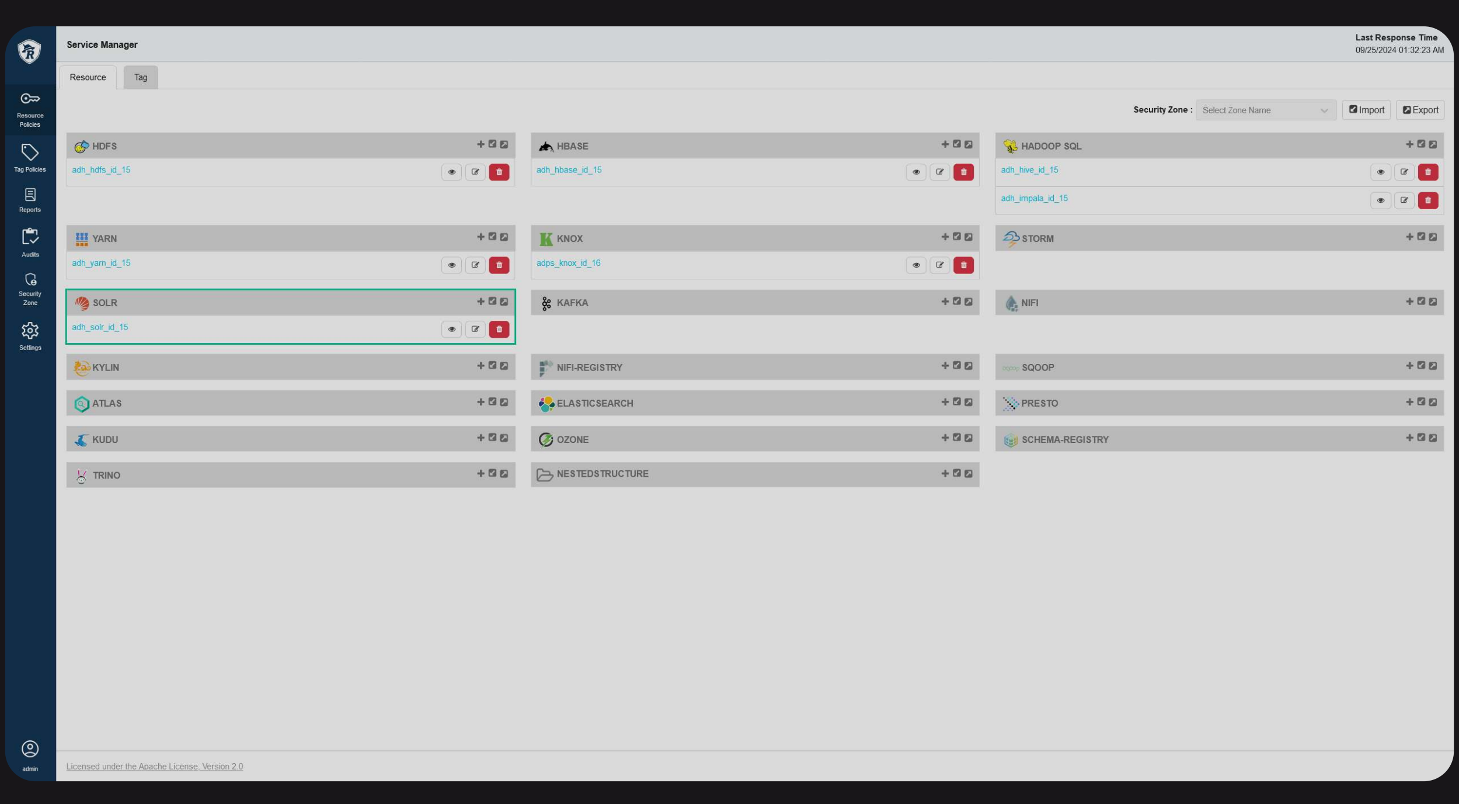
Manage Ranger plugin will be started.

Action confirmation

Add a new policy in Ranger

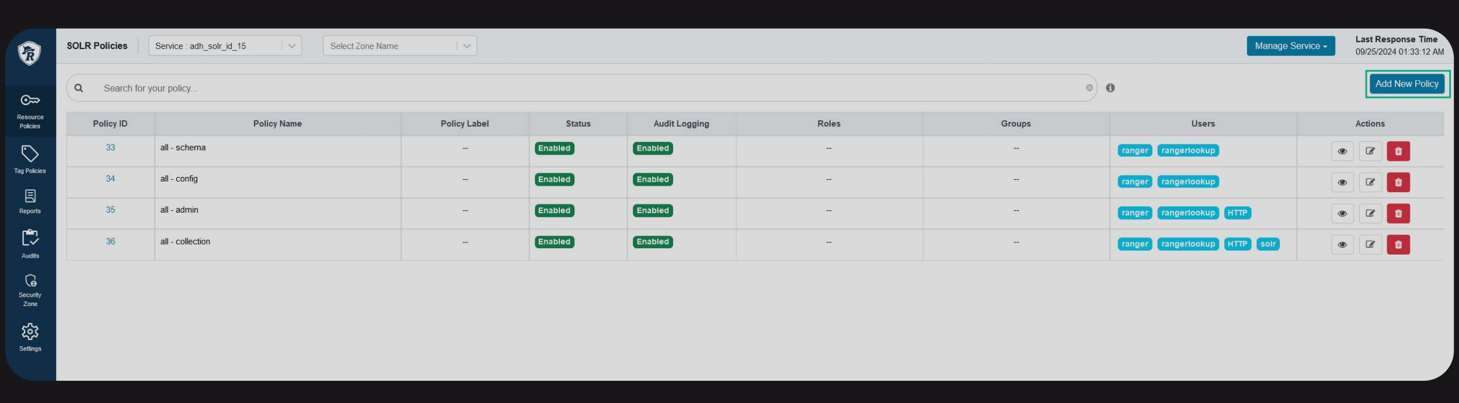
To add a new policy to an existing Solr service, you should perform the following actions:

1. On the ***Service Manager*** page, click an existing service on the Solr tab.


Service Manager
Last Response Time
09/25/2024 01:32:23 AM
Resource
Tag
Security Zone :
Select Zone Name
Import
Export
HDFS
adh_hdfs_id_15
HBASE
adh_hbase_id_15
HADOOP SQL
adh_hive_id_15
adh_impala_id_15
YARN
adh_yarn_id_15
KNOX
adps_knox_id_15
STORM
SOLR
adh_solr_id_15
KAFKA
NIFI
KYLIN
NIFI-REGISTRY
SQOOP
ATLAS
ELASTICSEARCH
PRESTO
KUDU
OZONE
SCHEMA-REGISTRY
TRINO
NESTEDSTRUCTURE
Licensed under the Apache License, Version 2.0

Service Manager

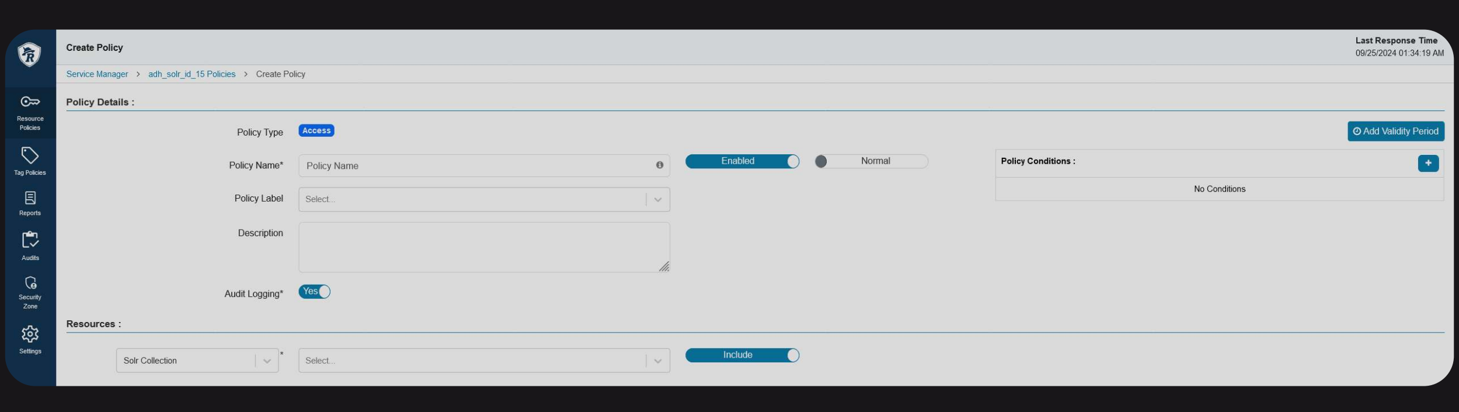
2. On the Solr policy page, click ***Add New Policy***.


SOLR Policies
Service : adh_solr_id_15
Select Zone Name
Manage Service
Last Response Time
09/25/2024 01:33:12 AM
Search for your policy...
Add New Policy

Policy ID	Policy Name	Policy Label	Status	Audit Logging	Roles	Groups	Users	Actions
33	all - schema	–	Enabled	Enabled	–	–	ranger rangerlookup	
34	all - config	–	Enabled	Enabled	–	–	ranger rangerlookup	
35	all - admin	–	Enabled	Enabled	–	–	ranger rangerlookup HTTP	
36	all - collection	–	Enabled	Enabled	–	–	ranger rangerlookup HTTP solr	

Add new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.

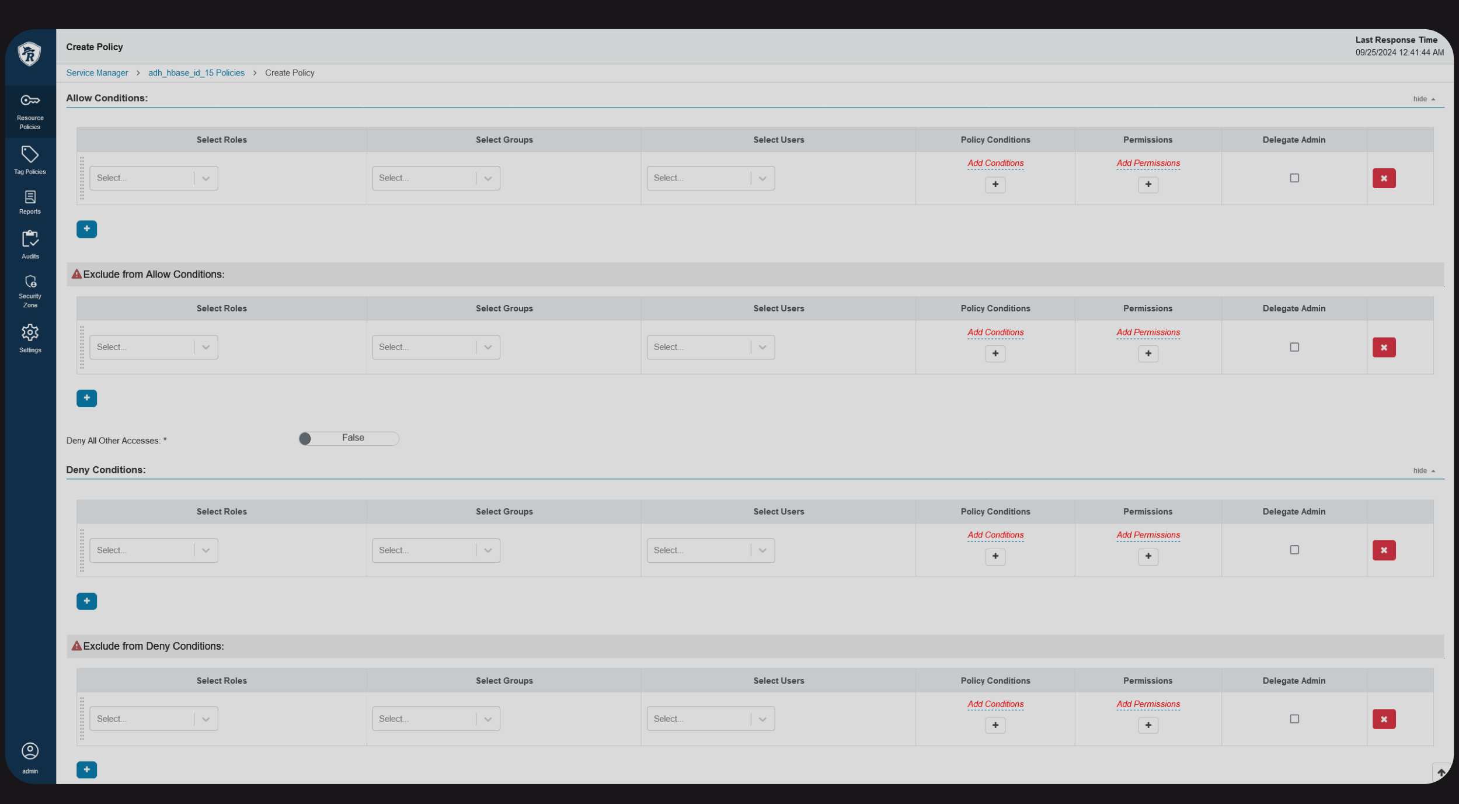

Create Policy
Last Response Time
09/25/2024 01:34:19 AM
Service Manager › adh_solr_id_15 Policies › Create Policy
Policy Details :
Policy Type
Access
Add Validity Period
Policy Name*
Policy Name
Enabled
Normal
Policy Conditions :
No Conditions
Policy Label
Select...
Description
Audit Logging*
Yes
Resources :
Solr Collection
Select...
Include

Solr policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
Solr Collection	URL to a Solr collection (e.g, *http://<solr_host>:6083/solr/ranger_audits*)
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click **+**. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.


Create Policy
Last Response Time
09/25/2024 12:41:44 AM
Service Manager › adh_hbase_id_15 Policies › Create Policy
Allow Conditions:
Exclude from Allow Conditions:
Deny All Other Accesses: *
False
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Select...
Add Conditions
Add Permissions

Solr allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger Spark3 plugin

Sergei Tikhomirov

Contents

To Table of Contents

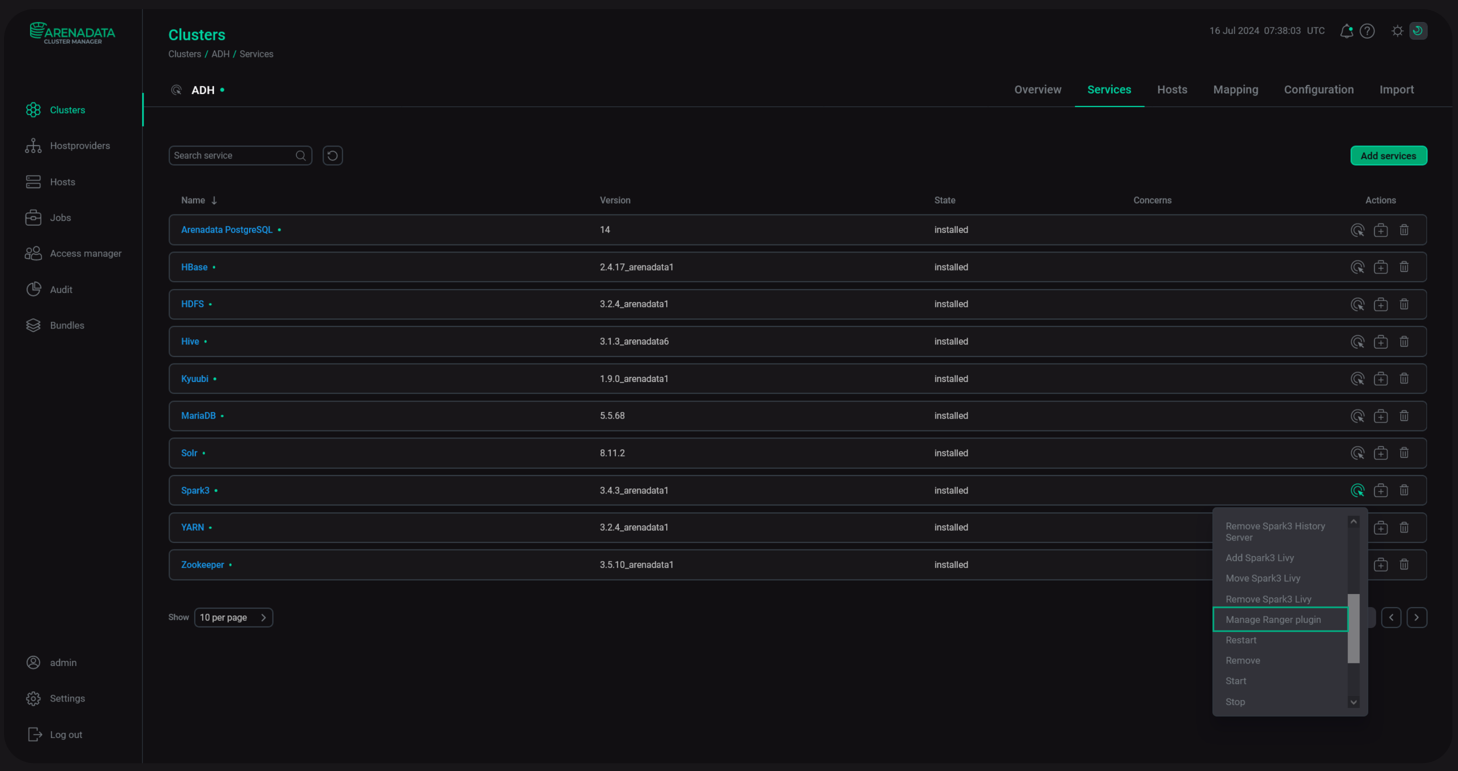
Enable Spark3 plugin

Enable Spark3 plugin

To enable the Ranger Spark3 plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.

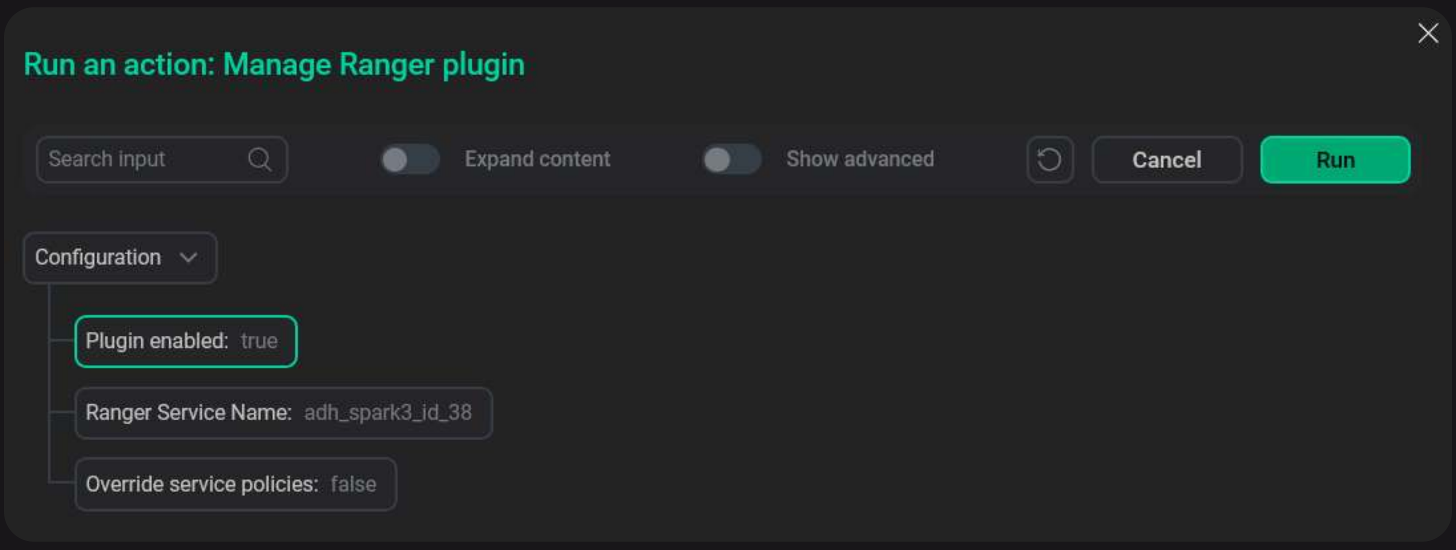
2. Find Spark3 and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / ADH / Services
16 Jul 2024 07:38:03 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Remove Spark3 History Server
Add Spark3 Livy
Move Spark3 Livy
Remove Spark3 Livy
Manage Ranger plugin
Restart
Remove
Start
Stop
Show
10 per page

Name	Version	State	Concerns	Actions
Arenadata PostgreSQL	14	installed		
HBase	2.4.17_arenadata1	installed		
HDFS	3.2.4_arenadata1	installed		
Hive	3.1.3_arenadata6	installed		
Kyuubi	1.9.0_arenadata1	installed		
MariaDB	5.5.68	installed		
Solr	8.11.2	installed		
Spark3	3.4.3_arenadata1	installed		
YARN	3.2.4_arenadata1	installed		
Zookeeper	3.5.10_arenadata1	installed		

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.


Run an action: Manage Ranger plugin
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Plugin enabled: true

Ranger Service Name: adh_spark3_id_38

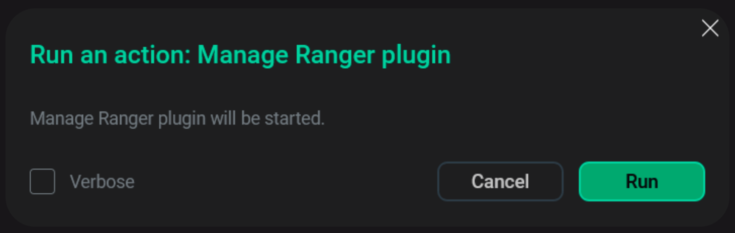
Override service policies: false

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.


Run an action: Manage Ranger plugin
☐ Verbose
Cancel
Run

Manage Ranger plugin will be started.

Action confirmation

Ranger Trino plugin

Sergei Tikhomirov

Contents

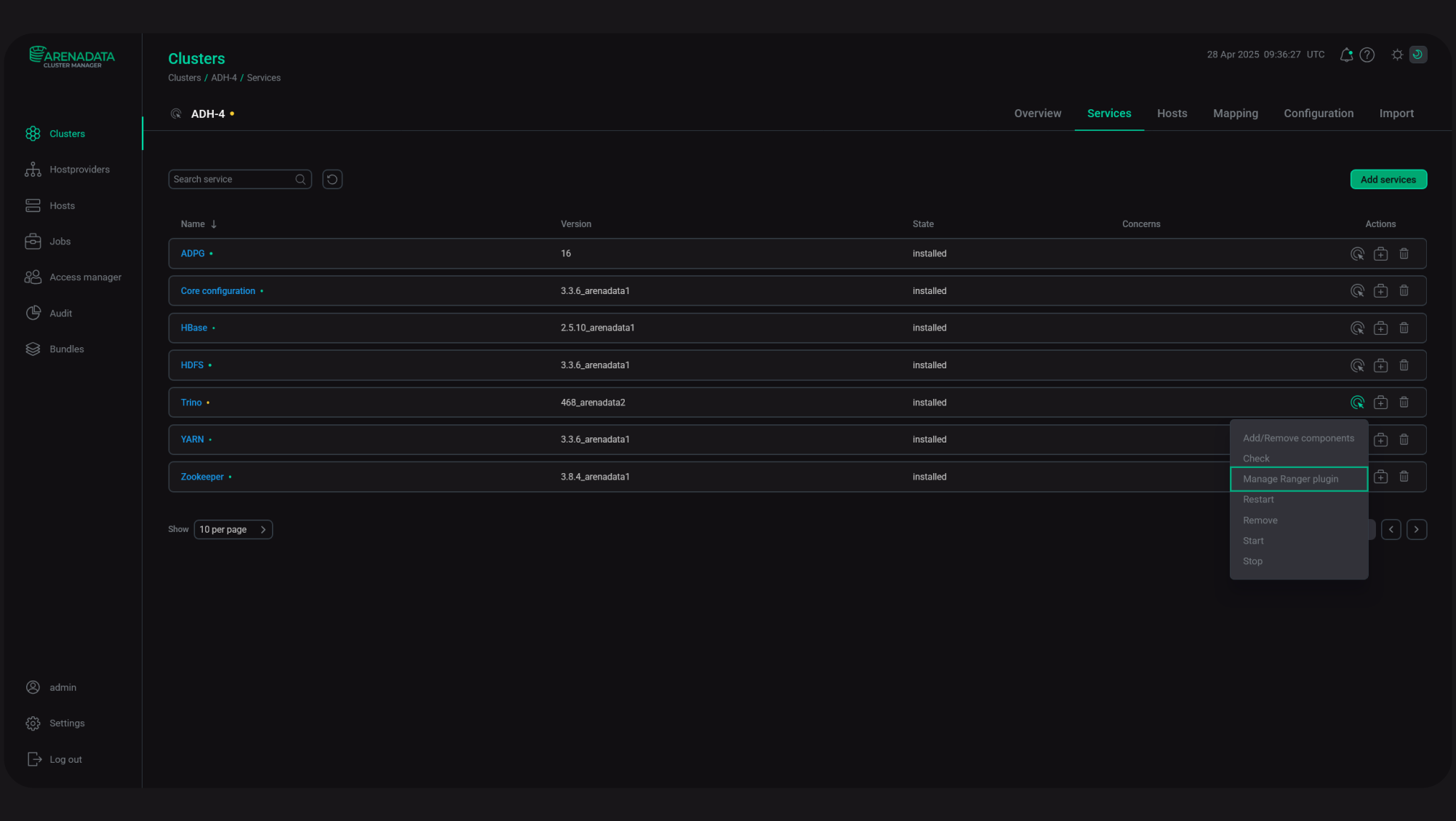
To Table of Contents

Enable Trino plugin
Trino to Hive resource and access type mapping
Add a new policy in Ranger

Enable Trino plugin

To enable the Ranger Trino plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.
2. Find Trino and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.



ARENADATA
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / ADH-4 / Services
ADH-4
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
Name
Version
State
Concerns
Actions
Add/Remove components
Check
Manage Ranger plugin
Restart
Remove
Start
Stop
Show 10 per page

Name	Version	State
ADPG	16	installed
Core configuration	3.3.6_arenadata1	installed
HBase	2.5.10_arenadata1	installed
HDFS	3.3.6_arenadata1	installed
Trino	468_arenadata2	installed
YARN	3.3.6_arenadata1	installed
Zookeeper	3.8.4_arenadata1	installed

Cluster services

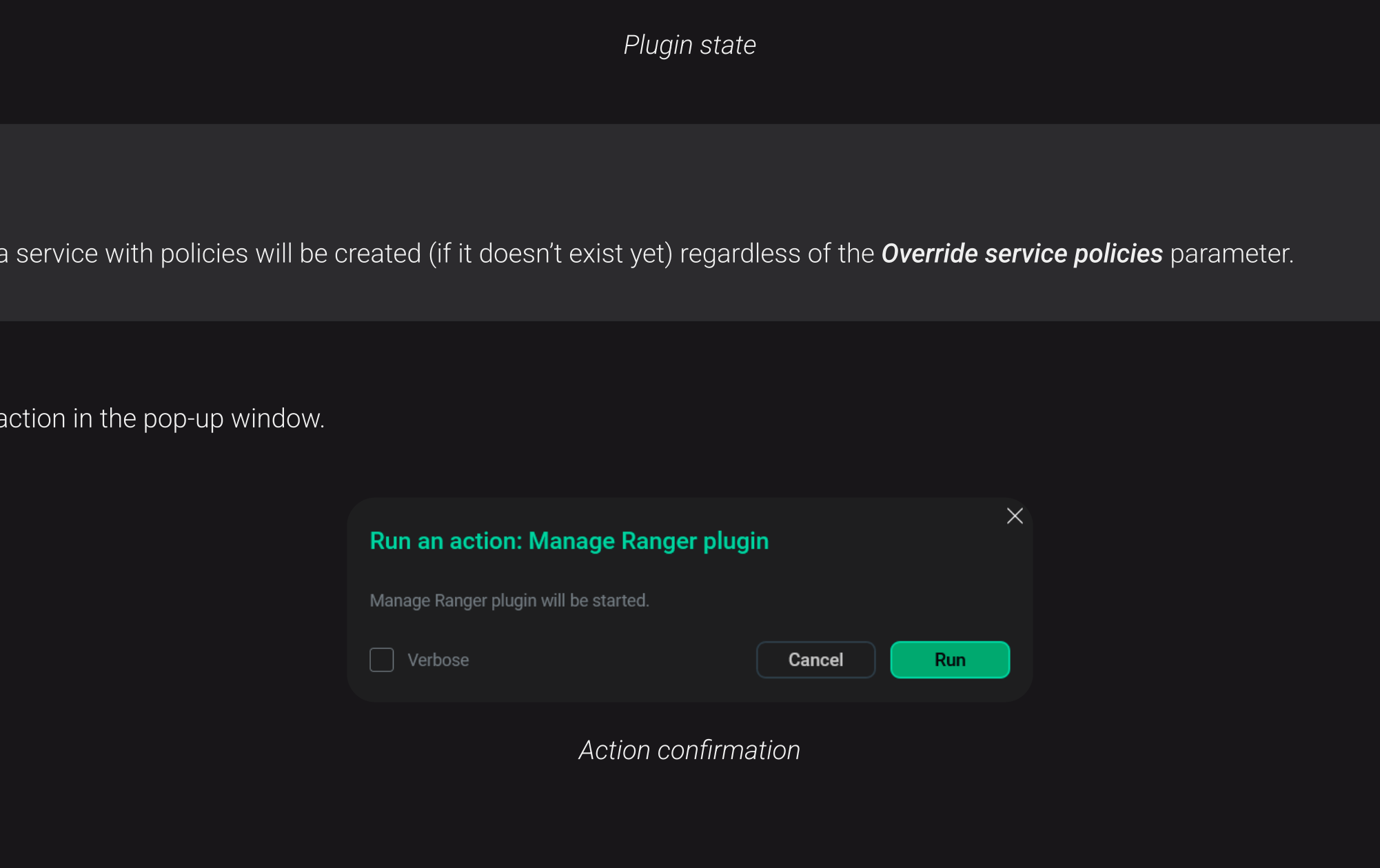
3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.

IMPORTANT

To make the ***Hive SQL support*** parameter available, you must turn on the Hive Ranger plugin first. This parameter allows using a shared Hadoop SQL policy with such services as Hive/Impala/SparkSQL.

The policies are applied in the following order:

1. Trino-specific policies are applied first.
2. If a policy is defined in both places, the Trino policy is applied.
3. If a policy is only in one place, it is applied.



Run an action: Manage Ranger plugin
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

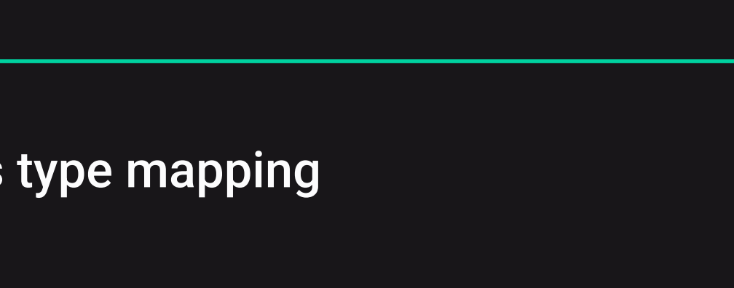
Plugin enabled: false
Ranger Service Name: adh_trino_id_2
Override service policies: false
Rewrite current access-control type: true
Hive SQL support
ranger.plugin.hive.service.name: adh_hive_id_2

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin
Verbose
Cancel
Run

Manage Ranger plugin will be started.

Action confirmation

Required policies

- Users must have permission to execute queries in Trino. Without a policy in Apache Ranger to grant this permission, users are not able to execute any query. To allow this, create a policy in Apache Ranger for a `queryId` resource with a value `*` and with the `execute` permission for the user `{USER}`.
- Users must have permission to impersonate themselves in Trino. Without a policy in Apache Ranger to grant this permission, users are not able to execute any query. To allow this, create a policy in Apache Ranger for a `trinouser` resource with value `{USER}` and with the `impersonate` permission for user `{USER}`.

Trino to Hive resource and access type mapping

The access objects mapping is as follows:

- `{schema}` → `{database}`
- `{schema}.{table}` → `{database}.{table}`
- `{catalog}.{schema}.{table}.{column}` → `{database}.{table}.{column}`

Database access types

Trino access type	Hive access type
all	all
_any	use
create	create
drop	drop
alter	alter
select	use
show	use
use	use

Table access types

Trino access type	Hive access type
all	all
_any	use
create	create
drop	drop
alter	alter
insert	update
delete	update
select	select
show	use
use	use

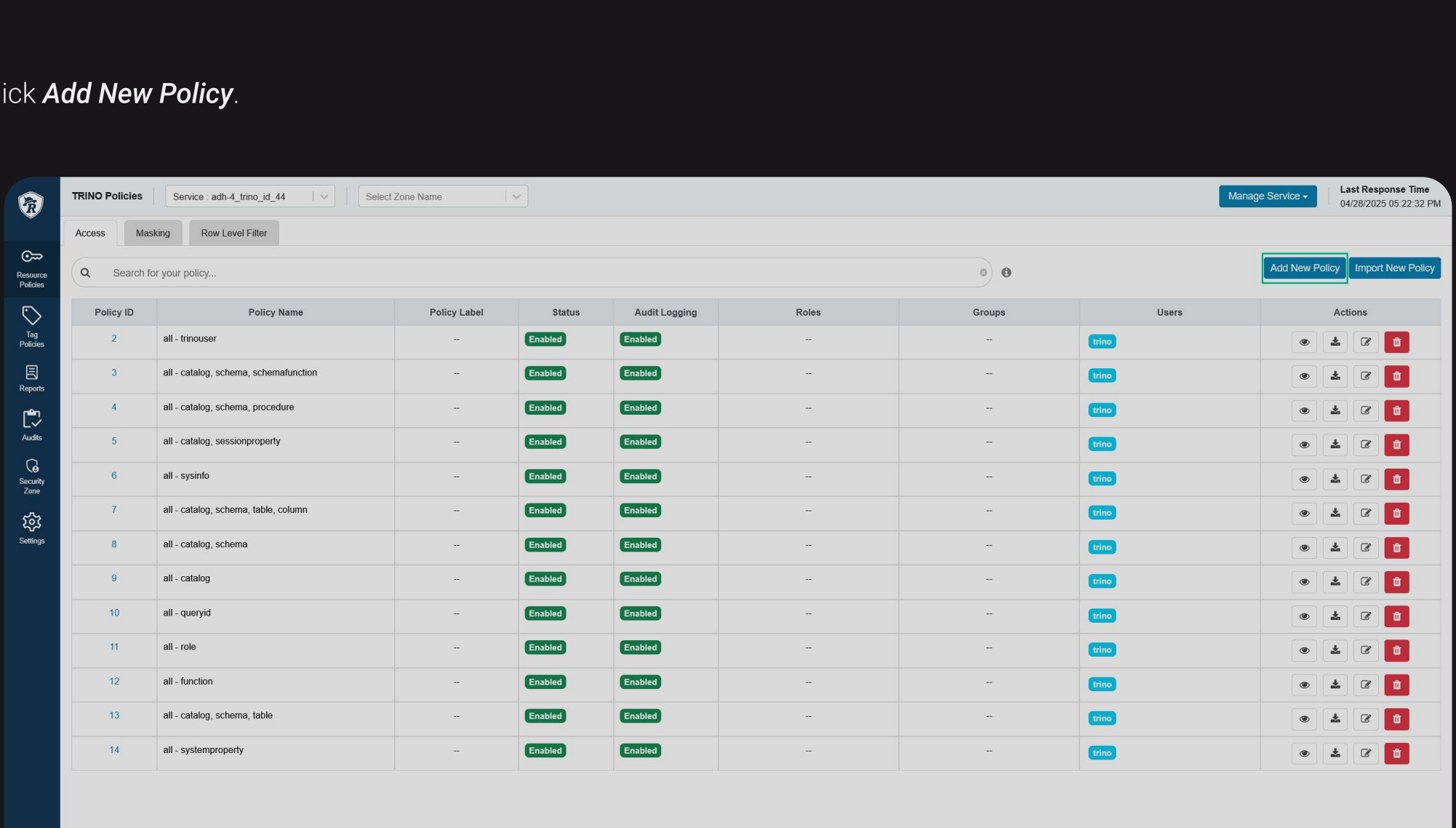
Column access types

Trino access type	Hive access type
all	all
_any	use
select	select

Add a new policy in Ranger

To add a new policy to an existing Trino service, you should perform the following actions:

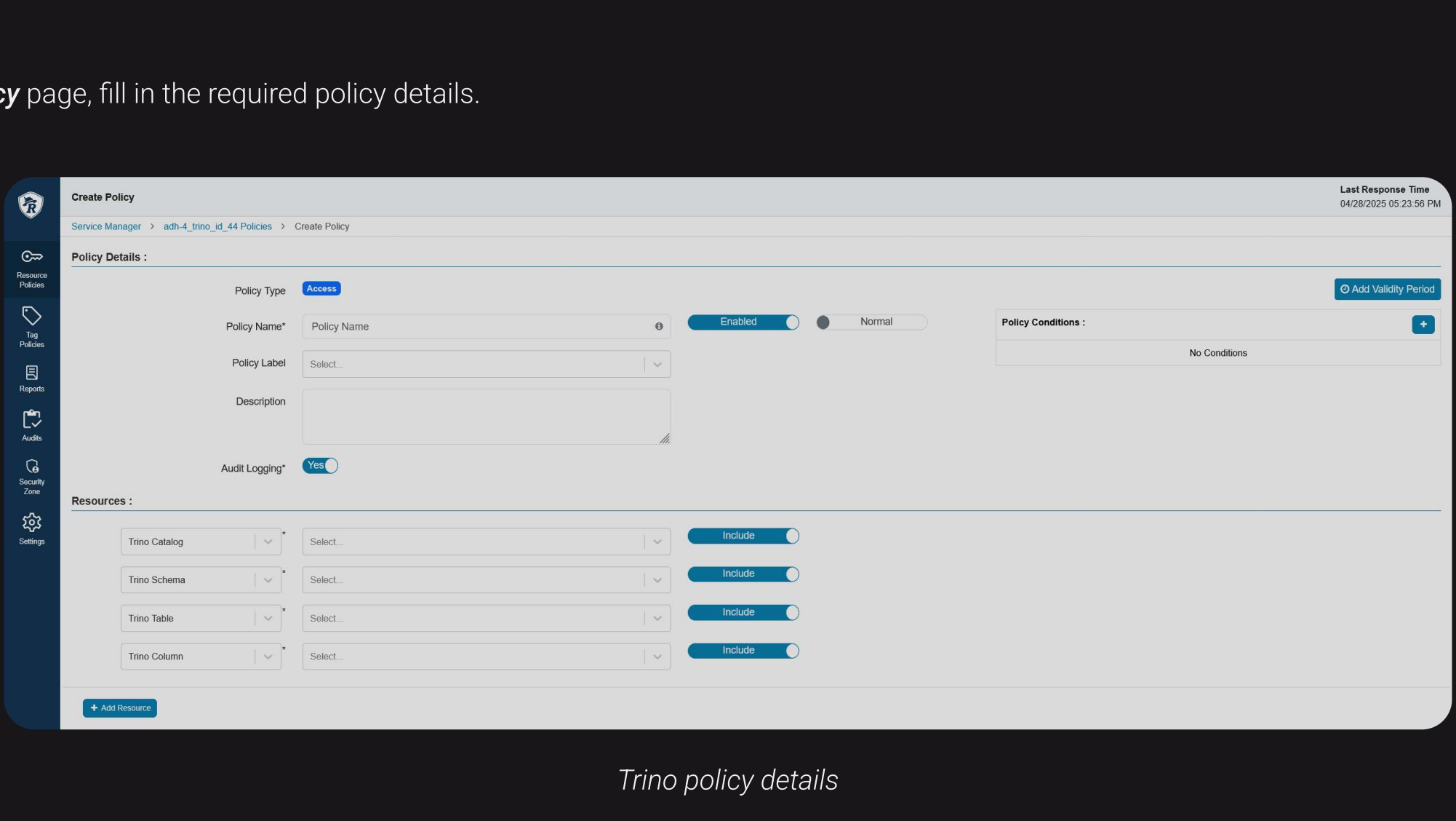
1. On the ***Service Manager*** page, click an existing Trino service in the Trino pane.



Service Manager
Last Response Time
04/28/2025 05:16:19 PM
Resource
Tag
Security Zone :
Select Zone Name
Import
Export
HDFS
YARN
SOLR
KYLIN
ATLAS
KUDU
TRINO
adh-4_trino_id_44
HBASE
KNOX
KAFKA
NIFI-REGISTRY
ELASTICSEARCH
OZONE
NESTEDSTRUCTURE
HADOOP SQL
STORM
NIFI
SQOOP
PRESTO
SCHEMA-REGISTRY
Licensed under the Apache License, Version 2.0

Service Manager

2. On the Trino policy page, click ***Add New Policy***.

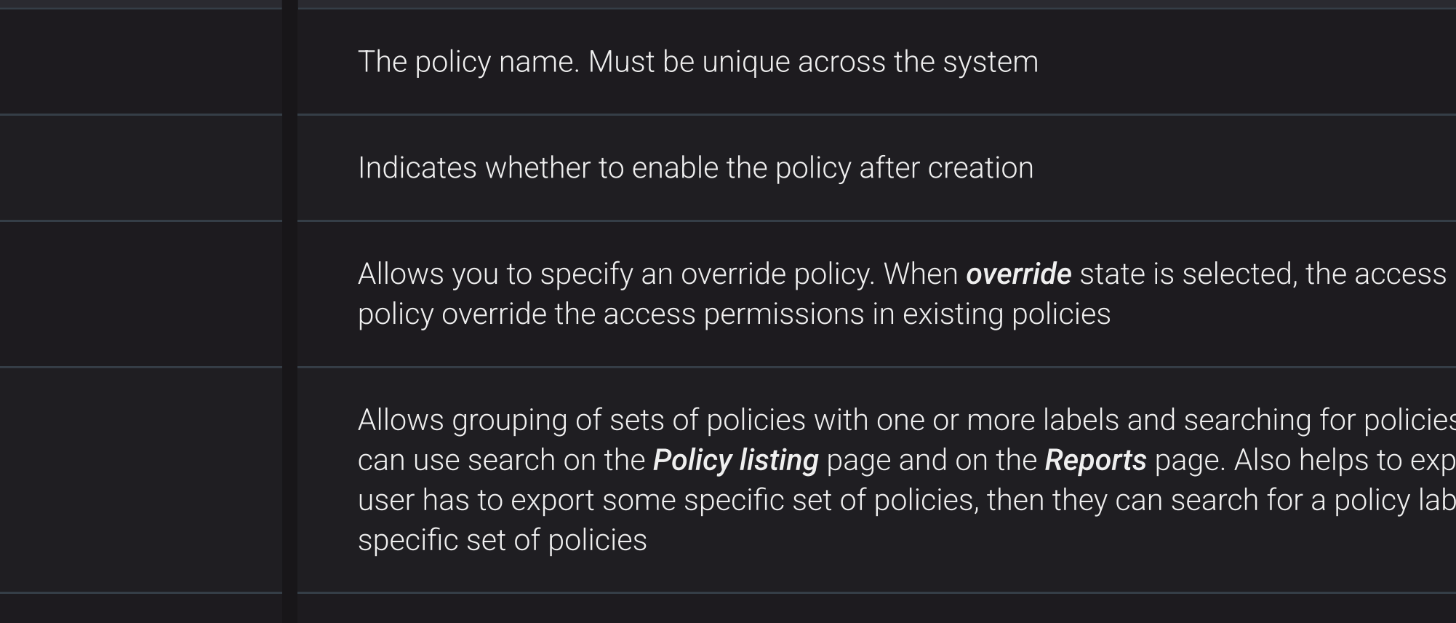


TRINO Policies
Service : adh-4_trino_id_44
Select Zone Name
Manage Service
Last Response Time
04/28/2025 05:22:32 PM
Access
Masking
Row Level Filter
Search for your policy...
Add New Policy
Import New Policy
Licensed under the Apache License, Version 2.0

Policy ID	Policy Name	Policy Label	Status	Audit Logging	Roles	Groups	Users
2	all - trinouser	--	Enabled	Enabled	--	--	trino
3	all - catalog, schema, schemafunction	--	Enabled	Enabled	--	--	trino
4	all - catalog, schema, procedure	--	Enabled	Enabled	--	--	trino
5	all - catalog, sessionproperty	--	Enabled	Enabled	--	--	trino
6	all - sysinfo	--	Enabled	Enabled	--	--	trino
7	all - catalog, schema, table, column	--	Enabled	Enabled	--	--	trino
8	all - catalog, schema	--	Enabled	Enabled	--	--	trino
9	all - catalog	--	Enabled	Enabled	--	--	trino
10	all - queryid	--	Enabled	Enabled	--	--	trino
11	all - role	--	Enabled	Enabled	--	--	trino
12	all - function	--	Enabled	Enabled	--	--	trino
13	all - catalog, schema, table	--	Enabled	Enabled	--	--	trino
14	all - systemproperty	--	Enabled	Enabled	--	--	trino

Add new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.



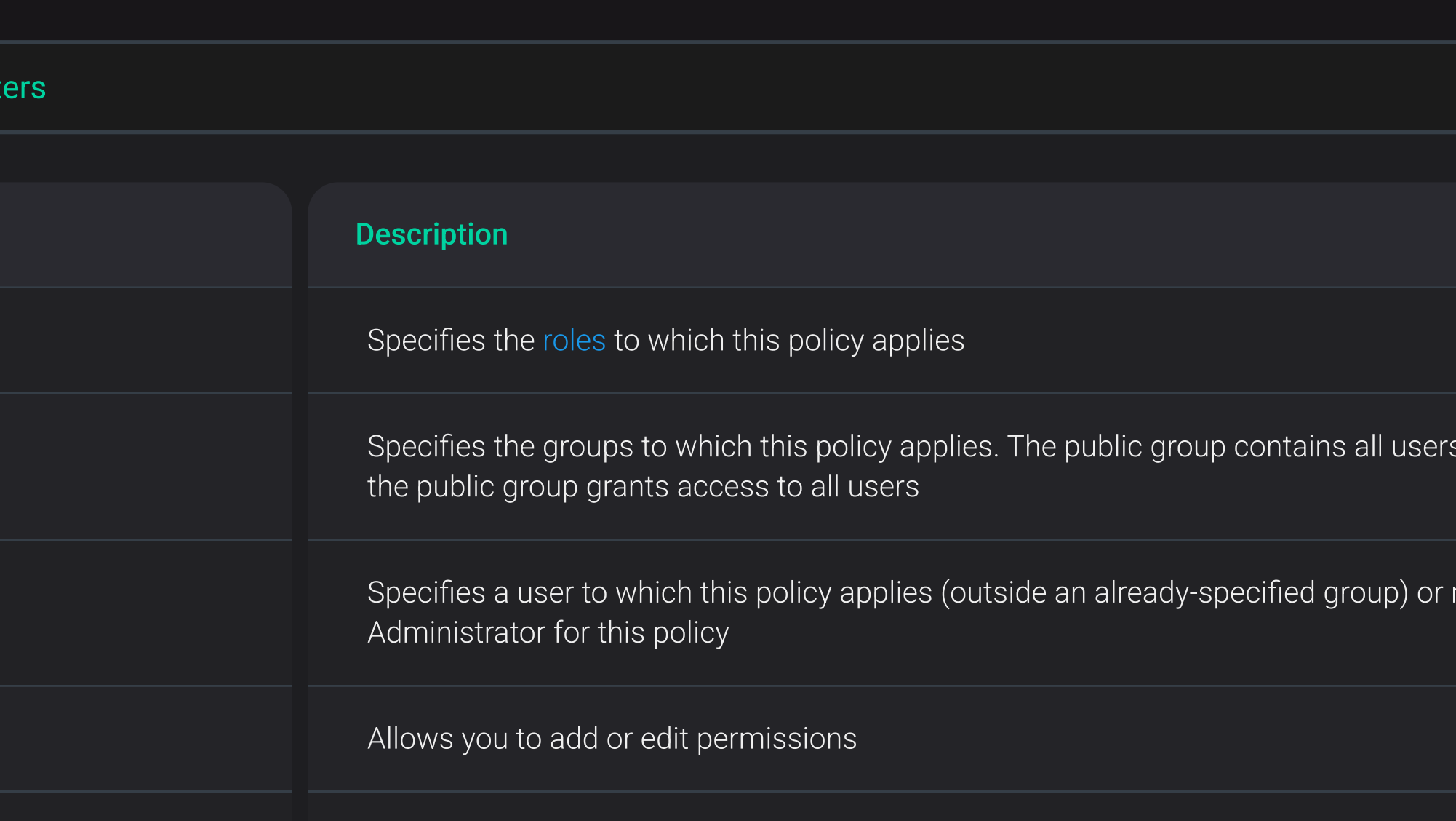
Create Policy
Service Manager › adh-4_trino_id_44 Policies › Create Policy
Last Response Time
04/28/2025 05:23:56 PM
Policy Details :
Policy Type
Access
Policy Name*
Policy Label
Description
Audit Logging*
Enabled
Normal
Add Validity Period
Policy Conditions :
No Conditions
Resources :
Trino Catalog
Trino Schema
Trino Table
Trino Column
Select...
Include
Add Resource

Trino policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy
Trino Catalog	Name of a Trino catalog
Trino Schema	Name of a Trino schema
Trino Table	Name of a Trino table
Trino Column	Name of a Trino column
Trino User	Name of a Trino user
System Property	Name of a system property
Trino Function	Name of a Trino function
Query ID	Query identifier
System Information	Information about the Trino cluster system environment
Role	Name of a Trino role

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click **+**. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Create Policy
Service Manager › adh_hbase_id_15 Policies › Create Policy
Last Response Time
09/25/2024 12:41:44 AM
Allow Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Add Conditions
Add Permissions
Exclude from Allow Conditions:
Deny All Other Accesses: *
False
Deny Conditions:
Exclude from Deny Conditions:

Trino allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Ranger YARN plugin

Sergei Tikhomirov

Contents

To Table of Contents

Enable YARN plugin

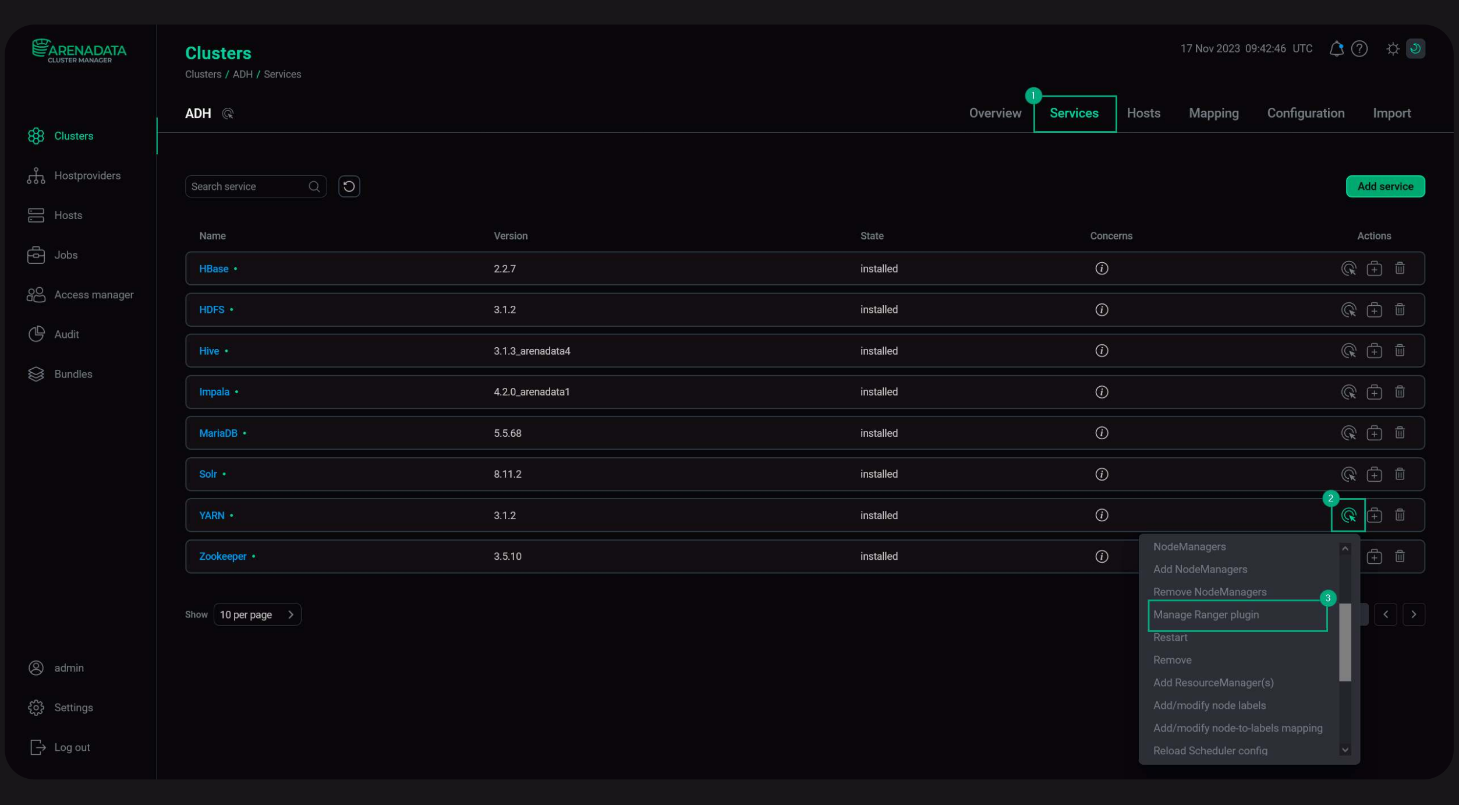
Add a new policy in Ranger

Enable YARN plugin

To enable the Ranger YARN plugin, follow the steps below:

1. Go to the ***Clusters*** → ***<ADH_cluster>*** → ***Services*** page.

2. Find YARN and click the icon in the ***Actions*** column. In the drop-down menu, select the ***Manage Ranger plugin*** action.

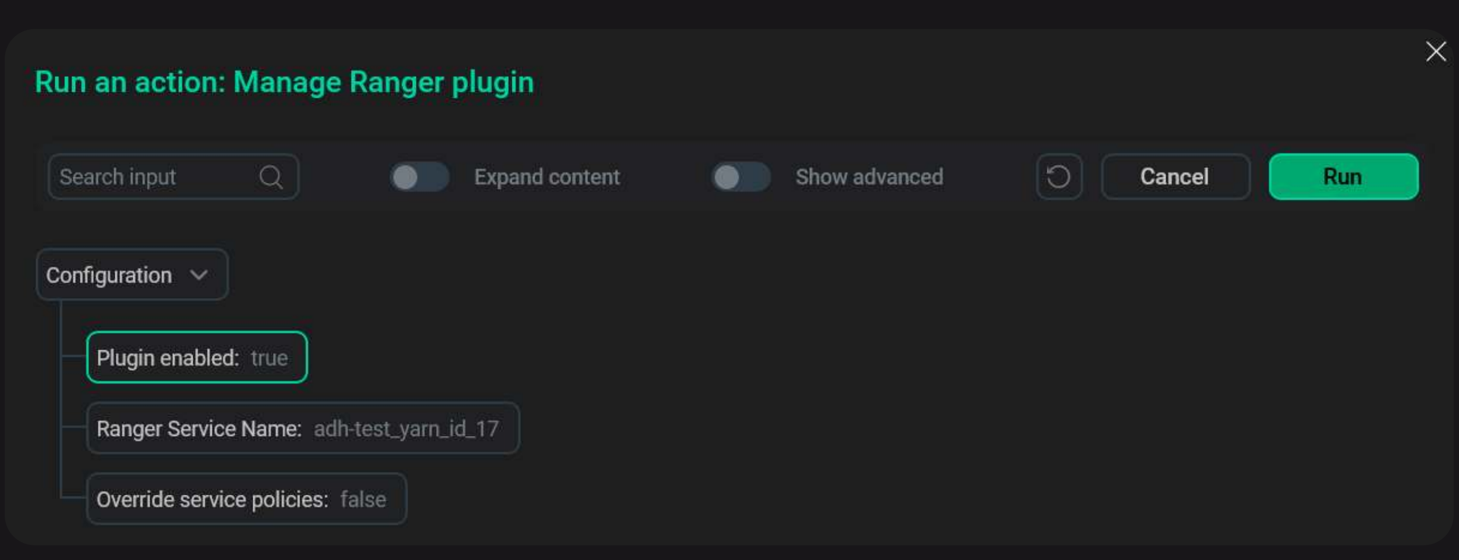


Name	Version	State
HBase	2.2.7	installed
HDFS	3.1.2	installed
Hive	3.1.3_arenadata4	installed
Impala	4.2.0_arenadata1	installed
MariaDB	5.5.68	installed
Solr	8.11.2	installed
YARN	3.1.2	installed
Zookeeper	3.5.10	installed

ARENADATA
CLUSTER MANAGER
Clusters
Clusters / ADH / Services
17 Nov 2023 09:42:46 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add service
Concerns
Actions
NodeManagers
Add NodeManagers
Remove NodeManagers
Manage Ranger plugin
Restart
Remove
Add ResourceManager(s)
Add/modify node labels
Add/modify node-to-labels mapping
Reload Scheduler config
Show
10 per page
admin
Settings
Log out
1
2
3

Cluster services

3. Select the required state of the ***Plugin enabled*** flag. Also, here you can set the name of the Ranger service that will be added. If a service with such name already exists, you can override it by enabling the ***Override service policies*** parameter — in that case, the old service will be deleted and policies will be generated for the new service.



Run an action: Manage Ranger plugin

Plugin enabled: true

Ranger Service Name: adh-test_yarn_id_17

Override service policies: false

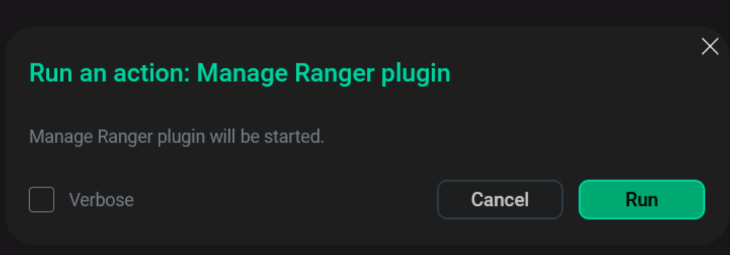
Search input
Expand content
Show advanced
Cancel
Run
Configuration

Plugin state

NOTE

On the first plugin enabling, a service with policies will be created (if it doesn't exist yet) regardless of the ***Override service policies*** parameter.

4. Click ***Run*** and confirm the action in the pop-up window.



Run an action: Manage Ranger plugin

Manage Ranger plugin will be started.

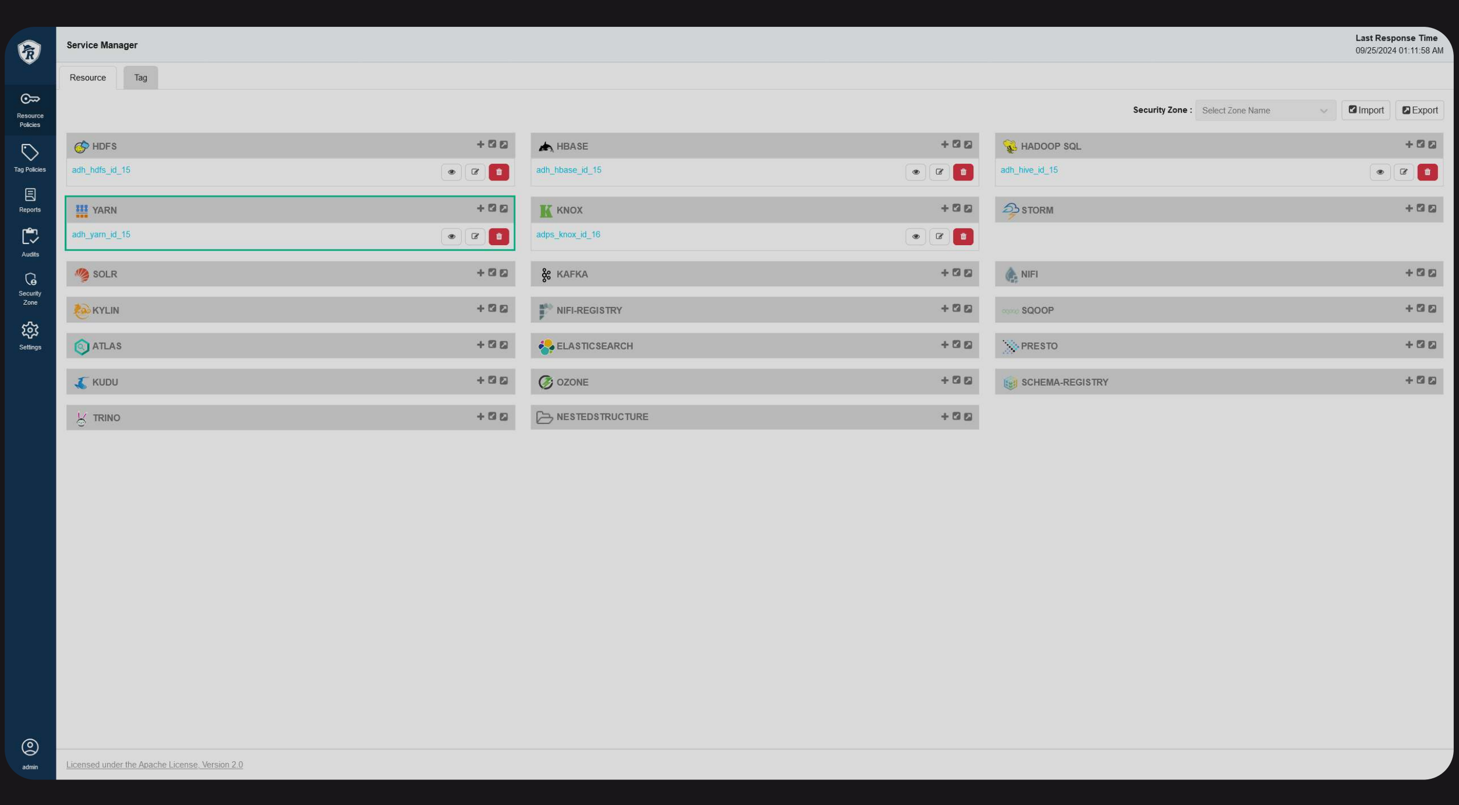
☐ Verbose
Cancel
Run

Action confirmation

Add a new policy in Ranger

To add a new policy to an existing YARN service, you should perform the following actions:

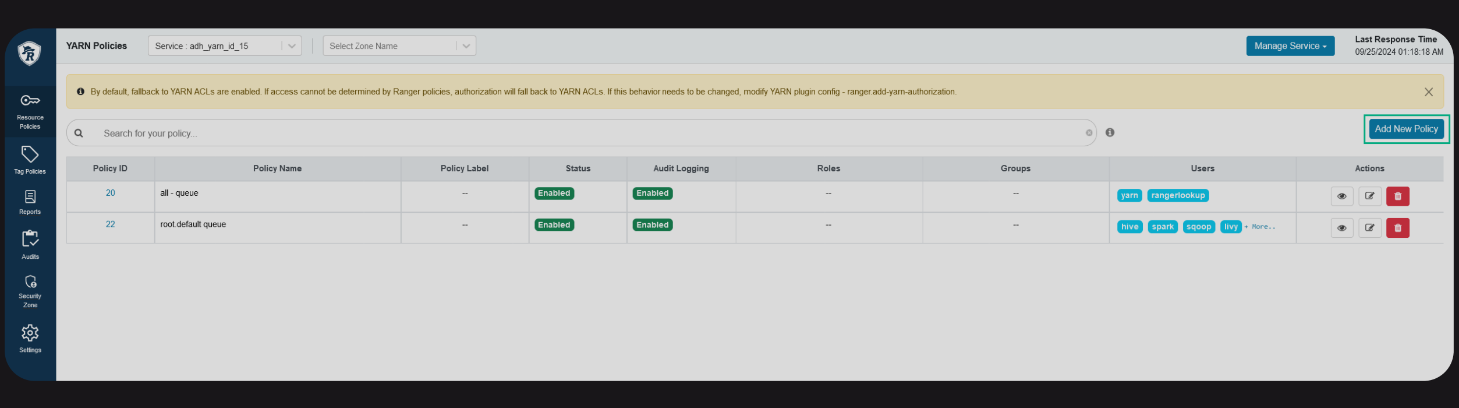
1. On the ***Service Manager*** page, click an existing service in the YARN pane.



Service Manager
Last Response Time
09/25/2024 01:11:58 AM
Resource
Tag
Security Zone :
Select Zone Name
Import
Export
HDFS
adh_hdfs_id_15
HBASE
adh_hbase_id_15
HADOOP SQL
adh_hive_id_15
YARN
adh_yarn_id_15
KNOX
adps_knox_id_16
STORM
SOLR
KAFKA
NIFI
KYLIN
NIFI-REGISTRY
SQOOP
ATLAS
ELASTICSEARCH
PRESTO
KUDU
OZONE
SCHEMA-REGISTRY
TRINO
NESTEDSTRUCTURE
Resource Policies
Tag Policies
Reports
Audits
Security Zone
Settings
admin
Licensed under the Apache License, Version 2.0

Service Manager

2. On the YARN policy page, click ***Add New Policy***.



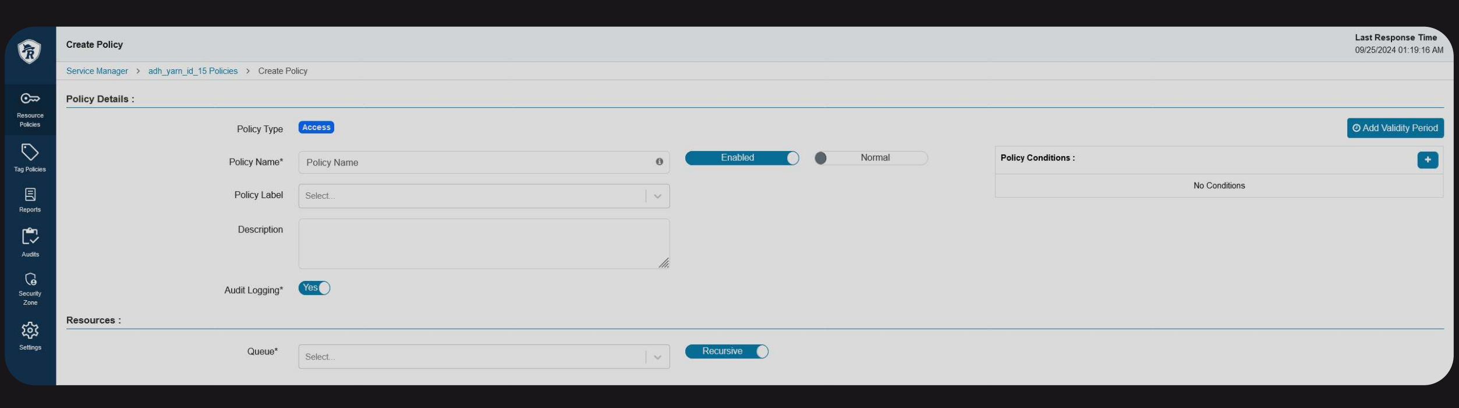
By default, fallback to YARN ACLs are enabled. If access cannot be determined by Ranger policies, authorization will fall back to YARN ACLs. If this behavior needs to be changed, modify YARN plugin config - ranger.add-yarn-authorization.

Policy ID	Policy Name	Policy Label	Status	Audit Logging	Roles	Groups	Users	Actions
20	all - queue	–	Enabled	Enabled	–	–	yarn rangerlookup	
22	root default queue	–	Enabled	Enabled	–	–	hive spark sqoop livy + More..	

YARN Policies
Service : adh_yarn_id_15
Select Zone Name
Manage Service
Last Response Time
09/25/2024 01:18:18 AM
Search for your policy...
Add New Policy

Add new policy

3. On the opened ***Create Policy*** page, fill in the required policy details.



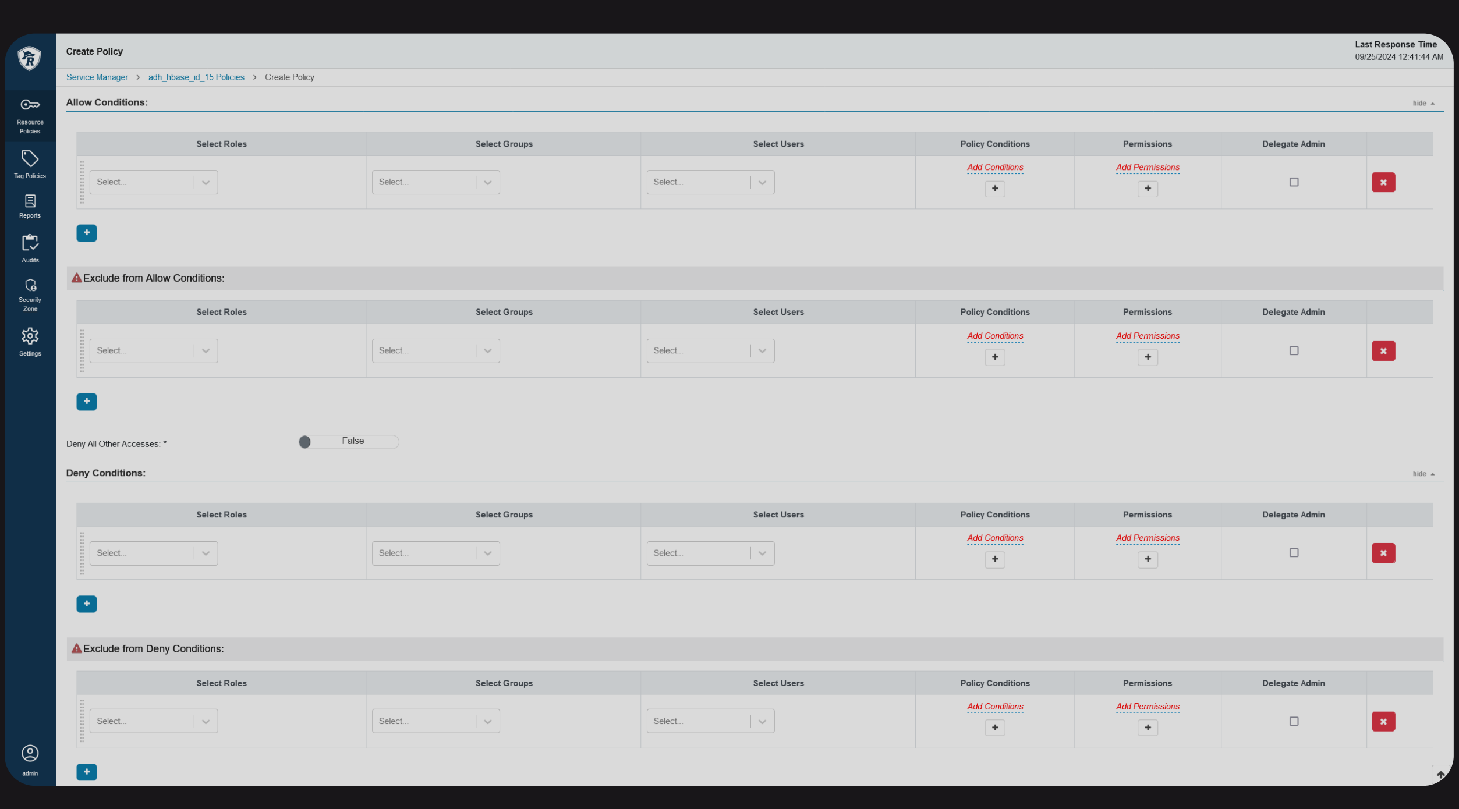
Create Policy
Last Response Time
09/25/2024 01:19:16 AM
Service Manager > adh_yarn_id_15 Policies > Create Policy
Policy Details :
Policy Type
Access
Policy Name*
Policy Name
Enabled
Normal
Policy Label
Select...
Description
Audit Logging*
Yes
Add Validity Period
Policy Conditions :
No Conditions
Resources :
Queue*
Select...
Recursive

YARN policy details

Policy details parameters

Parameter	Description
Policy Name	The policy name. Must be unique across the system
Enabled	Indicates whether to enable the policy after creation
Normal/Override	Allows you to specify an override policy. When ***override*** state is selected, the access permissions of the new policy override the access permissions in existing policies
Policy Label	Allows grouping of sets of policies with one or more labels and searching for policies by label names. You can use search on the ***Policy listing*** page and on the ***Reports*** page. Also helps to export/import policies. If a user has to export some specific set of policies, then they can search for a policy label and export the specific set of policies
Queue	The fundamental unit of scheduling in YARN
Recursive/Non-recursive	Indicates whether all files and folders within the existing folder should be affected by the policy. Can be used instead of wildcard characters
Description	Describes the purpose of the policy
Audit Logging	Enables audit for the policy
Add Validity Period	Allows you to set the lifetime for the policy

4. Set allow/deny condition parameters. The ***Allow Conditions*** section lets you grant access permissions to certain roles/groups/users. It makes sense to use this section if you want to allow acess to a few roles/groups/users and deny it to all others. The ***Deny Conditions*** sections in the reverse logic compared to ***Allow Conditions*** — it's useful when you want to deny access to a few roles/groups/users and allow it to everyone else. You can also combine rules from both sections. To add additional conditions, click +. Conditions take priority in the order listed in the policy. The condition at the top of the list is applied first, then the second, then the third, and so on.



Deny All Other Accesses: * False

Create Policy
Last Response Time
09/25/2024 12:41:44 AM
Service Manager > adh_hbase_id_15 Policies > Create Policy
Allow Conditions:
Exclude from Allow Conditions:
Deny Conditions:
Exclude from Deny Conditions:
Select Roles
Select Groups
Select Users
Policy Conditions
Permissions
Delegate Admin
Select...
Add Conditions
Add Permissions

YARN allow conditions parameters

Allow Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy applies
Select Group	Specifies the groups to which this policy applies. The public group contains all users, so granting access to the public group grants access to all users
Select User	Specifies a user to which this policy applies (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

Deny Conditions parameters

Parameter	Description
Select Role	Specifies the roles to which this policy does not apply
Select Group	Specifies the groups to which this policy does not apply. The public group contains all users, so denying access to the public group denies access to all users
Select User	Specifies a user to which this policy does not apply (outside an already-specified group) or makes the user an Administrator for this policy
Permissions	Allows you to add or edit permissions
Delegate Admin	Grants administrator privileges to the users or groups specified in the policy. Administrators can edit or delete policies and create child policies based on the existing ones

5. Click ***Add*** at the bottom of the page.

Kerberos overview

Mikhail Serov

Contents

To Table of Contents

Common requirements

Configuring Hadoop Group Mapping for LDAP/AD

The cluster can be kerberized with the use of the following KDC types:

- MIT KDC that consists of the principals database and Kerberos keys storage.
- Active Directory that consists of the principals database and Windows Server keys storage.
- FreeIPA which is a free open source identity management system for Linux/UNIX environments.
- Samba — software that can be used as an Active Directory domain controller.

The process of identification and authentication via either scheme is practically similar. The only difference is the KDC type being used.

Common requirements

This section describes the common prerequisites that are necessary to perform the kerberization of an ADH cluster.

To kerberize an ADH cluster, you have to assure the following:

- All machines have IPTables and SELinux turned off.
- ADH cluster is installed and set up. You can turn on Kerberos only if the cluster is ready for work.
- All cluster nodes are set up for straight and for reverse DNS.
- There is an Active Directory or a MIT Kerberos Administrator account that has full access to create, remove, and manage user accounts.
- Active Directory or MIT Kerberos keys and users storage is set up and ready.

Configuring Hadoop Group Mapping for LDAP/AD

This section describes the prerequisites that are necessary to set up Active Directory for Kerberos.

To provide forced authorization at the LDAP/AD group level in Hadoop, you need to configure Hadoop group mapping for LDAP/AD.

TIP

LDAP settings may vary depending on the LDAP implementation used.

There are two ways to configure Hadoop group mapping.

- Configuring Hadoop Group mapping for LDAP/AD using SSSD (recommended).

  For the groups mapping, we recommend using SSSD or one of the following services for the Linux and LDAP connection:

  - Centrify;
  - NSLCD;
  - Winbind;
  - SAMBA.

  Most of the listed services allow you to search for a user, list groups, and perform other actions on the host. However, none of these actions are required to map LDAP groups in Hadoop. Therefore, when evaluating these services it's necessary to understand the difference between the NSS module (which performs user/group authorization) and the PAM module (which performs user authentication). NSS is required to search or validate a user in LDAP and enumerate groups. PAM can also pose a security threat.

- Configuring Hadoop group mapping via ADCM (***core-site.xml*** section).
  Configuration of the LDAP-based group mapping in the *core-site file.xml* file is implemented in the following order:

  - Add the properties shown in the example below to the *core-site.xml* file. You should specify a value for the linked user, a password, and other properties specific to the LDAP instance. Make sure that the filters of object classes, users, and groups match the values specified in the LDAP instance.

Configuring authentication settings in core-site.xml

```
<property>
<name>hadoop.security.group.mapping</name>
<value>org.apache.hadoop.security.LdapGroupsMapping</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.bind.user</name>
<value>cn=Manager,dc=hadoop,dc=apache,dc=org</value>
</property>

<!–
<property>
<name>hadoop.security.group.mapping.ldap.bind.password.file</name>
<value>/etc/hadoop/conf/ldap-conn-pass.txt</value>
</property>
–>

<property>
<name>hadoop.security.group.mapping.ldap.bind.password</name>
<value>hadoop</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.url</name>
<value>ldap://localhost:389/dc=hadoop,dc=apache,dc=org</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.base</name>
<value></value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.user</name>
<value>(&amp;(|(objectclass=person)(objectclass=applicationProcess))(cn={0}))</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.filter.group</name>
<value>(objectclass=groupOfNames)</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.member</name>
<value>member</value>
</property>

<property>
<name>hadoop.security.group.mapping.ldap.search.attr.group.name</name>
<value>cn</value>
</property>
```

  - Depending on the configuration, you can update user and group mappings using the following HDFS and YARN commands:

```
$ hdfs dfsadmin -refreshUserToGroupsMappings
$ yarn rmadmin -refreshUserToGroupsMappings
```

  - Check the mapping of LDAP groups by running the `hdfs groups` command. The command displays groups from LDAP for the current user. With LDAP group mapping configured, HDFS permissions can use LDAP-defined groups for access control.

Set up MIT Kerberos server

Mikhail Serov

To Table of Contents

The cluster to be kerberized should be fully prepared and configured. The configuration can be any. Before you start configuring Kerberos for the cluster, the scheme should contain the MIT KDC principals database and ADPS users database. The logins for all users and services should match in both databases. After Kerberos is configured for the cluster, every user and service will have the corresponding principal.

Additionally to MIT Kerberos, the scheme uses ADPS to manage users permissions. MIT Kerberos handles principals authentication, while ADPS handles principals permissions.

The MIT Kerberos principals database should be synchronized with the ADPS users database. This synchronization is performed manually before configuring Kerberos in the cluster. Every further change to the MIT Kerberos principals database should be duplicated in the ADPS users database.

In this case, while working with ADH cluster, all principals receive TGT from MIT Kerberos KDC.

Kerberos server settings

Mikhail Serov

Contents

To Table of Contents

Main Kerberos server setup

Secondary Kerberos server setup for high availability

Main Kerberos server setup

TIP

The instruction described in this section can be used to interchange the main and secondary KDC, if needed.

1. To install the Kerberos server package to the server, run the command:

```
$ sudo yum install -y krb5-server pam_krb5 krb5-workstation xinetd
```

2. To change the configuration, edit the */etc/krb5.conf* file:

krb5.conf

```
# Configuration snippets may be placed in this directory as well
includedir /etc/krb5.conf.d/
[logging]
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log
[libdefaults]
dns_lookup_realm = false
ticket_lifetime = 24h
renew_lifetime = 7d
forwardable = true
rdns = false
pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
default_realm = ADREALM.IO
default_ccache_name = KEYRING:persistent:%{uid}
[realms]
ADREALM.IO = {
kdc = kerberos-primary-host.ru-central1.internal #primary KDC host
kdc = kerberos-secondary-host.ru-central1.internal #secondary KDC host
admin_server = kerberos-primary-host.ru-central1.internal
}
[domain_realm]
.adrealm.io = ADREALM.IO
adrealm.io  = ADREALM.IO
```

3. You also need to edit the */var/kerberos/krb5kdc/kdc.conf* file:

kdc.conf

```
[kdcdefaults]
kdc_ports = 88
kdc_tcp_ports = 88

[realms]
ADREALM.IO = {
#master_key_type = aes256-cts
acl_file = /var/kerberos/krb5kdc/kadm5.acl
dict_file = /usr/share/dict/words
admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal arcfour-hmac:normal camellia256-cts:normal
camellia128-cts:normal des-hmac-sha1:normal des-cbc-md5:normal des-cbc-crc:normal
}
```

4. The last file that should be edited before the Kerberos database setup is */var/keberos/krb5kdc/kadm5.acl*:

kadm5.acl

```
*/admin@ADREALM.IO *
```

5. Now you have to configure the Kerberos database. For this, use the `kdb5_util` utility:

```
$ sudo kdb5_util create -s -r ADREALM.IO
```

6. Memorize the Master KDC password and run the services:

```
$ sudo systemctl enable kadmin
$ sudo systemctl enable krb5kdc
$ sudo systemctl start kadmin
$ sudo systemctl start krb5kdc
```

7. Now you have to create the new principals. You need to create host principals for every KDC instance. In order to do this, use the Kerberos administrator console:

```
$ sudo kadmin.local
$ addprinc root/admin
$ addprinc -randkey host/kerberos-primary-host.ru-central1.internal
$ addprinc -randkey host/kerberos-secondary-host.ru-central1.internal
$ ktadd host/kerberos-primary-host.ru-central1.internal
```

8. Generate the keytab of the secondary Kerberos server on the main Kerberos server:

```
$ ktadd -k /tmp/kerberos-secondary-host.keytab host/kerberos-secondary-host.ru-central1.internal
```

Secondary Kerberos server setup for high availability

1. Install the necessary packages to the second server:

```
$ sudo yum install -y krb5-server krb5-workstation xinetd
```

2. To enable password authentication, uncomment the following string in the */etc/ssh/sshd_config* file:

```
PasswordAuthentication yes
```

3. Restart the service:

```
$ sudo systemctl restart sshd
```

4. Copy the secondary server keytab file (that we obtained at the previous step) to the secondary KDC host:

```
$ sudo scp /tmp/kerberos-secondary-host.keytab <user>@<ip address>:/tmp/
```

5. While at secondary host, move the keytab file to the */etc/* directory and rename it:

```
$ sudo cp /tmp/kerberos-secondary-host.keytab /etc/krb5.keytab
```

6. The principal database is transferred from the main KDC to KDC replicas via the *kpropd* daemon. You have to specify the principals that are permitted to receive the updates of secondary Kerberos server dumps. Those updates contain the new database. You have to create the *kpropd.acl* file in the KDC state-monitoring directory. The *kpropd.acl* file will contain the principal host names for every KDC:

```
$ sudo vi /var/kerberos/krb5kdc/kpropd.acl
```

7. Add the following lines to *kpropd.acl*:

```
host/kerberos-primary-host.ru-central1.internal@ADREALM.IO
host/kerberos-secondary-host.ru-central1.internal@ADREALM.IO
```

NOTE

If you expect that the primary and secondary KDC might swap their places, you should specify the host addresses of all KDC servers in the *kpropd.acl* files on every KDC. Otherwise, you have to specify only the primary host in the *kpropd.acl* files on KDC replicas.

8. You should copy the following files from the primary KDC host to the secondary KDC host:
   - *krb5.conf*
   - *kdc.conf*
   - *kadm5.acl*
   - master key stash file

9. Move the files that were copied to the corresponding directories. The files and directories structure should be exactly like it is at the primary KDC.

TIP

The *kadm5.acl* file is needed to allow the replica to communicate with the primary KDC.

10. Run the following commands on the primary server:

```
$ sudo scp /etc/krb5.conf <user>@<ip address>:/tmp/
$ sudo scp /var/kerberos/krb5kdc/kdc.conf <user>@<ip address>:/tmp/
$ sudo scp /var/kerberos/krb5kdc/kadm5.acl <user>@<ip address>:/tmp/
$ sudo scp /var/kerberos/krb5kdc/.k5.ADREALM.IO <user>@<ip address>:/tmp/
$ sudo scp /var/kerberos/krb5kdc/kpropd.acl <user>@<ip address>:/tmp/
```

11. Run the following commands on the secondary server:

```
$ sudo cp /tmp/krb5.conf /etc/
$ sudo cp /tmp/kdc.conf /var/kerberos/krb5kdc/
$ sudo cp /tmp/kadm5.acl /var/kerberos/krb5kdc/
$ sudo cp /tmp/.k5.ADREALM.IO /var/kerberos/krb5kdc/
$ sudo cp /tmp/kpropd.acl /var/kerberos/krb5kdc/
```

12. Create the */etc/xinetd.d/krb5_prop* configuration file:

/etc/xinetd.d/krb5_prop

```
service krb5_prop
{
socket_type = stream
wait = no
user = root
server = /usr/sbin/kpropd
only_from = 0.0.0.0/0
log_on_success = PID HOST EXIT DURATION
}
```

13. Add the following line (if it’s missing) to the */etc/services* file on every KDC:

```
krb5_prop       754/tcp               # Kerberos replica propagation
```

14. Run the following command:

```
$ sudo systemctl restart xinetd
```

15. Before you copy the database, create the database dump file on the primary KDC:

```
$ sudo kdb5_util dump /var/kerberos/krb5kdc/slavedump
```

16. Copy the database manually to every KDC replica:

```
$ sudo kprop -f /var/kerberos/krb5kdc/slavedump kerberos-secondary-host.ru-central1.internal
```

The result should look like this:

```
Database propagation to kerberos-secondary-host.ru-central.internal: SUCCEEDED
```

17. Run KDC on the secondary host:

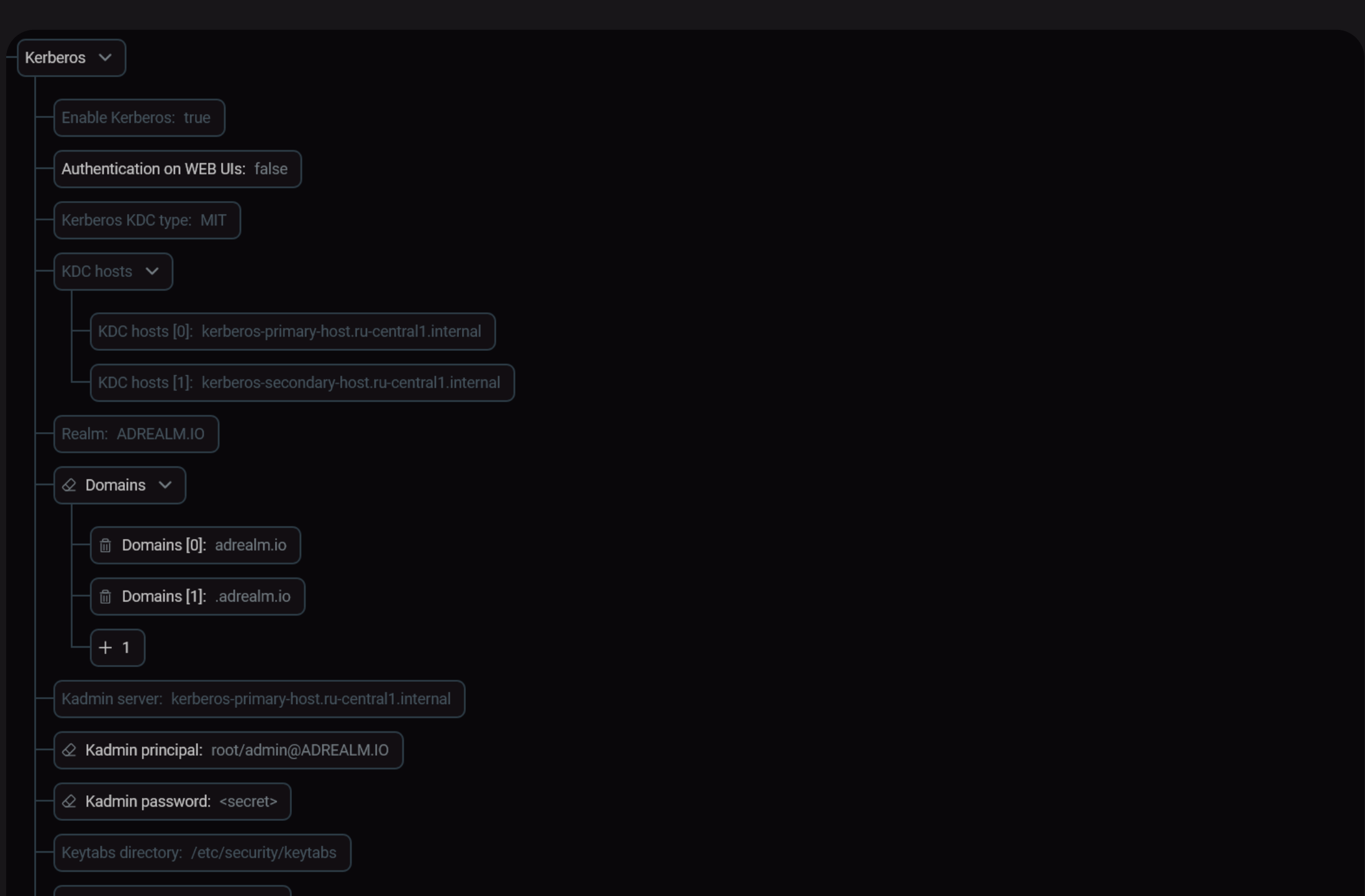
```
$ sudo systemctl enable krb5kdc
$ sudo systemctl start krb5kdc
```

18. Enter the `admin.local` console and execute the `listprincs` command to view the principals list and make sure that the list is the same as the list on the primary host.

In order to implement the principals database regular synchronization, you have to create a cron job. A script sample is below:

```
#!/bin/sh

kdclist = "kerberos-primary-host.ru-central1.internal kerberos-secondary-host.ru-central1.internal"
kdb5_util dump /var/kerberos/krb5kdc/slavedump
for kdc in $kdclist
do
    kprop -f /var/kerberos/krb5kdc/slavedump $kdc
done
```

An example of ADCM settings for the case of several KDC hosts:


Kerberos
Enable Kerberos: true
Authentication on WEB UIs: false
Kerberos KDC type: MIT
KDC hosts
KDC hosts [0]: kerberos-primary-host.ru-central1.internal
KDC hosts [1]: kerberos-secondary-host.ru-central1.internal
Realm: ADREALM.IO
Domains
Domains [0]: adrealm.io
Domains [1]: .adrealm.io
+ 1
Kadmin server: kerberos-primary-host.ru-central1.internal
Kadmin principal: root/admin@ADREALM.IO
Kadmin password: <secret>
Keytabs directory: /etc/security/keytabs
Additional realms <not set>

KDC hosts settings

Configure Kerberos authentication based on MIT Kerberos via ADCM

Mikhail Serov, Sergei Tikhomirov

Contents

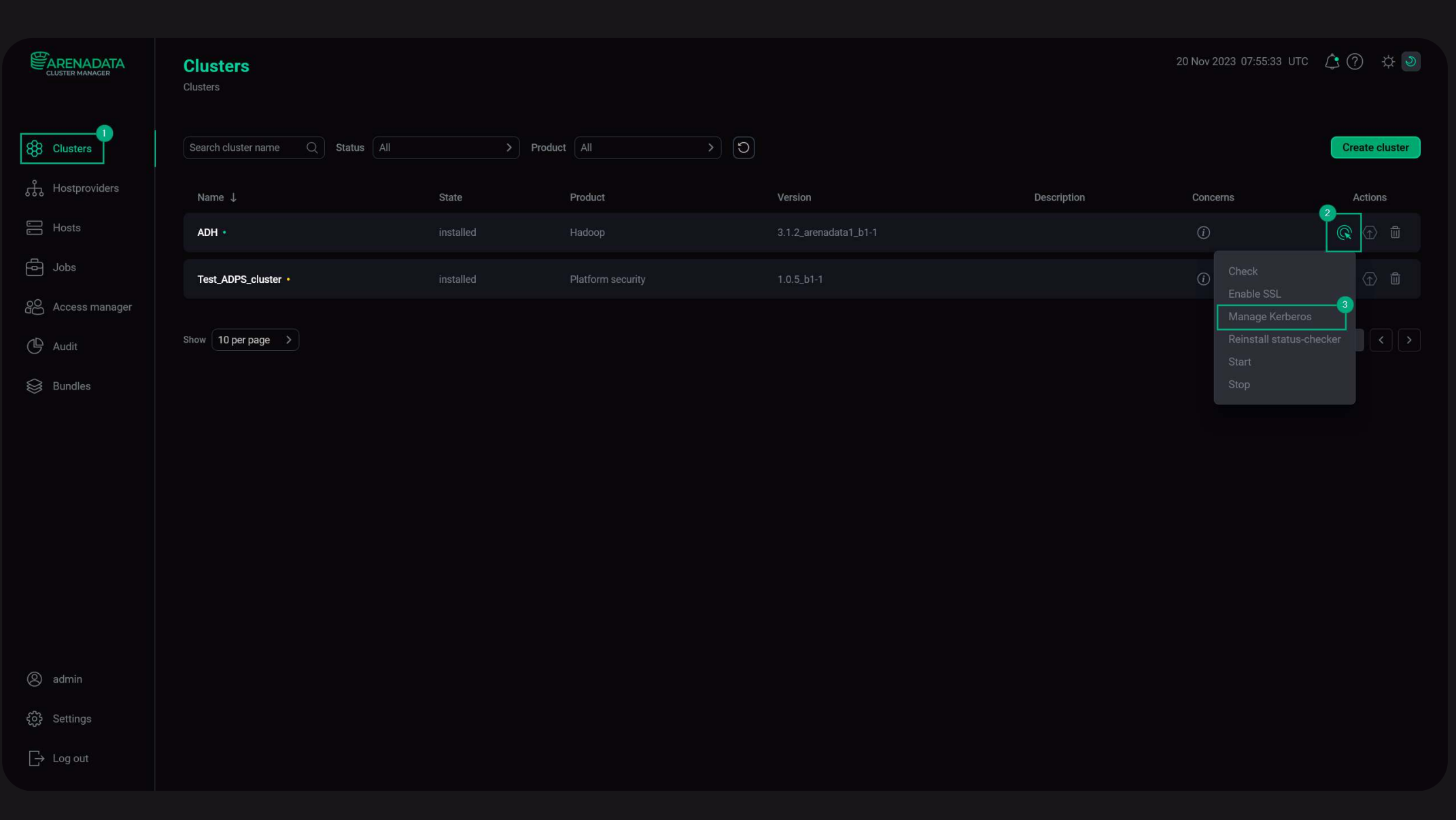
To Table of Contents

Overview
MIT Kerberos parameters
Verify Kerberos installation success using MIT Kerberos
  Check HDFS before the cluster is kerberized
  Check HDFS after the cluster is kerberized

Overview

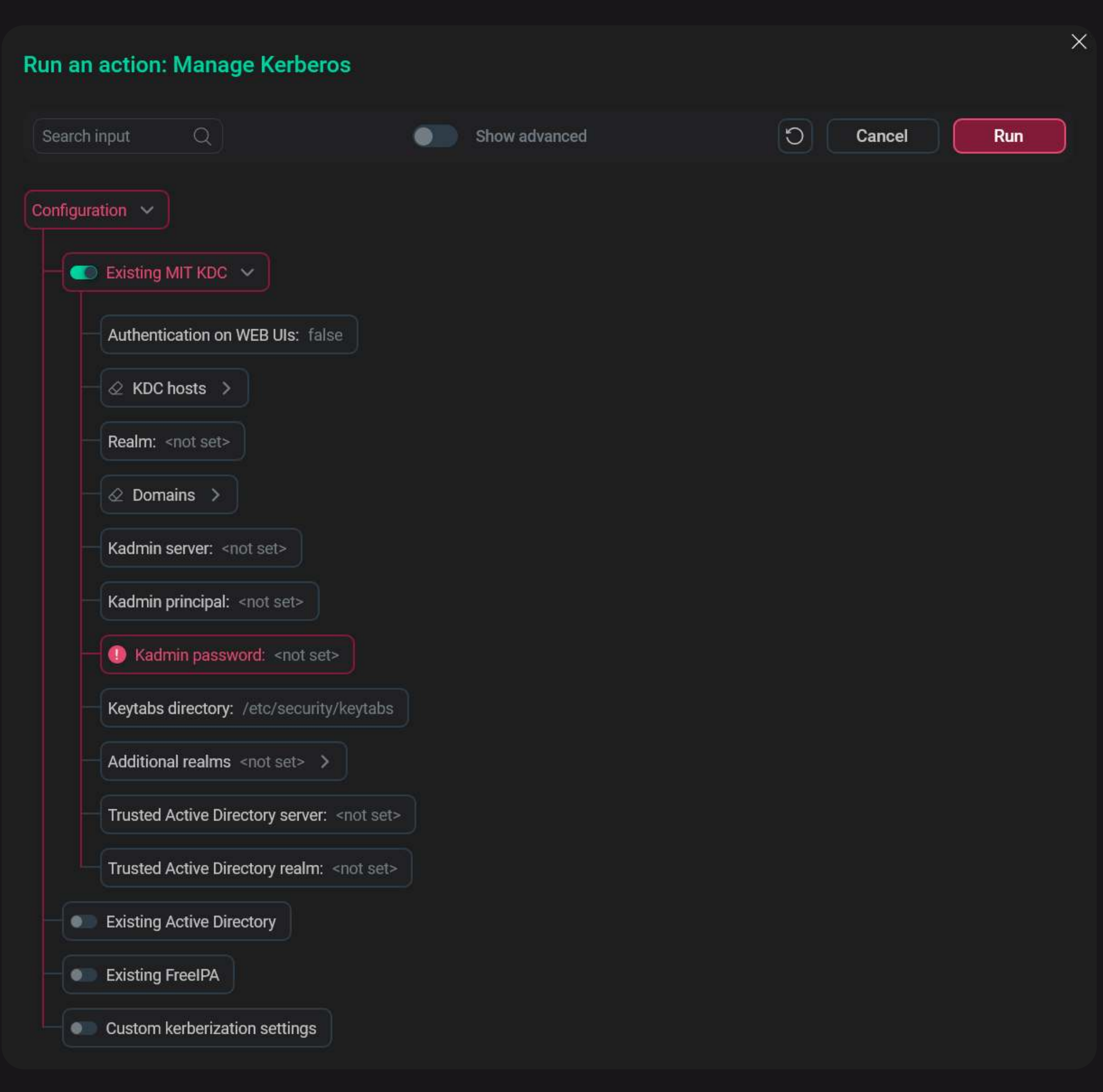
To kerberize a cluster using MIT KDC, follow the steps below:

1. In ADCM web UI, go to the ***Clusters*** page. Select an installed and prepared ADH cluster, and run the ***Manage Kerberos*** action.


ARENADATA CLUSTER MANAGER
Clusters
Clusters
20 Nov 2023 07:55:33 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
ADH
installed
Hadoop
3.1.2_arenadata1_b1-1
Test_ADPS_cluster
installed
Platform security
1.0.5_b1-1
Check
Enable SSL
Manage Kerberos
Reinstall status-checker
Start
Stop
Show
10 per page
1
2
3

Manage Kerberos

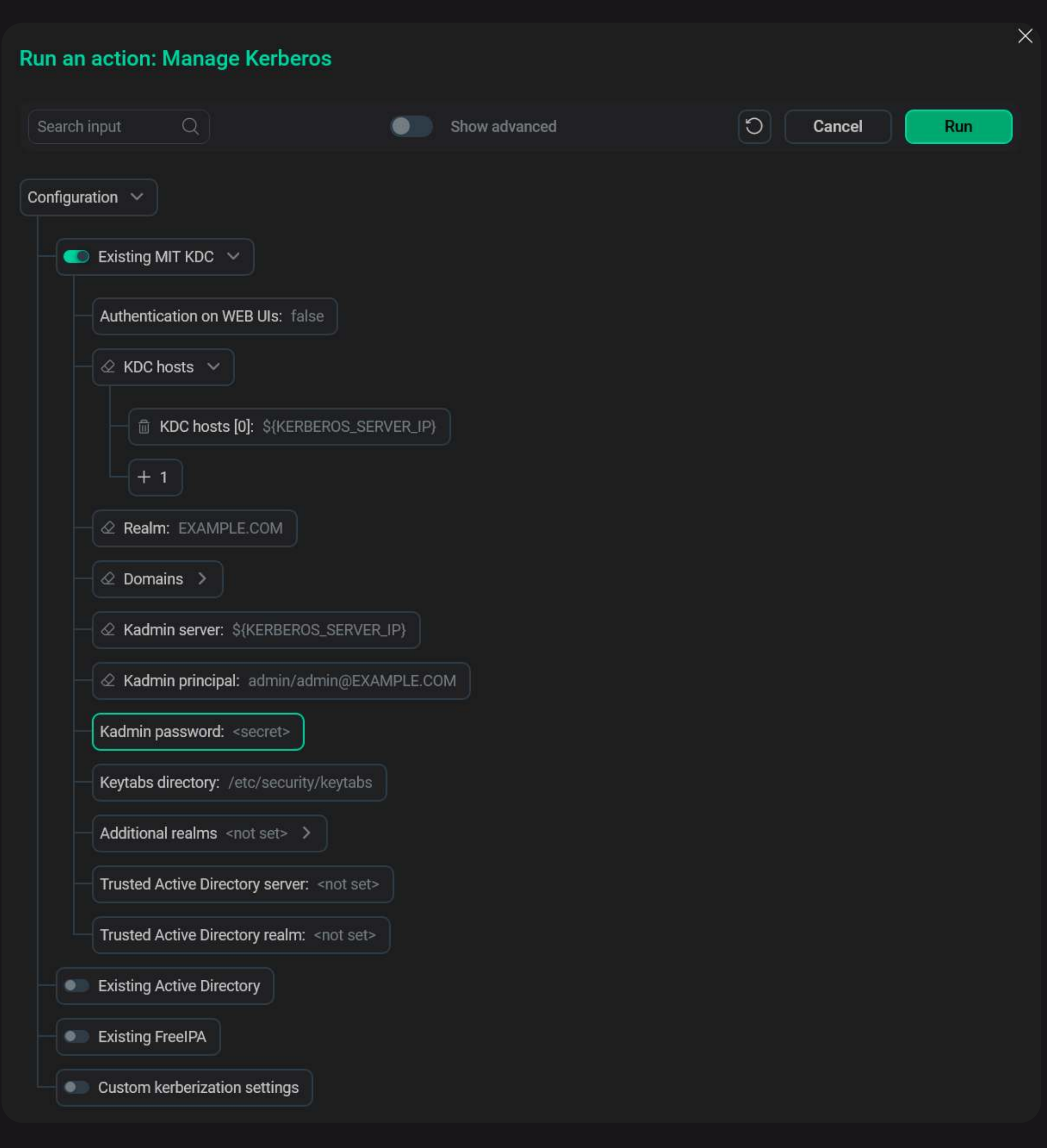
2. In the pop-up window, turn on the ***Existing MIT KDC*** option.


Run an action: Manage Kerberos
Search input
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
KDC hosts
Domains
Existing Active Directory
Existing FreeIPA
Custom kerberization settings

Authentication on WEB UIs: false
Realm: <not set>
Kadmin server: <not set>
Kadmin principal: <not set>
Kadmin password: <not set>
Keytabs directory: /etc/security/keytabs
Additional realms <not set>
Trusted Active Directory server: <not set>
Trusted Active Directory realm: <not set>

Choose the relevant option

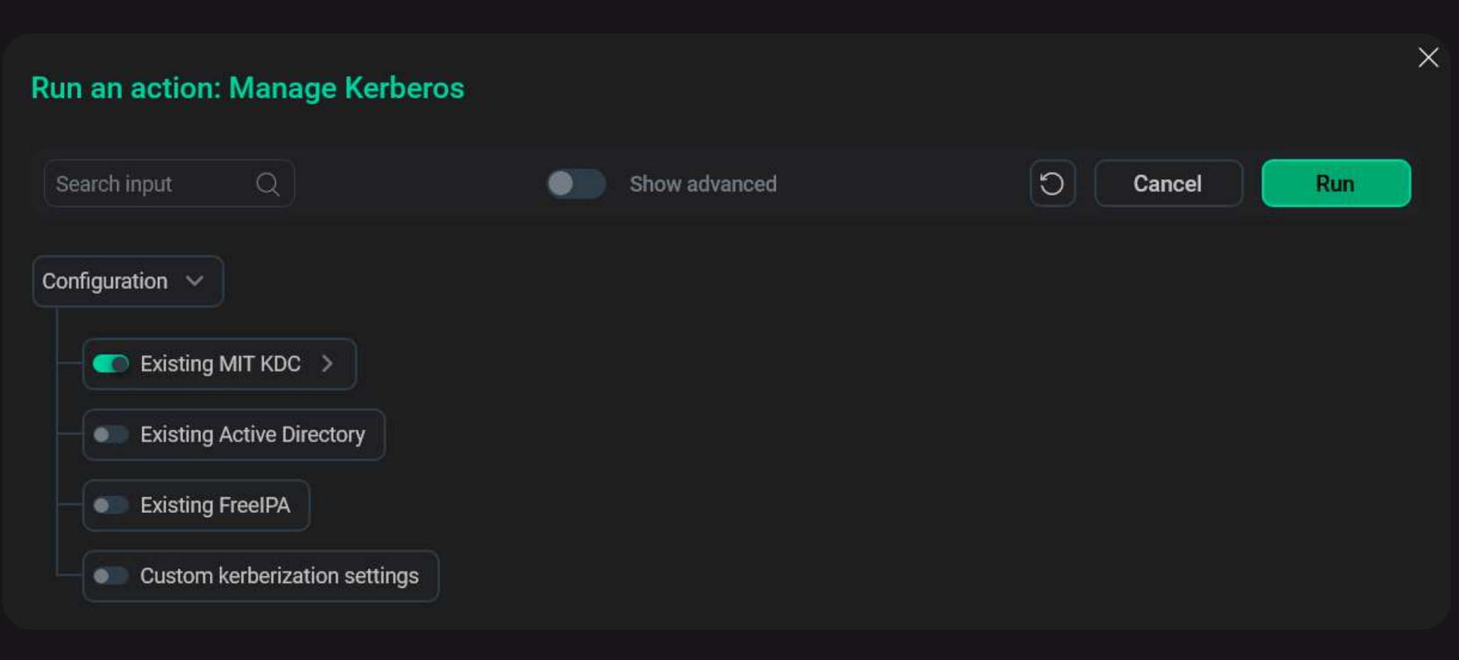
3. Fill in the prepared MIT Kerberos parameters.


Run an action: Manage Kerberos
Search input
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
KDC hosts
+ 1
Domains
Existing Active Directory
Existing FreeIPA
Custom kerberization settings

Authentication on WEB UIs: false
KDC hosts [0]: ${KERBEROS_SERVER_IP}
Realm: EXAMPLE.COM
Kadmin server: ${KERBEROS_SERVER_IP}
Kadmin principal: admin/admin@EXAMPLE.COM
Kadmin password: <secret>
Keytabs directory: /etc/security/keytabs
Additional realms <not set>
Trusted Active Directory server: <not set>
Trusted Active Directory realm: <not set>

MIT fields

4. Click ***Run***, wait for the job to complete and proceed to setting up Kerberos in the cluster.


Run an action: Manage Kerberos
Search input
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
Existing Active Directory
Existing FreeIPA
Custom kerberization settings

Run the action

MIT Kerberos parameters

Parameter	Description
Authentication on WEB UIs	Enable Kerberos authentication on Web UIs
KDC hosts	One or more MIT KDC hosts
Realm	A Kerberos realm — a network containing KDC hosts and clients
Domains	Domains associated with hosts
Kadmin server	A host where `kadmin` is running
Kadmin principal	A principal name used to connect via `kadmin`, for example `admin@RU-CENTRAL1.INTERNAL`
Kadmin password	A principal password used to connect via `kadmin`
Keytabs directory	Directory of the keytab file that contains one or several principals along with their keys
Additional realms	Additional Kerberos realms
Trusted Active Directory server	An Active Directory server for one-way cross-realm trust from the MIT Kerberos KDC
Trusted Active Directory realm	An Active Directory realm for one-way cross-realm trust from the MIT Kerberos KDC

Verify Kerberos installation success using MIT Kerberos

After the cluster is kerberized, any action of any user must run after the successful authentication. You can find the examples of HDFS requests before and after the cluster is kerberized using MIT Kerberos.

Check HDFS before the cluster is kerberized

HDFS CLI

```
$ sudo -u hdfs hdfs dfs -touch /tmp/arenadata_test.txt
$ sudo -u hdfs hdfs dfs -ls /tmp
```

The output looks as follows:

```
Found 1 items
-rw-r--r--   3 hdfs hadoop          0 2022-02-05 11:17 /tmp/arenadata_test.txt
```

HDFS HttpFS

The curl request to create an entity:

```
$ curl -I -H "Content-Type:application/octet-stream" -X PUT 'http://httpfs_hostname:14000/webhdfs/v1/tmp/arenadata_httpfs_test.
txt?op=CREATE&data=true&user.name=hdfs'
```

The response looks like the following:

```
HTTP/1.1 201 Created
Date: Sat, 05 Feb 2022 11:23:31 GMT
Cache-Control: no-cache
Expires: Sat, 05 Feb 2022 11:23:31 GMT
Date: Sat, 05 Feb 2022 11:23:31 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple-dt&e=1628198611064&s=gS6tylp5MZw+aiHs1EzuNfd1qqbJpFAGeGLxTtXxZfg="; Path=/;
HttpOnly
Content-Type: application/json;charset=utf-8
Content-Length: 0
```

The curl request to retrieve data about an HDFS file:

```
$ curl -H "Content-Type:application/json" -X GET 'http://httpfs_hostname:14000/webhdfs/v1/tmp?op=LISTSTATUS&user.name=hdfs'
```

The response JSON looks as follows:

```
{"FileStatuses":{"FileStatus":[{"pathSuffix":"arenadata_httpfs_test.
txt","type":"FILE","length":0,"owner":"hdfs","group":"hadoop","permission":"755","accessTime":1628162611089,"modificationTime"
:1628162611113,"blockSize":134217728,"replication":3}]}}
```

Check HDFS after the cluster is kerberized

HDFS CLI

HDFS request:

```
$ sudo -u hdfs hdfs dfs -ls /tmp
```

Response (unsuccessful):

```
2022-02-05 11:34:14,753 WARN ipc.Client: Exception encountered while connecting to the server : org.apache.hadoop.security.
AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS]
2022-02-05 11:34:14,765 WARN ipc.Client: Exception encountered while connecting to the server : org.apache.hadoop.security.
AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS]
2022-02-05 11:34:14,767 INFO retry.RetryInvocationHandler: java.io.IOException: DestHost:destPort namenode_host:8020 ,
LocalHost:localPort namenode_host/namenode_ip:0. Failed on local exception: java.io.IOException: org.apache.hadoop.security.
AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS], while invoking ClientNamenodeProtocolTranslatorPB.
getFileInfo over namenode_host/namenode_ip:8020 after 1 failover attempts. Trying to failover after sleeping for 1087ms.
```

Creating a ticket:

```
$ sudo -u hdfs kinit -k -t /etc/security/keytabs/hdfs.service.keytab hdfs/node_hostname@REALM
```

Response (successful):

```
Found 1 items
-rw-r--r--   3 hdfs hadoop          0 2022-02-05 11:17 /tmp/arenadata_test.txt
```

HDFS HttpFS

HDFS request:

```
$ sudo -u hdfs curl -H "Content-Type:application/json" -X GET 'http://httpfs_hostname:14000/webhdfs/v1/tmp?op=LISTSTATUS&user.
name=hdfs'
```

Response (unsuccessful):

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<title>Error 401 Authentication required</title>
</head>
<body><h2>HTTP ERROR 401</h2>
<p>Problem accessing /webhdfs/v1/tmp. Reason:
<pre>    Authentication required</pre></p>
</body>
</html>
```

Creating a ticket:

```
$ sudo -u hdfs kinit -k -t /etc/security/keytabs/HTTP.service.keytab HTTP/node_hostname@REALM
```

HDFS request:

```
$ sudo -u hdfs curl -sfSIL -I -H "Content-Type:application/octet-stream" -X PUT --negotiate -u :
'http://httpfs_hostname:14000/webhdfs/v1/tmp/arenadata_httpfs_test.txt?op=CREATE&data=true&user.name=hdfs'
```

Response (successful):

```
HTTP/1.1 201 Created
Date: Sat, 05 Feb 2022 11:23:31 GMT
Cache-Control: no-cache
Expires: Sat, 05 Feb 2022 11:23:31 GMT
Date: Sat, 05 Feb 2022 11:23:31 GMT
Pragma: no-cache
Set-Cookie: hadoop.auth="u=hdfs&p=hdfs&t=simple-dt&e=1628198611064&s=gS6tylp5MZw+aiHs1EzuNfd1qqbJpFAGeGLxTtXxZfg="; Path=/;
HttpOnly
Content-Type: application/json;charset=utf-8
Content-Length: 0
```

Set up Active Directory for Kerberos

Mikhail Serov

To Table of Contents

If the cluster is kerberized using the approach with Active Directory, Kerberos requests the information from KDC on the accounts of administrator and all users. After this information is obtained, Kerberos automatically creates principals for every user and service. You can create any configuration for your cluster, while every user and service will have the corresponding principal.

Additionally to Active Directory, the scheme uses ADPS to manage users permissions. The Active Directory principals database should be synchronized with the ADPS principals database. This synchronization is performed automatically before the cluster is kerberized. Every further change to the Active Directory principals database is synchronized with the ADPS database automatically.

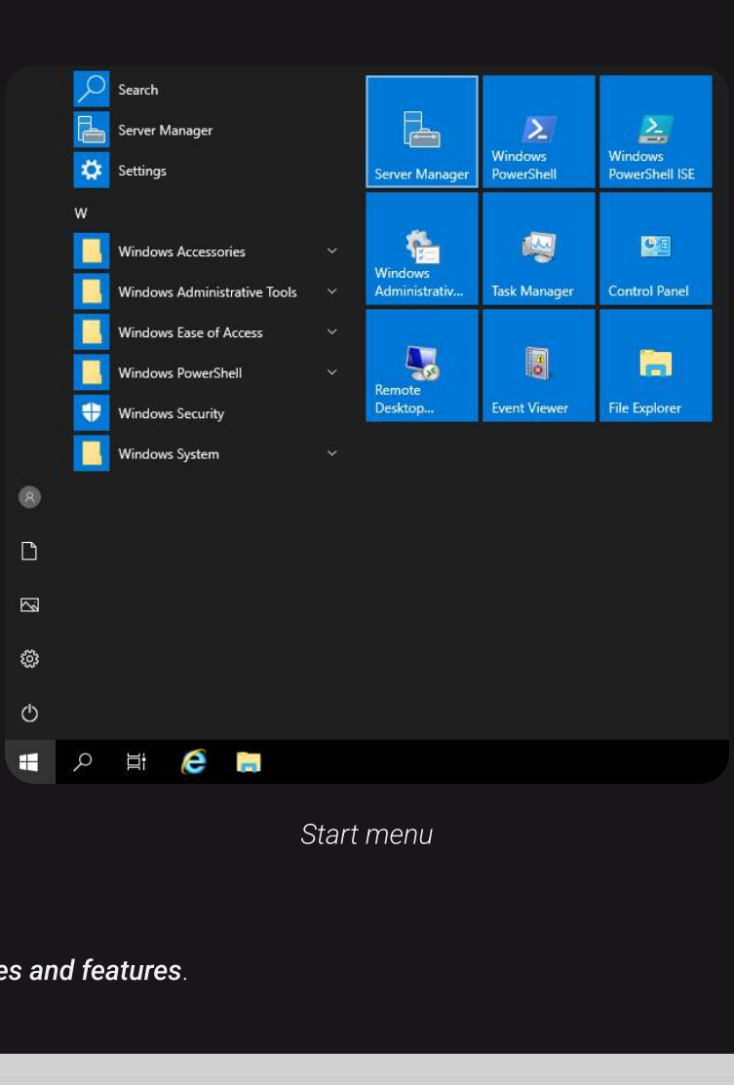
In this case, while working with an ADH cluster, all principals receive TGT from KDC AD.

Install AD Certification Service

Mikhail Serov

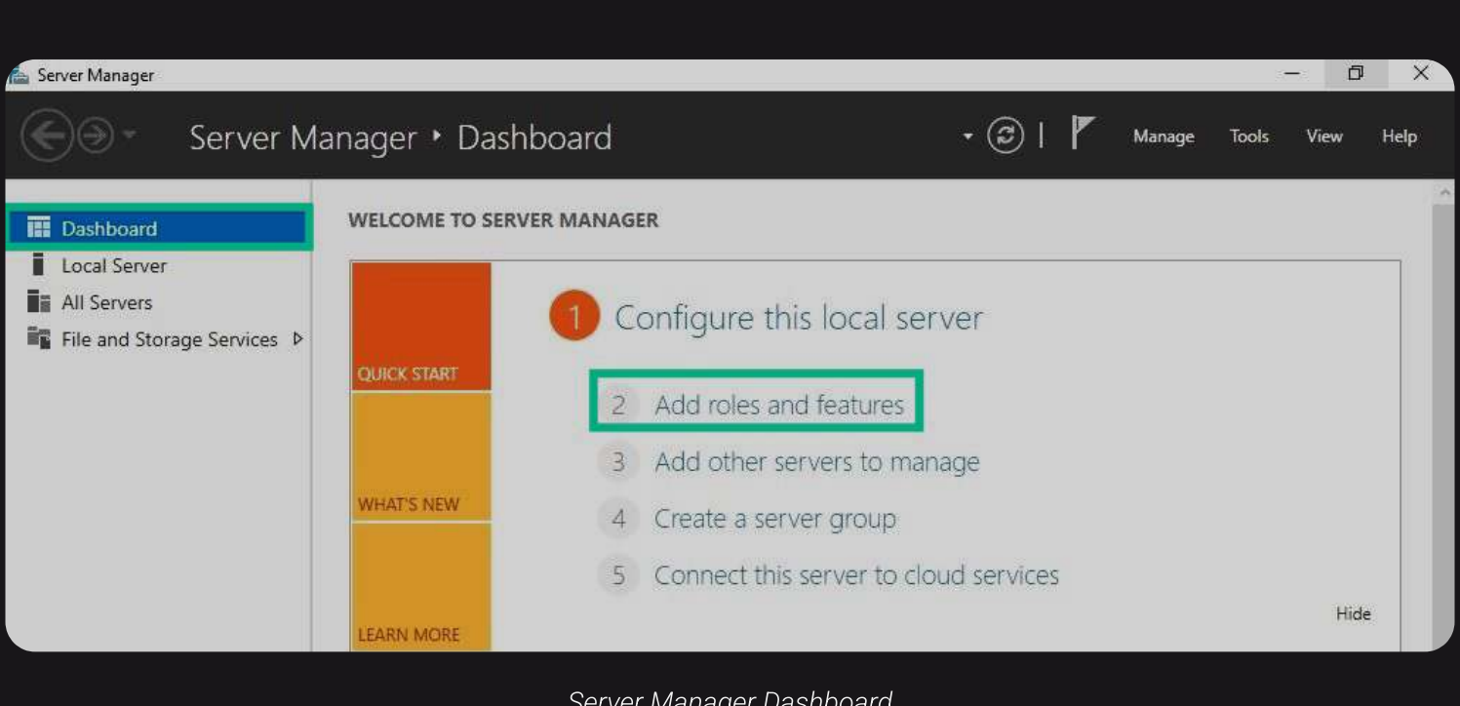
To Table of Contents

1. Run the Server Manager utility by clicking ***Start > Server Manager***.


Search
Server Manager
Settings
W
Windows Accessories
Windows Administrative Tools
Windows Ease of Access
Windows PowerShell
Windows Security
Windows System
Server Manager
Windows PowerShell
Windows PowerShell ISE
Windows Administrativ...
Task Manager
Control Panel
Remote Desktop...
Event Viewer
File Explorer

Start menu

2. When Server Manager opens, go to ***Dashboard > Add roles and features***.


Server Manager
Server Manager ‣ Dashboard
Manage
Tools
View
Help
Dashboard
Local Server
All Servers
File and Storage Services
QUICK START
WHAT'S NEW
LEARN MORE
Hide

WELCOME TO SERVER MANAGER

1 Configure this local server

2 Add roles and features

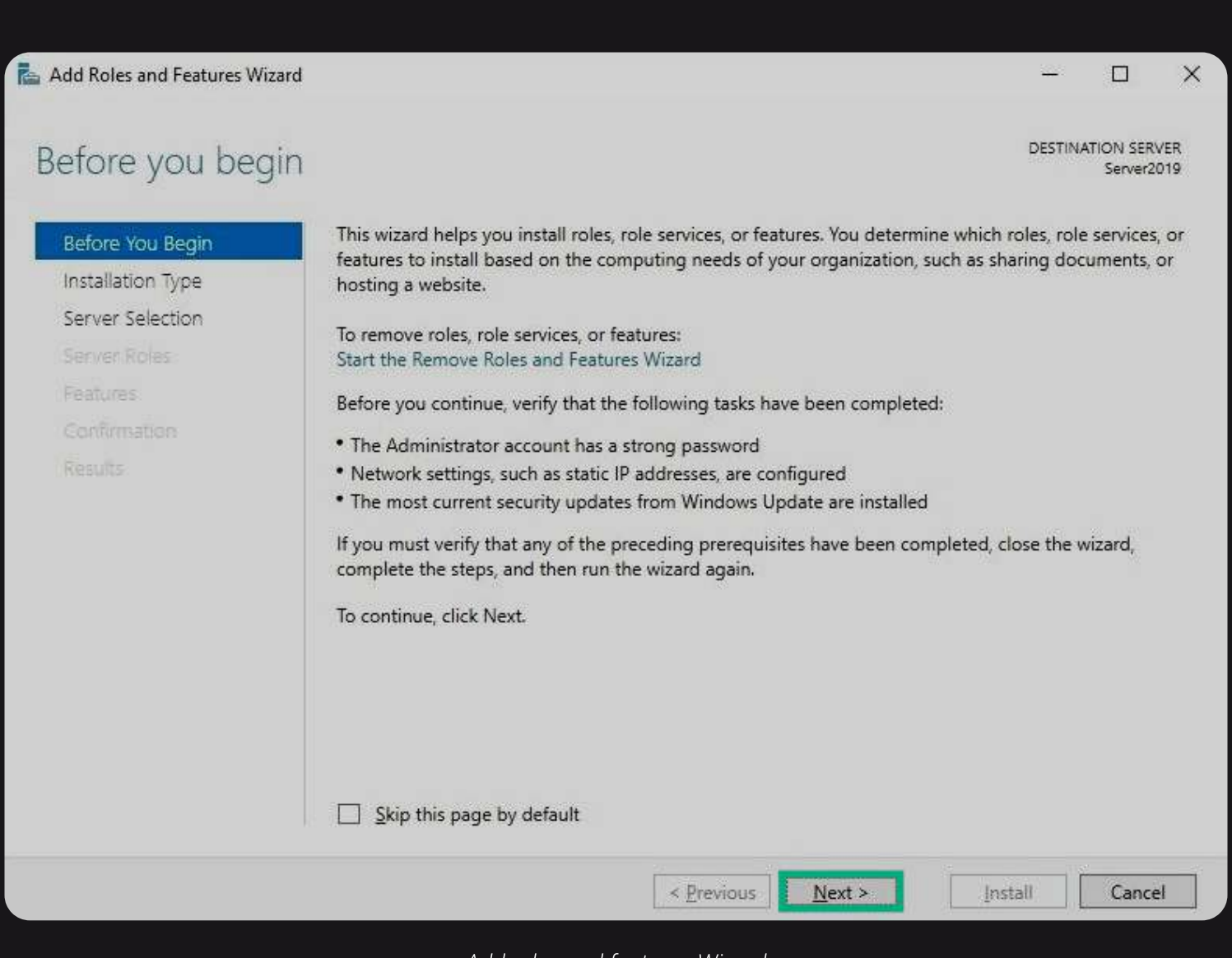
3 Add other servers to manage

4 Create a server group

5 Connect this server to cloud services

Server Manager Dashboard

3. Click ***Next***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
< Previous
Next >
Install
Cancel

Before you begin

DESTINATION SERVER
Server2019

This wizard helps you install roles, role services, or features. You determine which roles, role services, or features to install based on the computing needs of your organization, such as sharing documents, or hosting a website.

To remove roles, role services, or features:
Start the Remove Roles and Features Wizard

Before you continue, verify that the following tasks have been completed:

- The Administrator account has a strong password
- Network settings, such as static IP addresses, are configured
- The most current security updates from Windows Update are installed

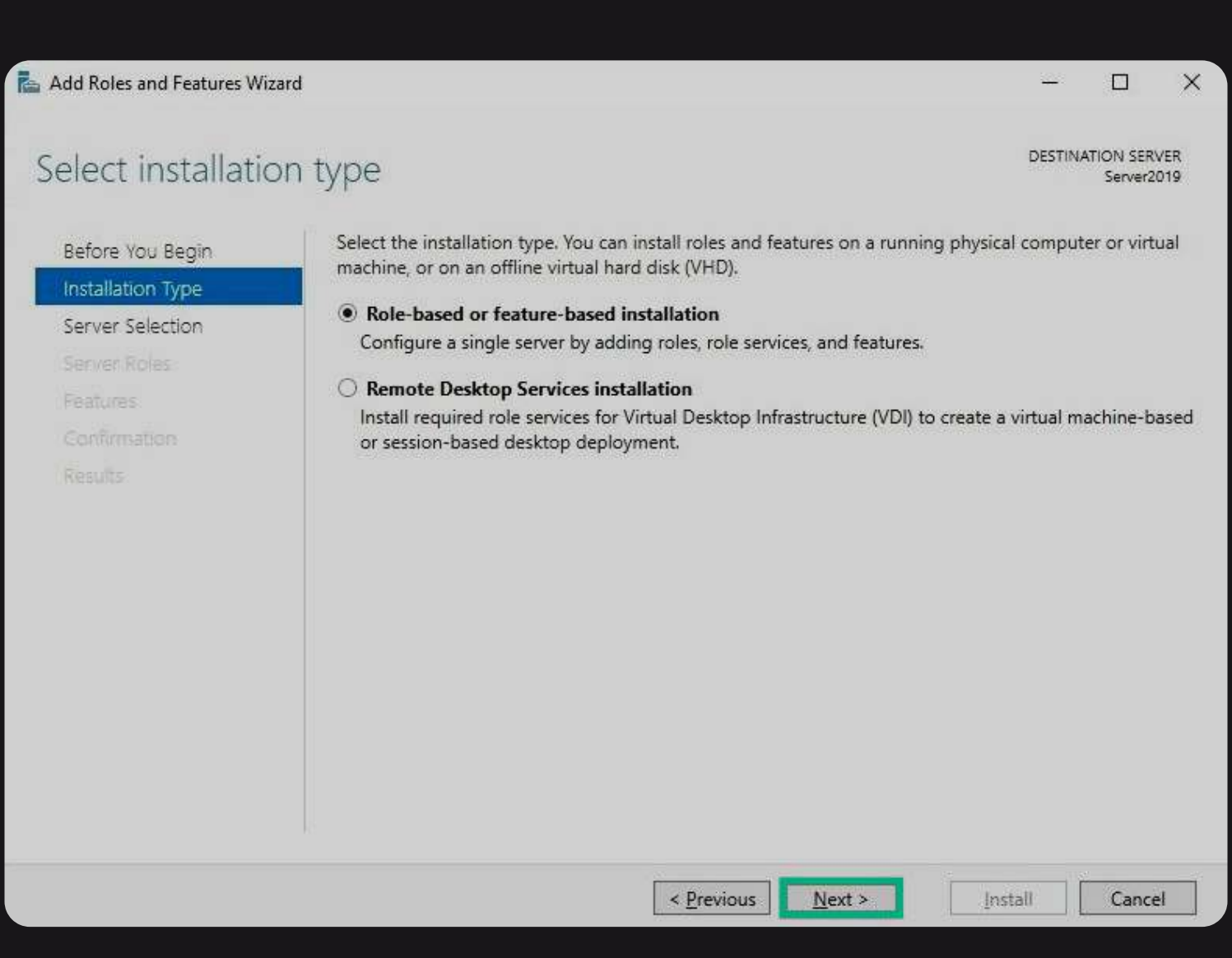
If you must verify that any of the preceding prerequisites have been completed, close the wizard, complete the steps, and then run the wizard again.

To continue, click Next.

☐ Skip this page by default

Add roles and features Wizard

4. Select ***Role-based or feature-based installation*** and click ***Next***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
< Previous
Next >
Install
Cancel

Select installation type

DESTINATION SERVER
Server2019

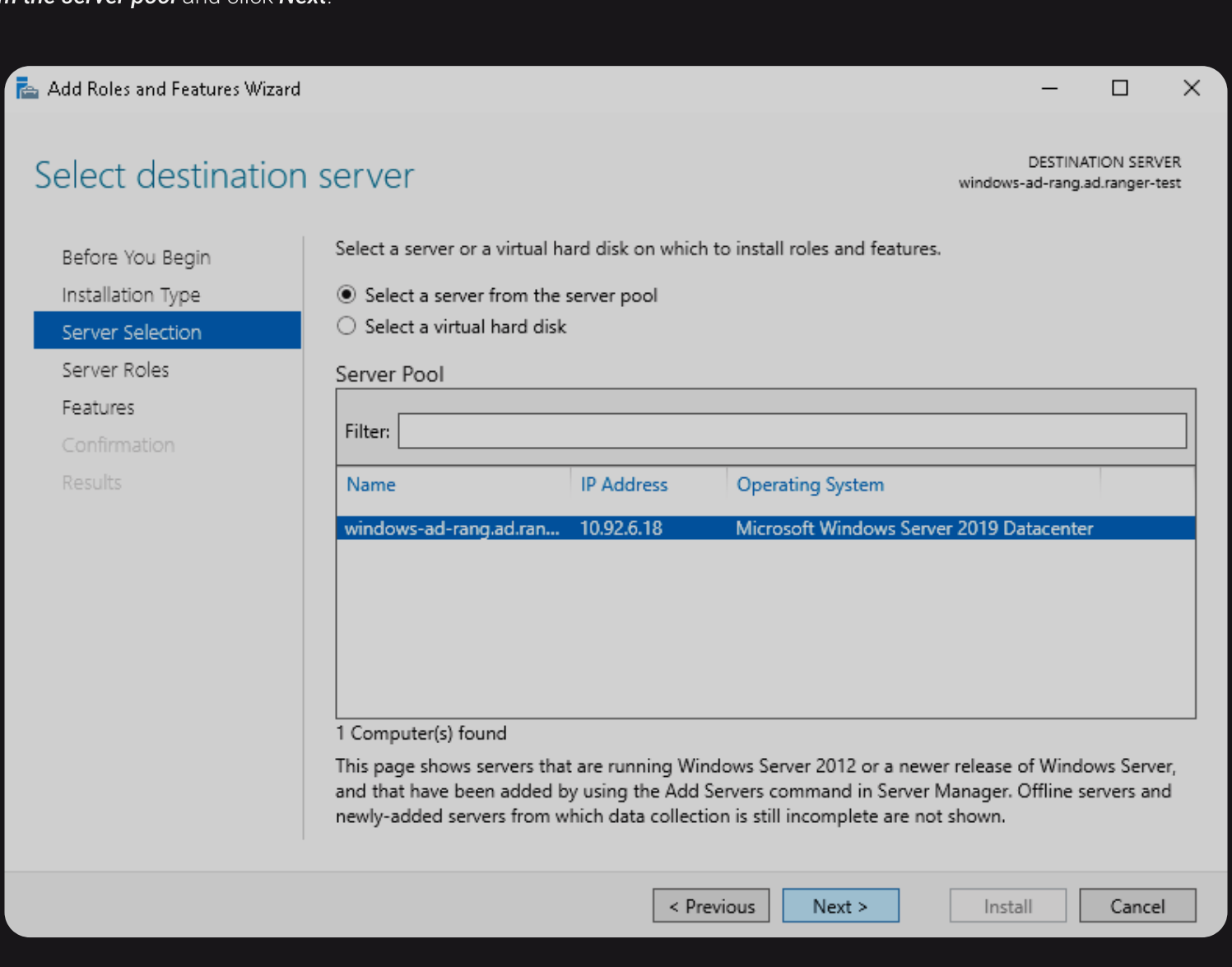
Select the installation type. You can install roles and features on a running physical computer or virtual machine, or on an offline virtual hard disk (VHD).

● Role-based or feature-based installation
Configure a single server by adding roles, role services, and features.

○ Remote Desktop Services installation
Install required role services for Virtual Desktop Infrastructure (VDI) to create a virtual machine-based or session-based desktop deployment.

Installation type

5. Choose ***Select a server from the server pool*** and click ***Next***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
Confirmation
Results
< Previous
Next >
Install
Cancel

Select destination server

DESTINATION SERVER
windows-ad-rang.ad.ranger-test

Select a server or a virtual hard disk on which to install roles and features.

● Select a server from the server pool
○ Select a virtual hard disk

Server Pool

Filter:

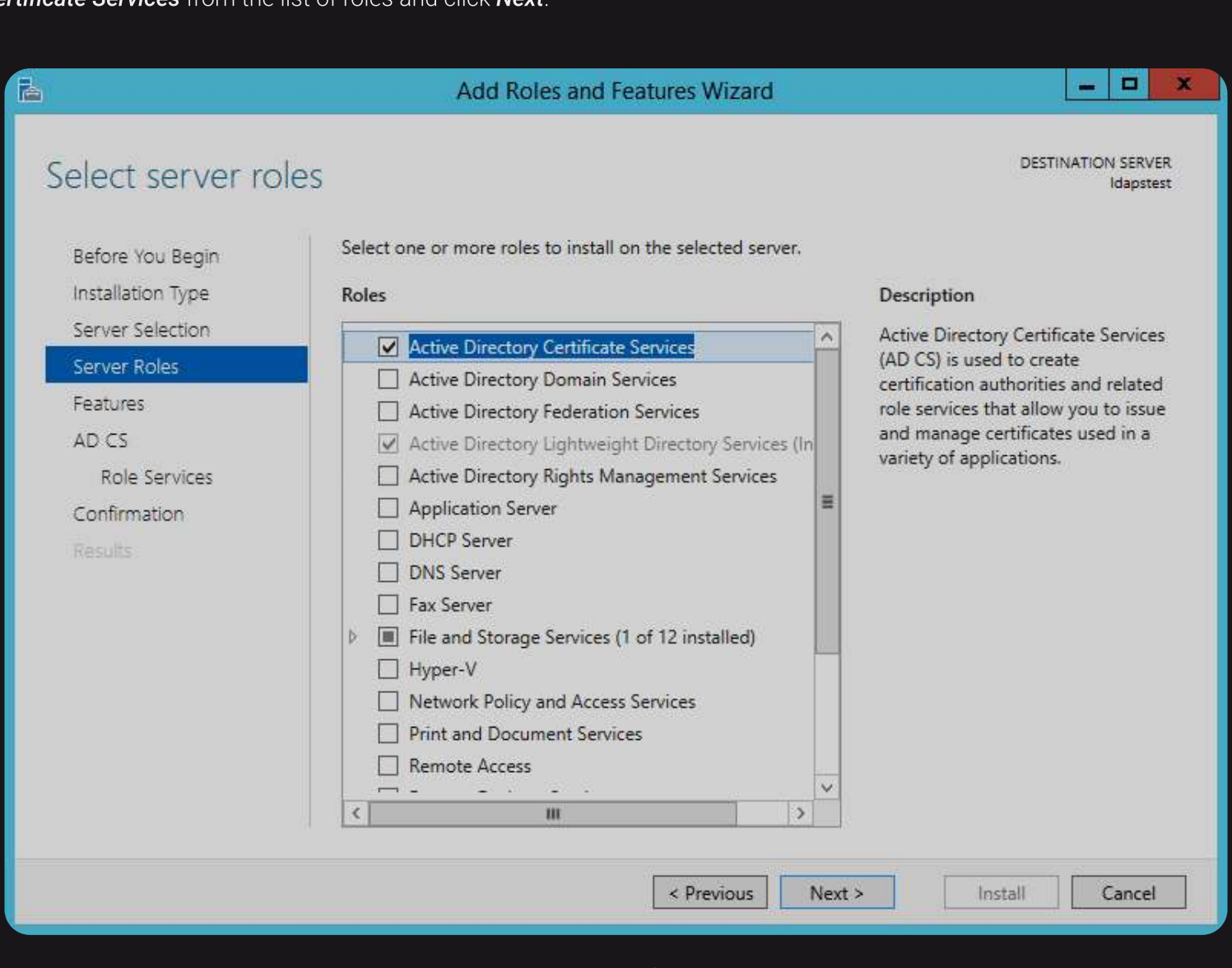
Name	IP Address	Operating System
windows-ad-rang.ad.ran...	10.92.6.18	Microsoft Windows Server 2019 Datacenter

1 Computer(s) found

This page shows servers that are running Windows Server 2012 or a newer release of Windows Server, and that have been added by using the Add Servers command in Server Manager. Offline servers and newly-added servers from which data collection is still incomplete are not shown.

Server Selection

6. Choose ***Active Directory Certificate Services*** from the list of roles and click ***Next***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD CS
Role Services
Confirmation
Results
< Previous
Next >
Install
Cancel

Select server roles

DESTINATION SERVER
ldapstest

Select one or more roles to install on the selected server.

Roles

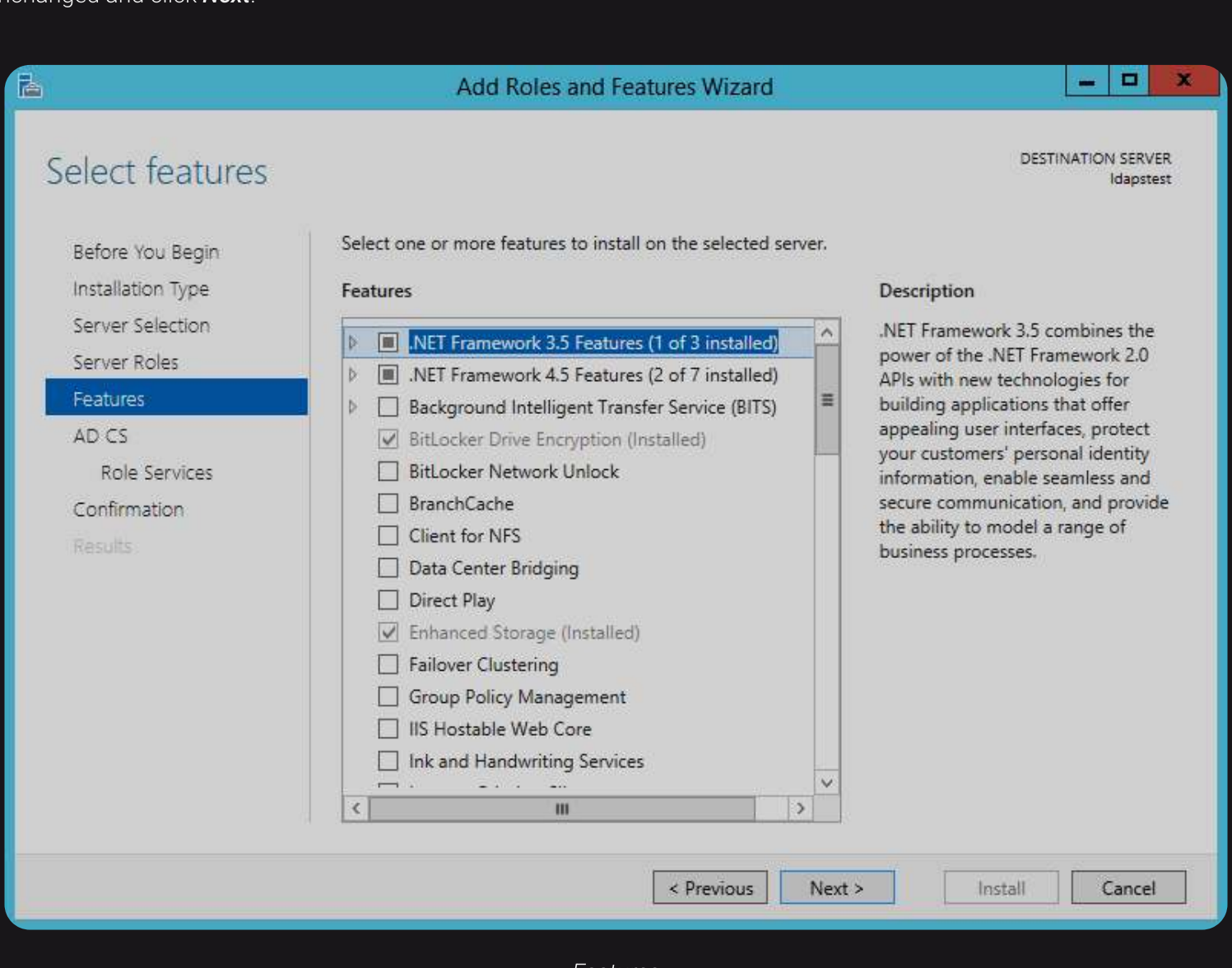
☑ Active Directory Certificate Services
☐ Active Directory Domain Services
☐ Active Directory Federation Services
☑ Active Directory Lightweight Directory Services (In
☐ Active Directory Rights Management Services
☐ Application Server
☐ DHCP Server
☐ DNS Server
☐ Fax Server
▣ File and Storage Services (1 of 12 installed)
☐ Hyper-V
☐ Network Policy and Access Services
☐ Print and Document Services
☐ Remote Access

Description

Active Directory Certificate Services (AD CS) is used to create certification authorities and related role services that allow you to issue and manage certificates used in a variety of applications.

Server Roles

7. Leave the list of features unchanged and click ***Next***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD CS
Role Services
Confirmation
Results
< Previous
Next >
Install
Cancel

Select features

DESTINATION SERVER
ldapstest

Select one or more features to install on the selected server.

Features

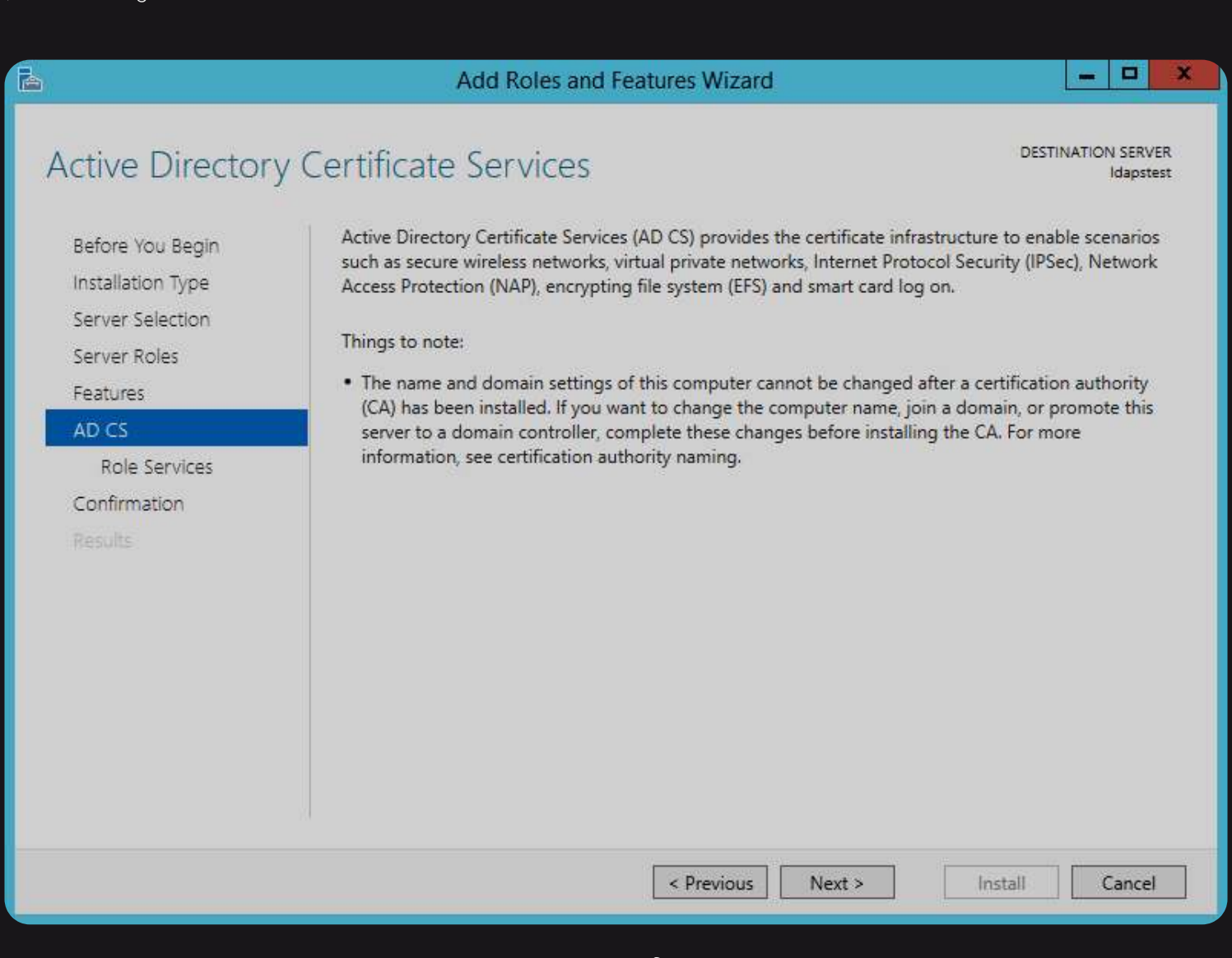
▣ .NET Framework 3.5 Features (1 of 3 installed)
▣ .NET Framework 4.5 Features (2 of 7 installed)
☐ Background Intelligent Transfer Service (BITS)
☑ BitLocker Drive Encryption (Installed)
☐ BitLocker Network Unlock
☐ BranchCache
☐ Client for NFS
☐ Data Center Bridging
☐ Direct Play
☑ Enhanced Storage (Installed)
☐ Failover Clustering
☐ Group Policy Management
☐ IIS Hostable Web Core
☐ Ink and Handwriting Services

Description

.NET Framework 3.5 combines the power of the .NET Framework 2.0 APIs with new technologies for building applications that offer appealing user interfaces, protect your customers' personal identity information, enable seamless and secure communication, and provide the ability to model a range of business processes.

Features

8. Click ***Next*** and on next step, click ***Next*** again.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD CS
Role Services
Confirmation
Results
< Previous
Next >
Install
Cancel

Active Directory Certificate Services

DESTINATION SERVER
ldapstest

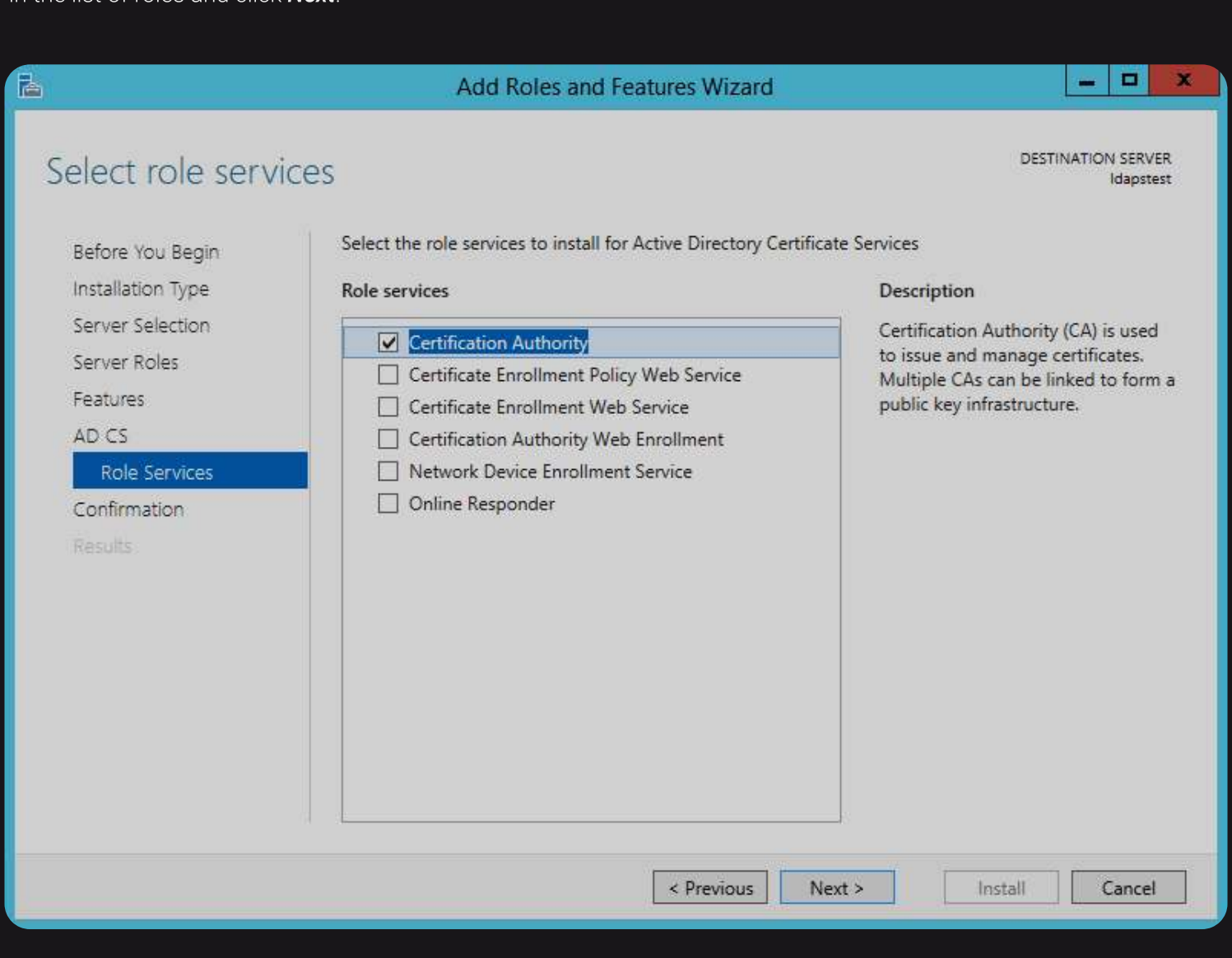
Active Directory Certificate Services (AD CS) provides the certificate infrastructure to enable scenarios such as secure wireless networks, virtual private networks, Internet Protocol Security (IPSec), Network Access Protection (NAP), encrypting file system (EFS) and smart card log on.

Things to note:

- The name and domain settings of this computer cannot be changed after a certification authority (CA) has been installed. If you want to change the computer name, join a domain, or promote this server to a domain controller, complete these changes before installing the CA. For more information, see certification authority naming.

Active Directory Certificate Services

9. Check ***Certificate Authority*** in the list of roles and click ***Next***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD CS
Role Services
Confirmation
Results
< Previous
Next >
Install
Cancel

Select role services

DESTINATION SERVER
ldapstest

Select the role services to install for Active Directory Certificate Services

Role services

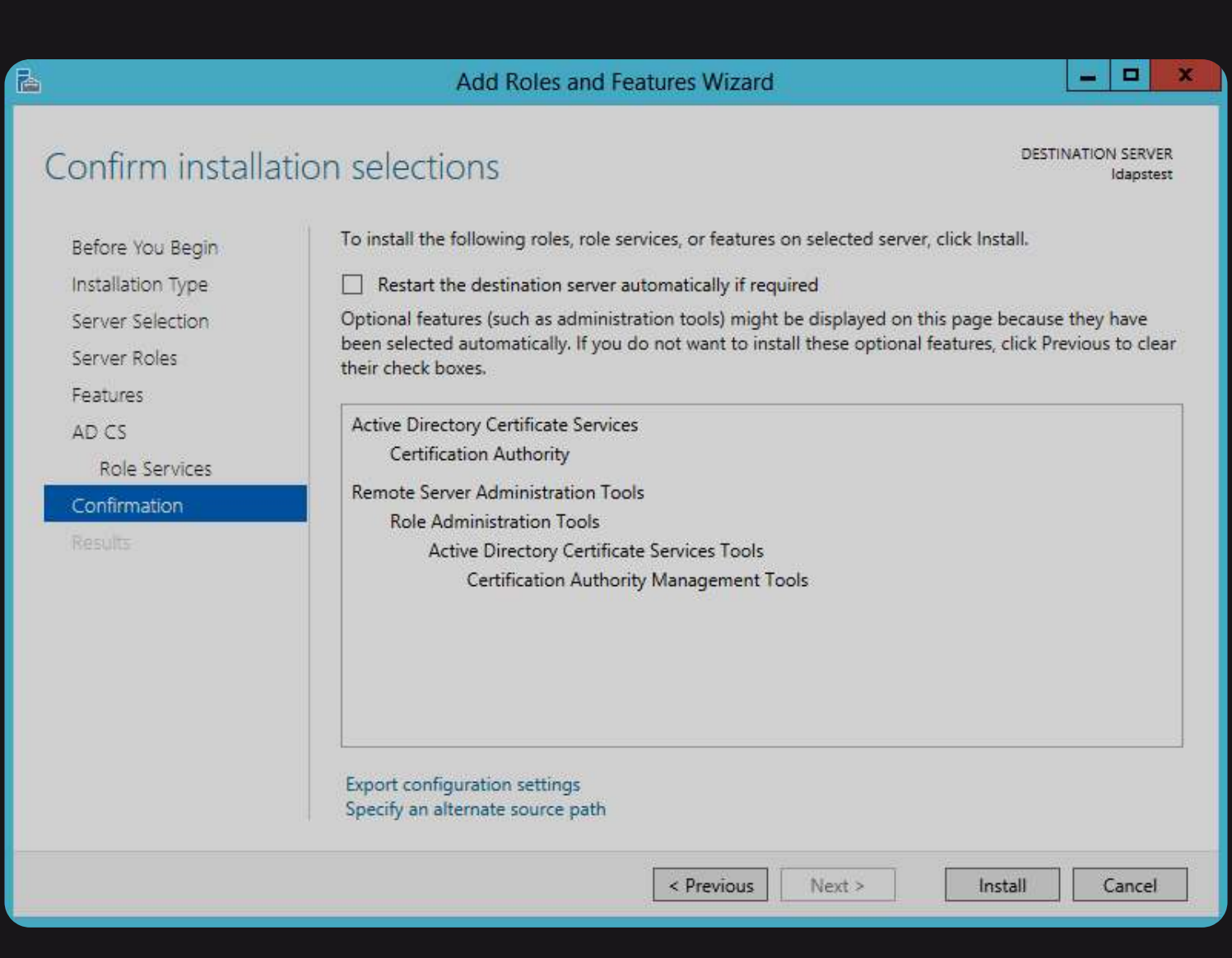
☑ Certification Authority
☐ Certificate Enrollment Policy Web Service
☐ Certificate Enrollment Web Service
☐ Certification Authority Web Enrollment
☐ Network Device Enrollment Service
☐ Online Responder

Description

Certification Authority (CA) is used to issue and manage certificates. Multiple CAs can be linked to form a public key infrastructure.

Role Services

10. Click ***Install*** to confirm the installation.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD CS
Role Services
Confirmation
Results
< Previous
Next >
Install
Cancel

Confirm installation selections

DESTINATION SERVER
ldapstest

To install the following roles, role services, or features on selected server, click Install.

☐ Restart the destination server automatically if required

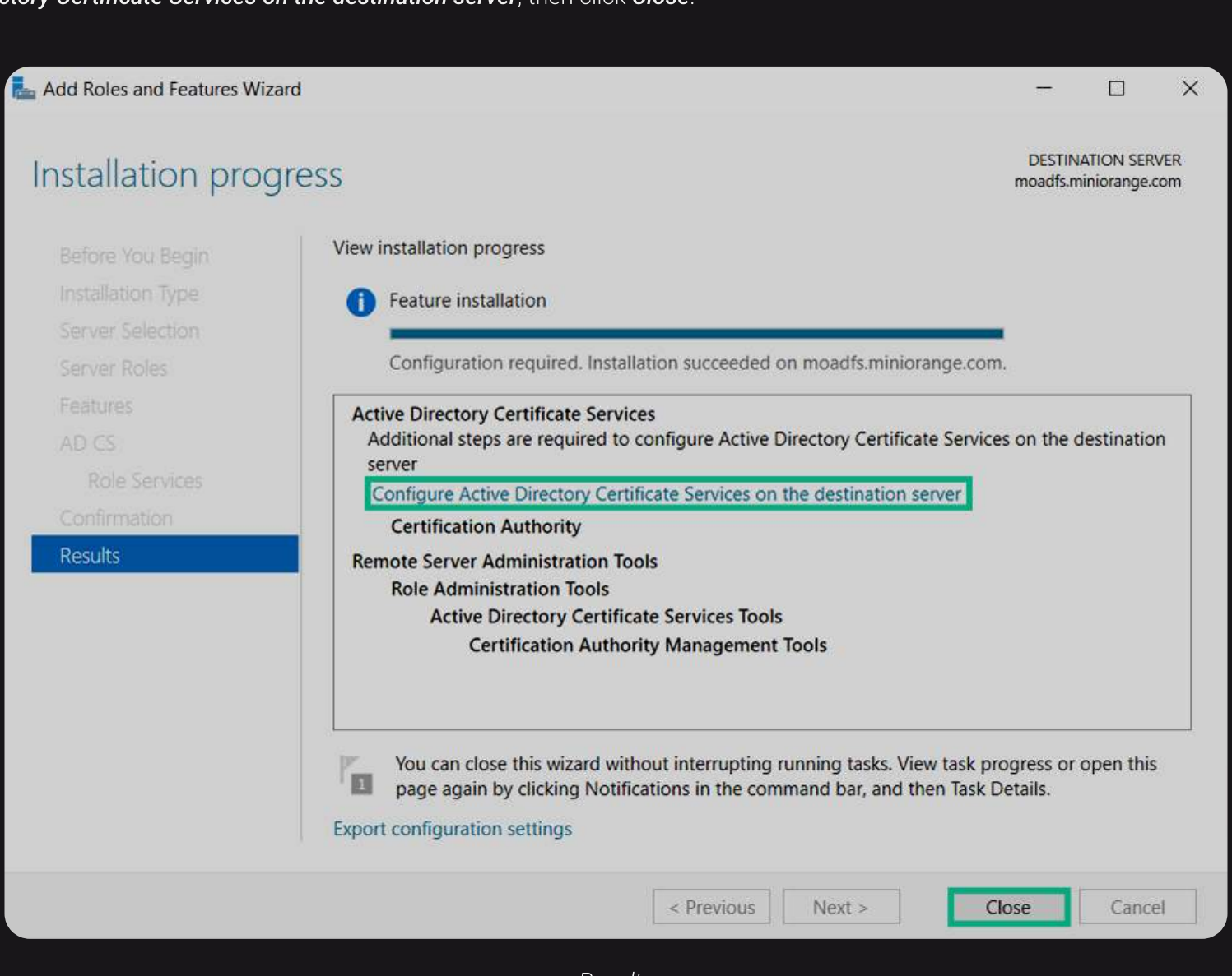
Optional features (such as administration tools) might be displayed on this page because they have been selected automatically. If you do not want to install these optional features, click Previous to clear their check boxes.

Active Directory Certificate Services
  Certification Authority
Remote Server Administration Tools
  Role Administration Tools
    Active Directory Certificate Services Tools
      Certification Authority Management Tools

Export configuration settings
Specify an alternate source path

Confirmation

11. Click ***Configure Active Directory Certificate Services on the destination server***, then click ***Close***.


Add Roles and Features Wizard
Before You Begin
Installation Type
Server Selection
Server Roles
Features
AD CS
Role Services
Confirmation
Results
< Previous
Next >
Close
Cancel

Installation progress

DESTINATION SERVER
moadfs.miniorange.com

View installation progress

Feature installation

Configuration required. Installation succeeded on moadfs.miniorange.com.

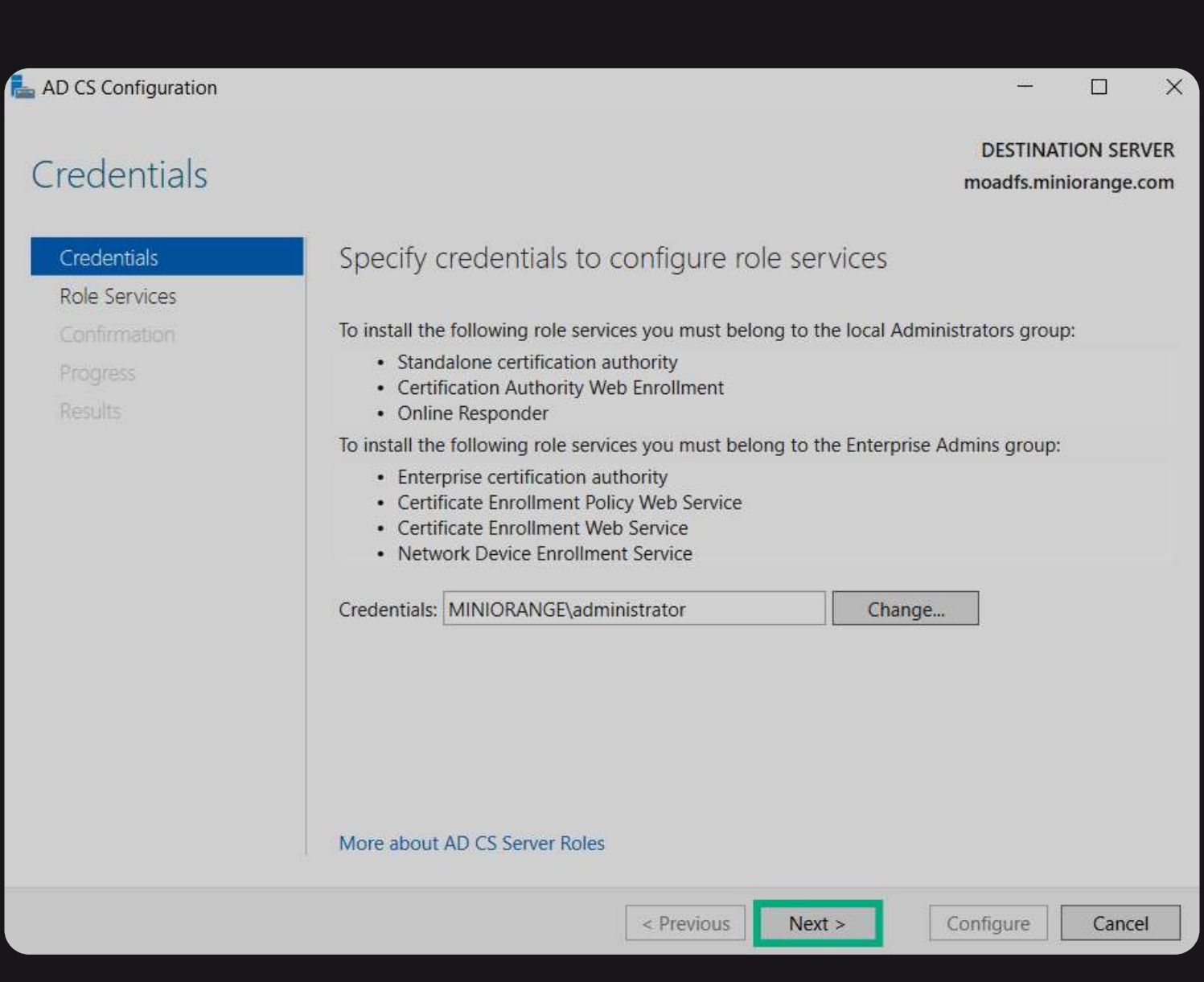
Active Directory Certificate Services
Additional steps are required to configure Active Directory Certificate Services on the destination server
Configure Active Directory Certificate Services on the destination server
Certification Authority
Remote Server Administration Tools
Role Administration Tools
Active Directory Certificate Services Tools
Certification Authority Management Tools

You can close this wizard without interrupting running tasks. View task progress or open this page again by clicking Notifications in the command bar, and then Task Details.

Export configuration settings

Results

12. Click ***Next***.


AD CS Configuration
Credentials
Role Services
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel
Change...

Credentials

DESTINATION SERVER
moadfs.miniorange.com

Specify credentials to configure role services

To install the following role services you must belong to the local Administrators group:

- Standalone certification authority
- Certification Authority Web Enrollment
- Online Responder

To install the following role services you must belong to the Enterprise Admins group:

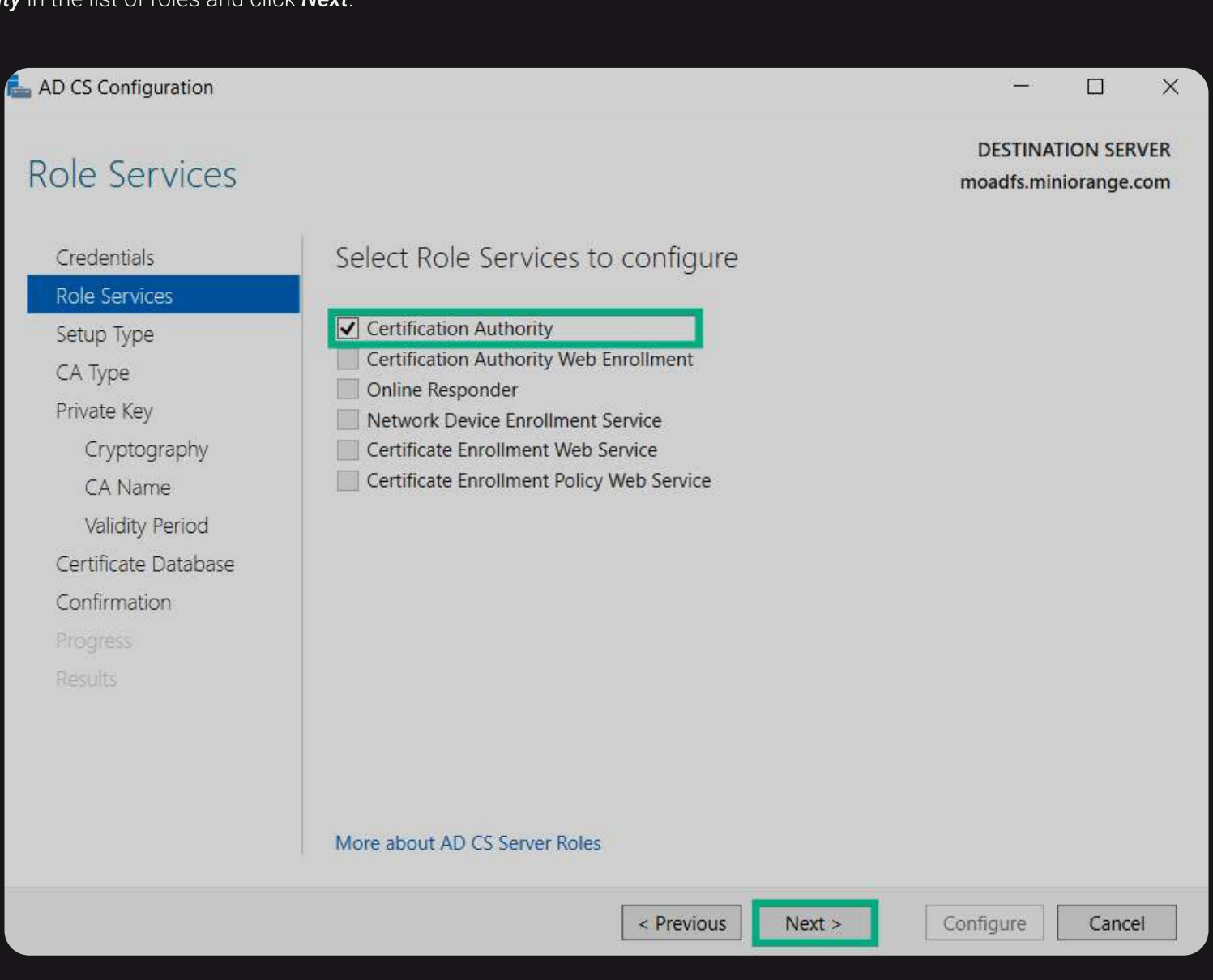
- Enterprise certification authority
- Certificate Enrollment Policy Web Service
- Certificate Enrollment Web Service
- Network Device Enrollment Service

Credentials: MINIORANGE\administrator

More about AD CS Server Roles

Credentials

13. Select ***Certification Authority*** in the list of roles and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

Role Services

DESTINATION SERVER
moadfs.miniorange.com

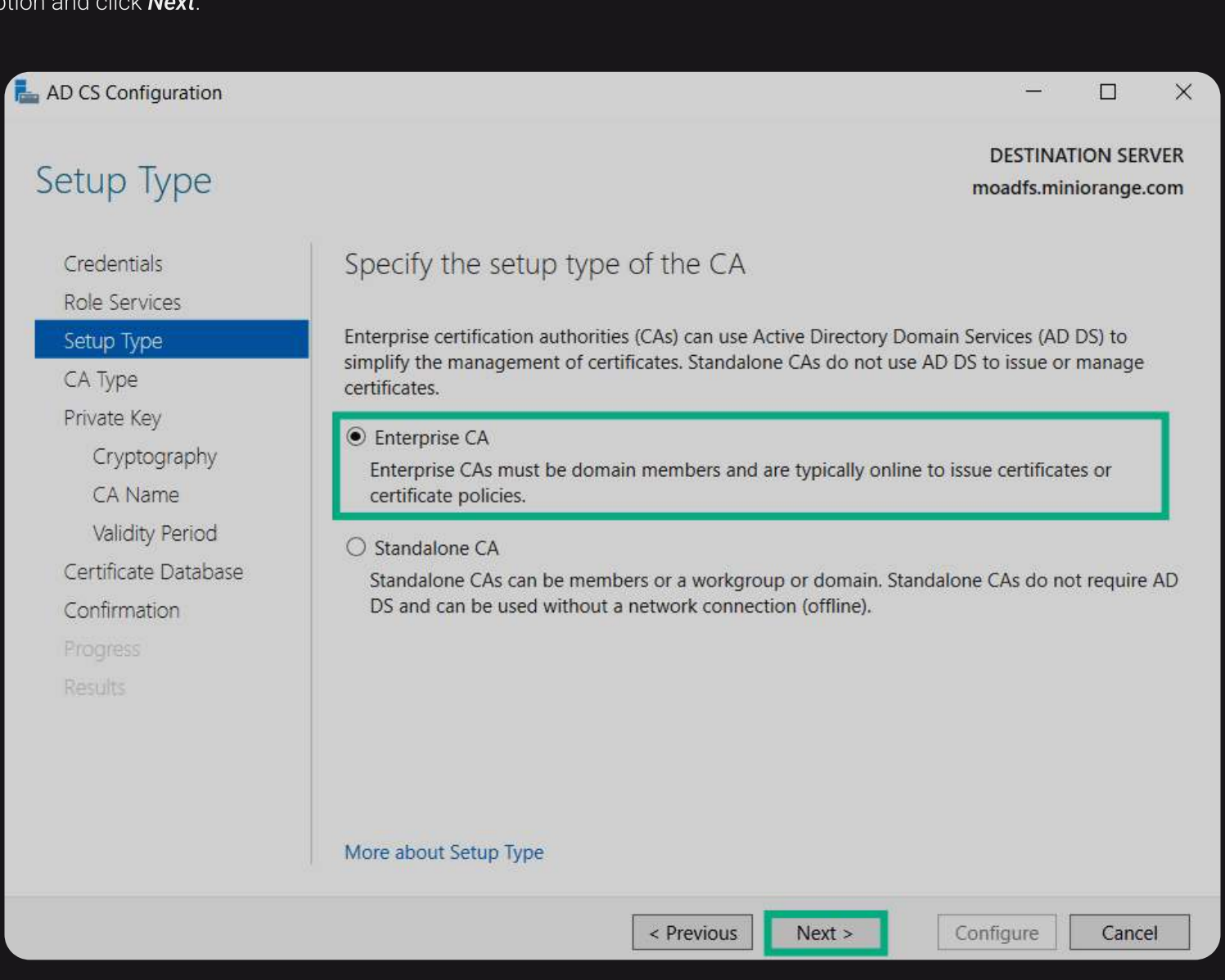
Select Role Services to configure

☑ Certification Authority
☐ Certification Authority Web Enrollment
☐ Online Responder
☐ Network Device Enrollment Service
☐ Certificate Enrollment Web Service
☐ Certificate Enrollment Policy Web Service

More about AD CS Server Roles

Role Services

14. Select the ***Enterprise CA*** option and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

Setup Type

DESTINATION SERVER
moadfs.miniorange.com

Specify the setup type of the CA

Enterprise certification authorities (CAs) can use Active Directory Domain Services (AD DS) to simplify the management of certificates. Standalone CAs do not use AD DS to issue or manage certificates.

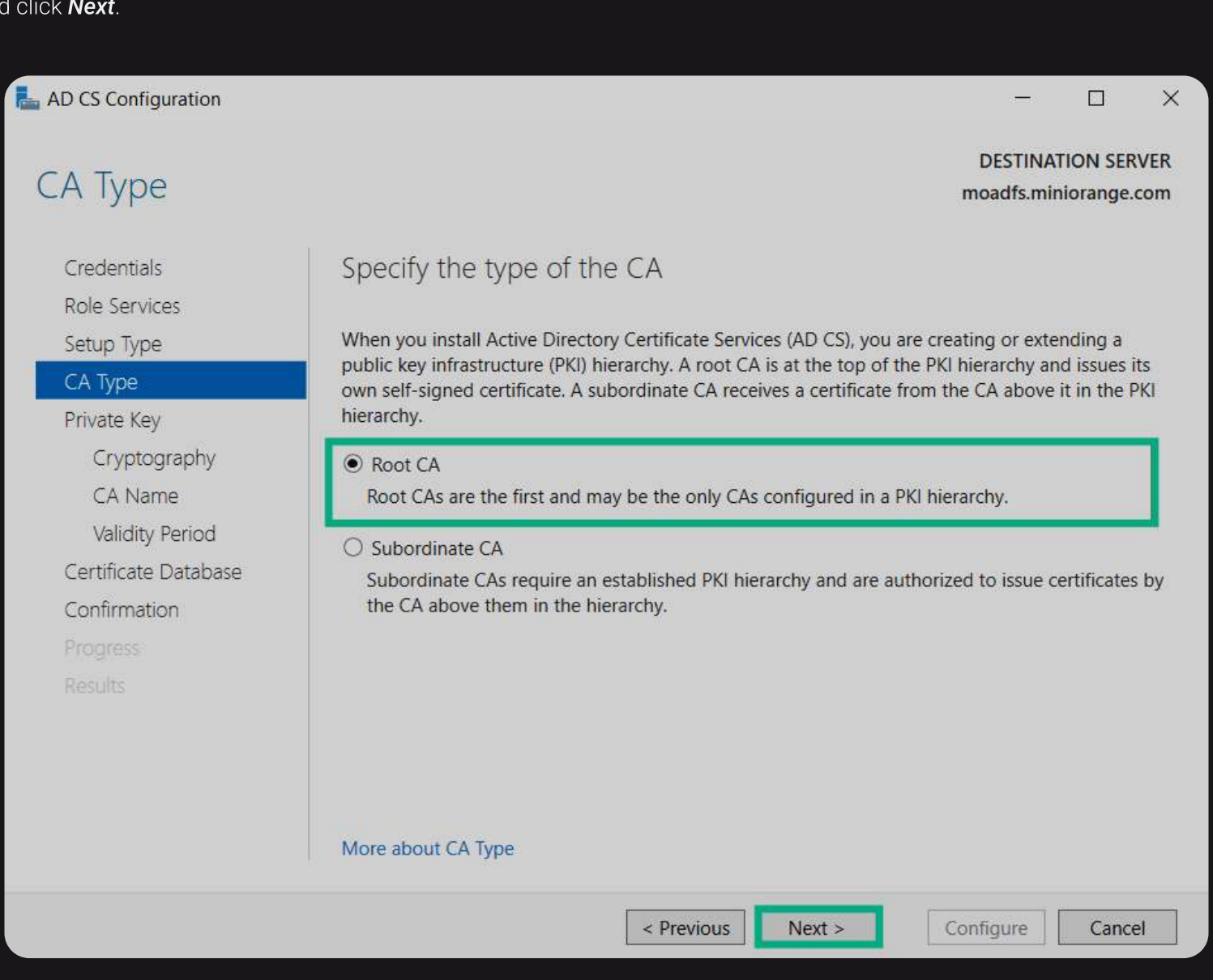
● Enterprise CA
Enterprise CAs must be domain members and are typically online to issue certificates or certificate policies.

○ Standalone CA
Standalone CAs can be members or a workgroup or domain. Standalone CAs do not require AD DS and can be used without a network connection (offline).

More about Setup Type

Setup Type

15. Choose ***Root CA*** option and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

CA Type

DESTINATION SERVER
moadfs.miniorange.com

Specify the type of the CA

When you install Active Directory Certificate Services (AD CS), you are creating or extending a public key infrastructure (PKI) hierarchy. A root CA is at the top of the PKI hierarchy and issues its own self-signed certificate. A subordinate CA receives a certificate from the CA above it in the PKI hierarchy.

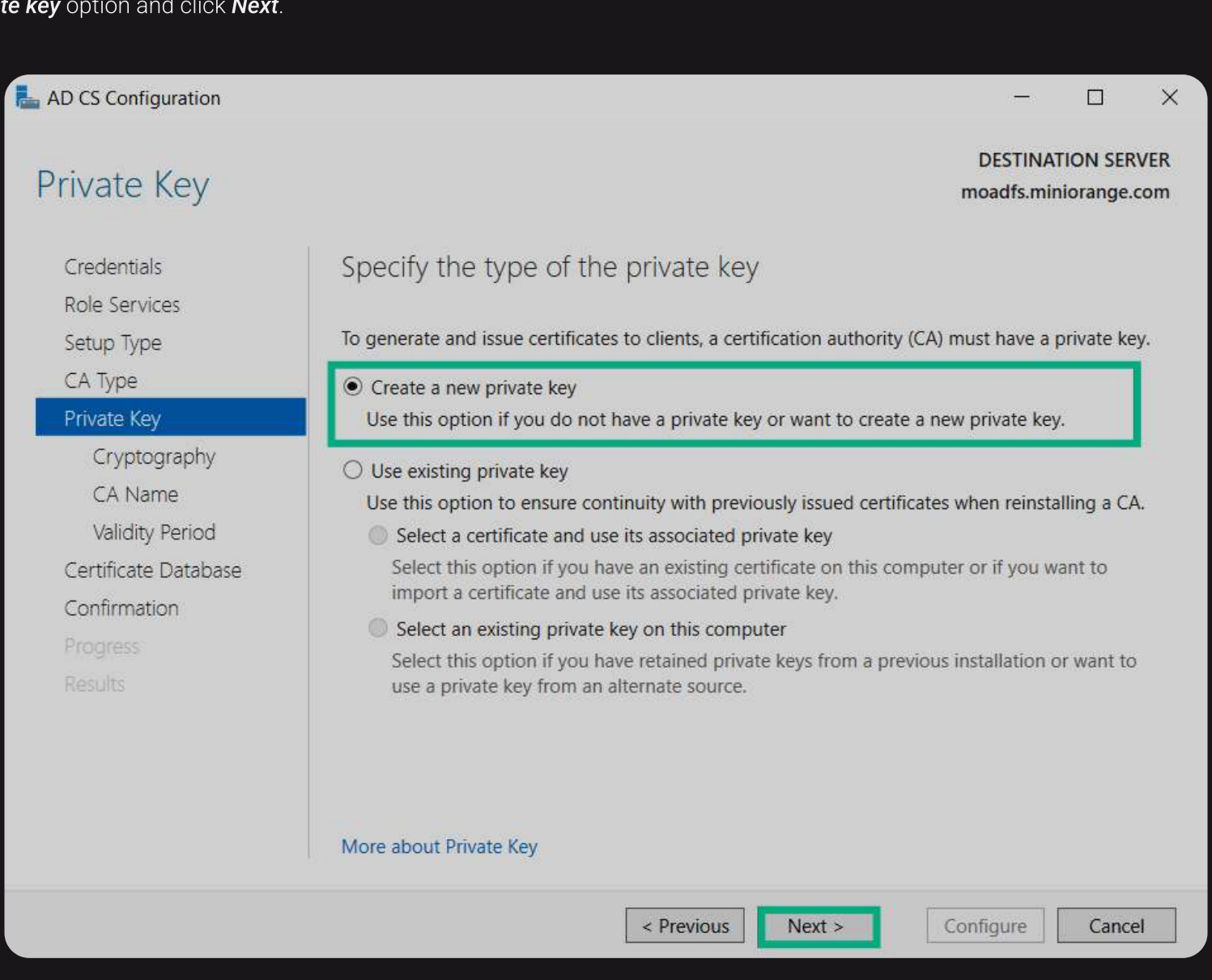
● Root CA
Root CAs are the first and may be the only CAs configured in a PKI hierarchy.

○ Subordinate CA
Subordinate CAs require an established PKI hierarchy and are authorized to issue certificates by the CA above them in the hierarchy.

More about CA Type

CA Type

16. Choose ***Create a new private key*** option and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

Private Key

DESTINATION SERVER
moadfs.miniorange.com

Specify the type of the private key

To generate and issue certificates to clients, a certification authority (CA) must have a private key.

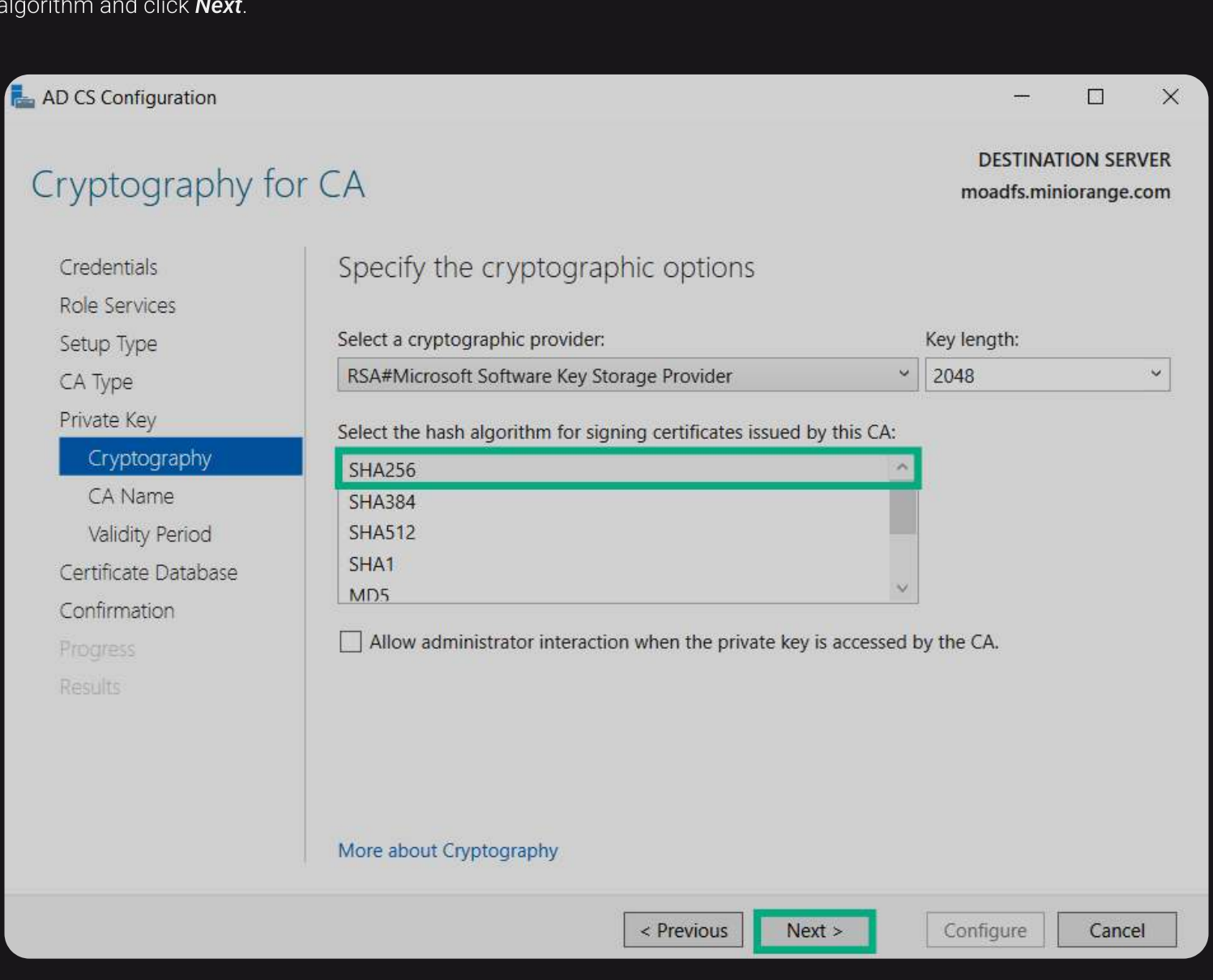
● Create a new private key
Use this option if you do not have a private key or want to create a new private key.

○ Use existing private key
Use this option to ensure continuity with previously issued certificates when reinstalling a CA.
  ○ Select a certificate and use its associated private key
  Select this option if you have an existing certificate on this computer or if you want to import a certificate and use its associated private key.
  ○ Select an existing private key on this computer
  Select this option if you have retained private keys from a previous installation or want to use a private key from an alternate source.

More about Private Key

Private Key

17. Choose the ***SHA256*** hash algorithm and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

Cryptography for CA

DESTINATION SERVER
moadfs.miniorange.com

Specify the cryptographic options

Select a cryptographic provider: RSA#Microsoft Software Key Storage Provider

Key length: 2048

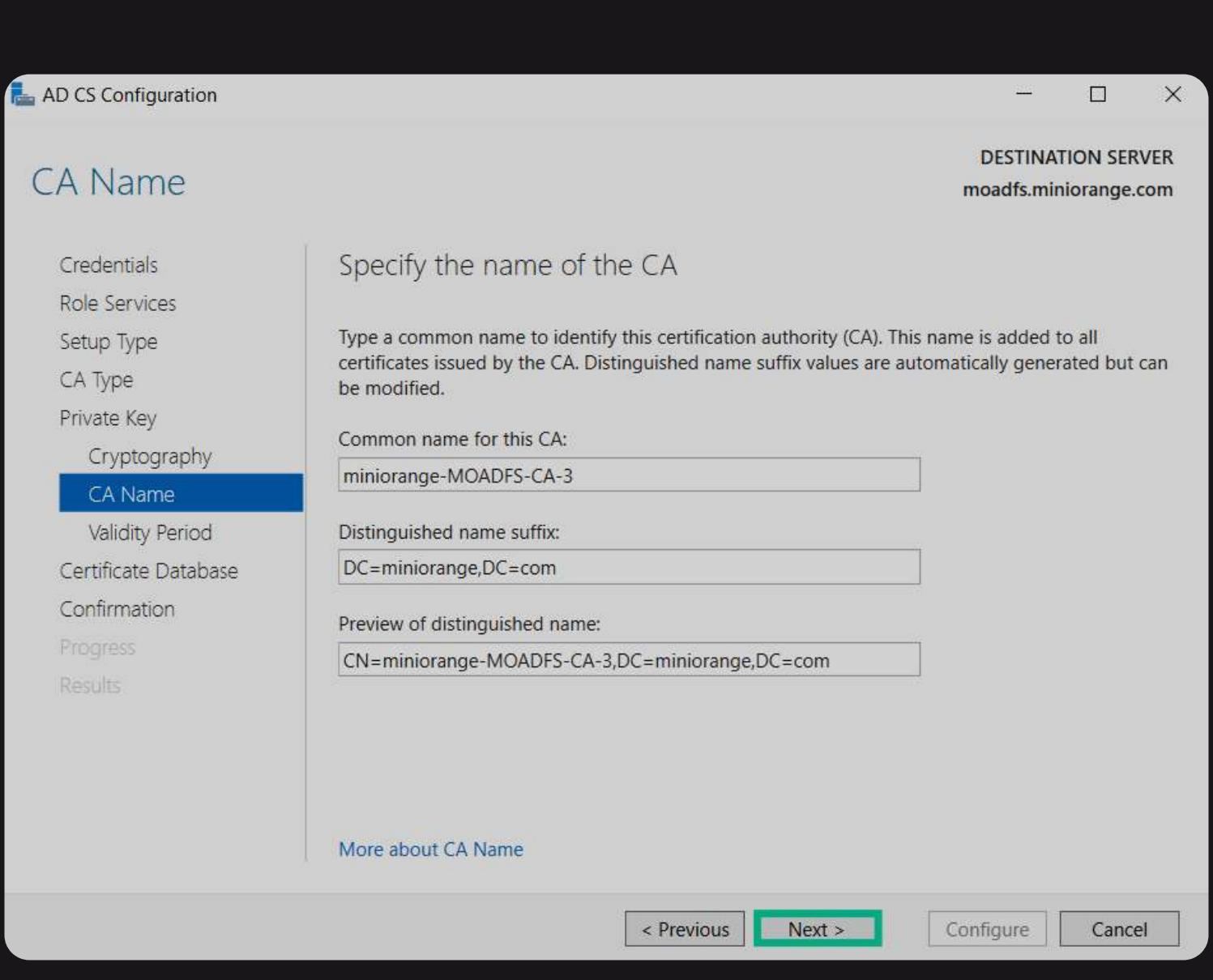
Select the hash algorithm for signing certificates issued by this CA:
SHA256
SHA384
SHA512
SHA1
MD5

☐ Allow administrator interaction when the private key is accessed by the CA.

More about Cryptography

Cryptography

18. Click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

CA Name

DESTINATION SERVER
moadfs.miniorange.com

Specify the name of the CA

Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.

Common name for this CA: miniorange-MOADFS-CA-3

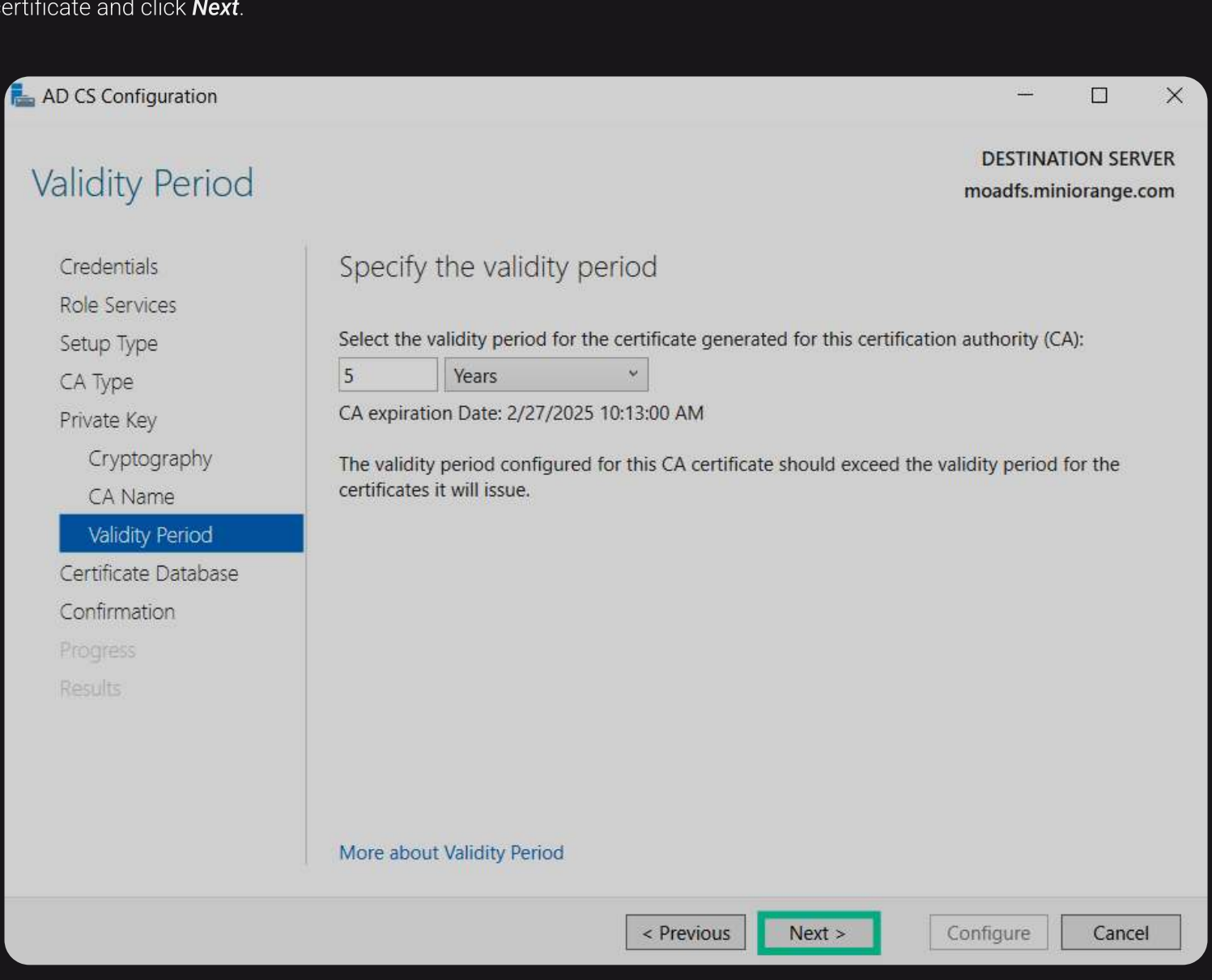
Distinguished name suffix: DC=miniorange,DC=com

Preview of distinguished name: CN=miniorange-MOADFS-CA-3,DC=miniorange,DC=com

More about CA Name

CA Name

19. Specify the validity of the certificate and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

Validity Period

DESTINATION SERVER
moadfs.miniorange.com

Specify the validity period

Select the validity period for the certificate generated for this certification authority (CA):
5 Years

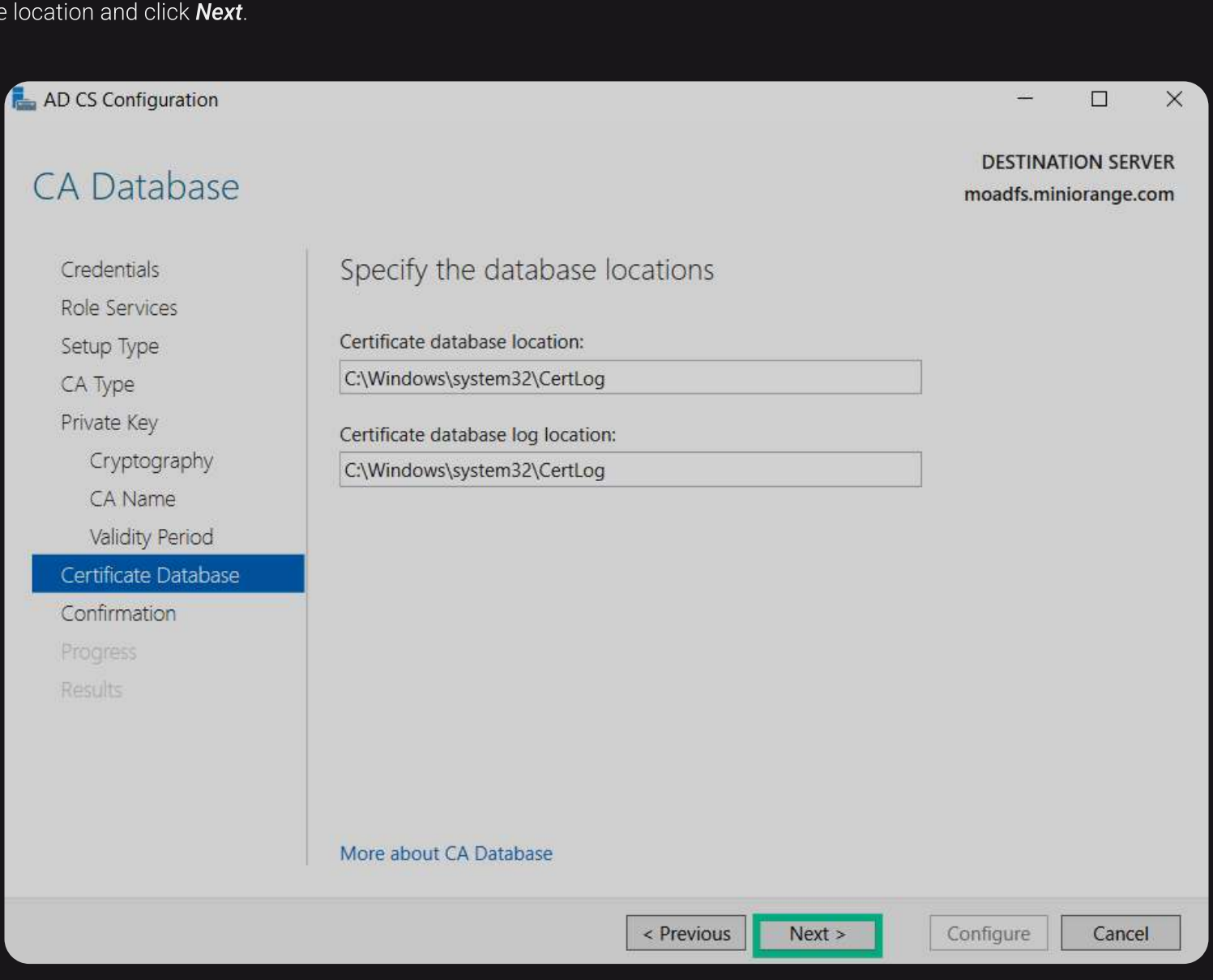
CA expiration Date: 2/27/2025 10:13:00 AM

The validity period configured for this CA certificate should exceed the validity period for the certificates it will issue.

More about Validity Period

Validity Period

20. Select the default database location and click ***Next***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

CA Database

DESTINATION SERVER
moadfs.miniorange.com

Specify the database locations

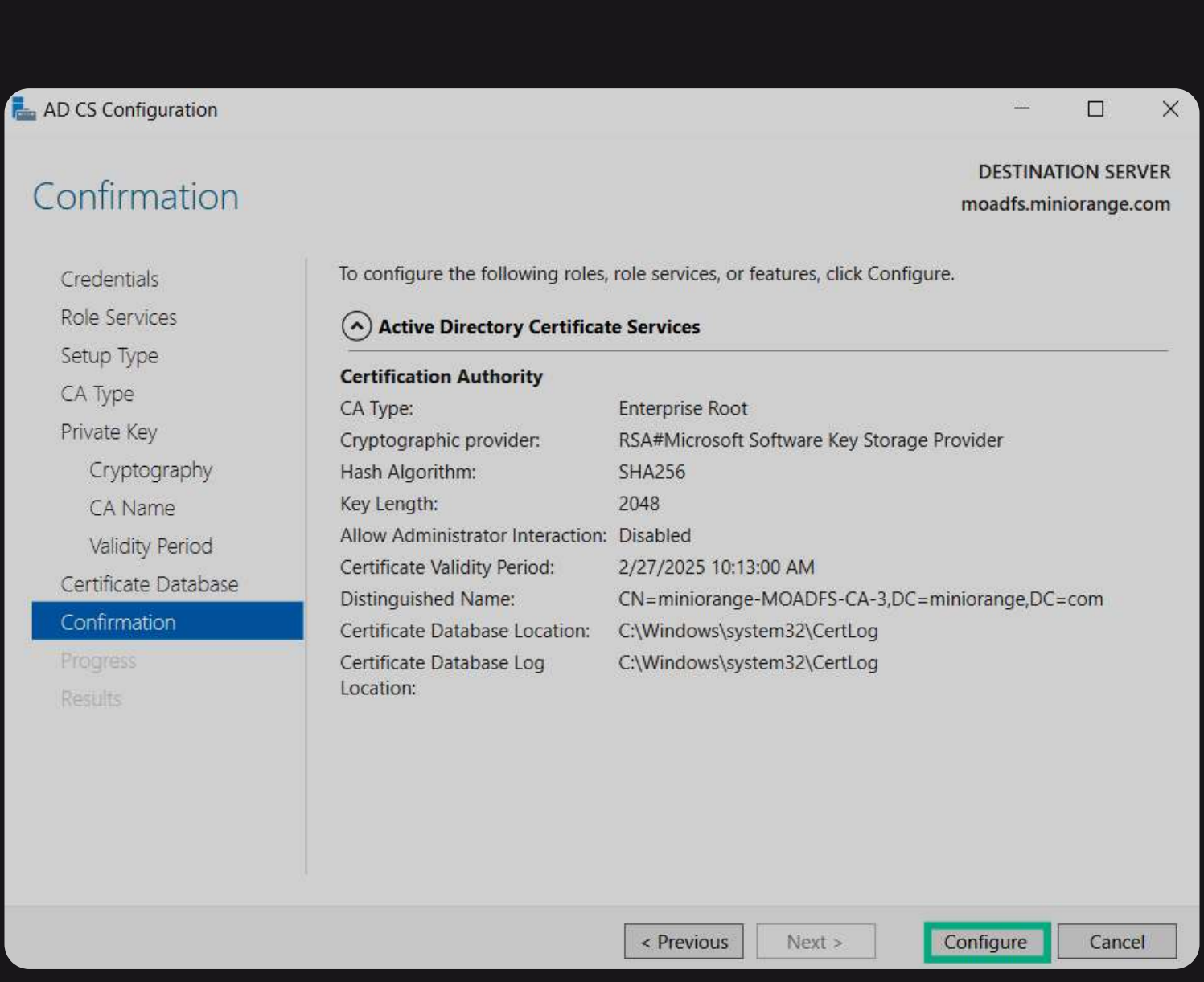
Certificate database location: C:\Windows\system32\CertLog

Certificate database log location: C:\Windows\system32\CertLog

More about CA Database

Certificate Database

21. Click ***Configure*** to confirm.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Configure
Cancel

Confirmation

DESTINATION SERVER
moadfs.miniorange.com

To configure the following roles, role services, or features, click Configure.

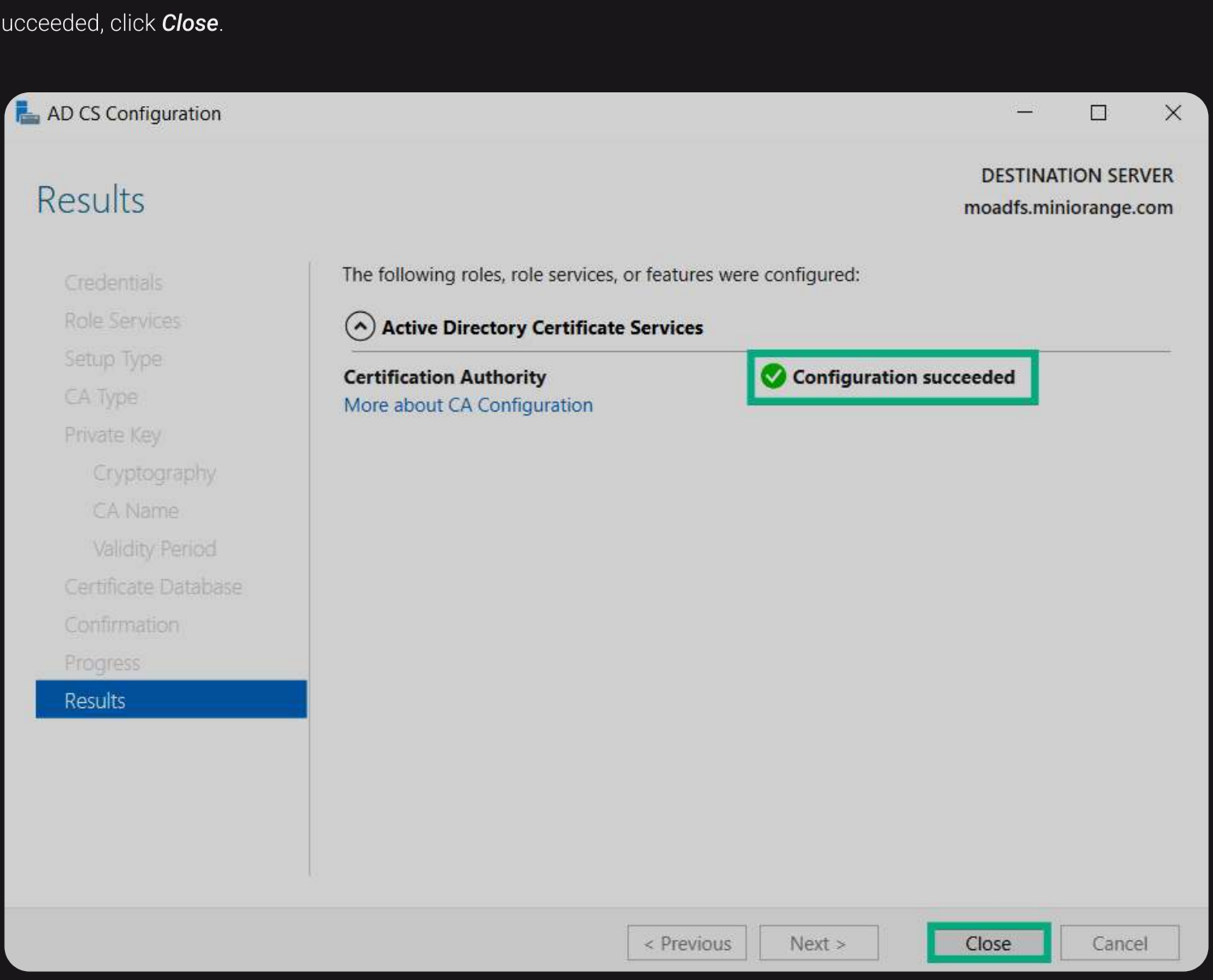
Active Directory Certificate Services

Certification Authority

CA Type:	Enterprise Root
Cryptographic provider:	RSA#Microsoft Software Key Storage Provider
Hash Algorithm:	SHA256
Key Length:	2048
Allow Administrator Interaction:	Disabled
Certificate Validity Period:	2/27/2025 10:13:00 AM
Distinguished Name:	CN=miniorange-MOADFS-CA-3,DC=miniorange,DC=com
Certificate Database Location:	C:\Windows\system32\CertLog
Certificate Database Log Location:	C:\Windows\system32\CertLog

Confirmation

22. Once the configuration is succeeded, click ***Close***.


AD CS Configuration
Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results
< Previous
Next >
Close
Cancel

Results

DESTINATION SERVER
moadfs.miniorange.com

The following roles, role services, or features were configured:

Active Directory Certificate Services

Certification Authority Configuration succeeded
More about CA Configuration

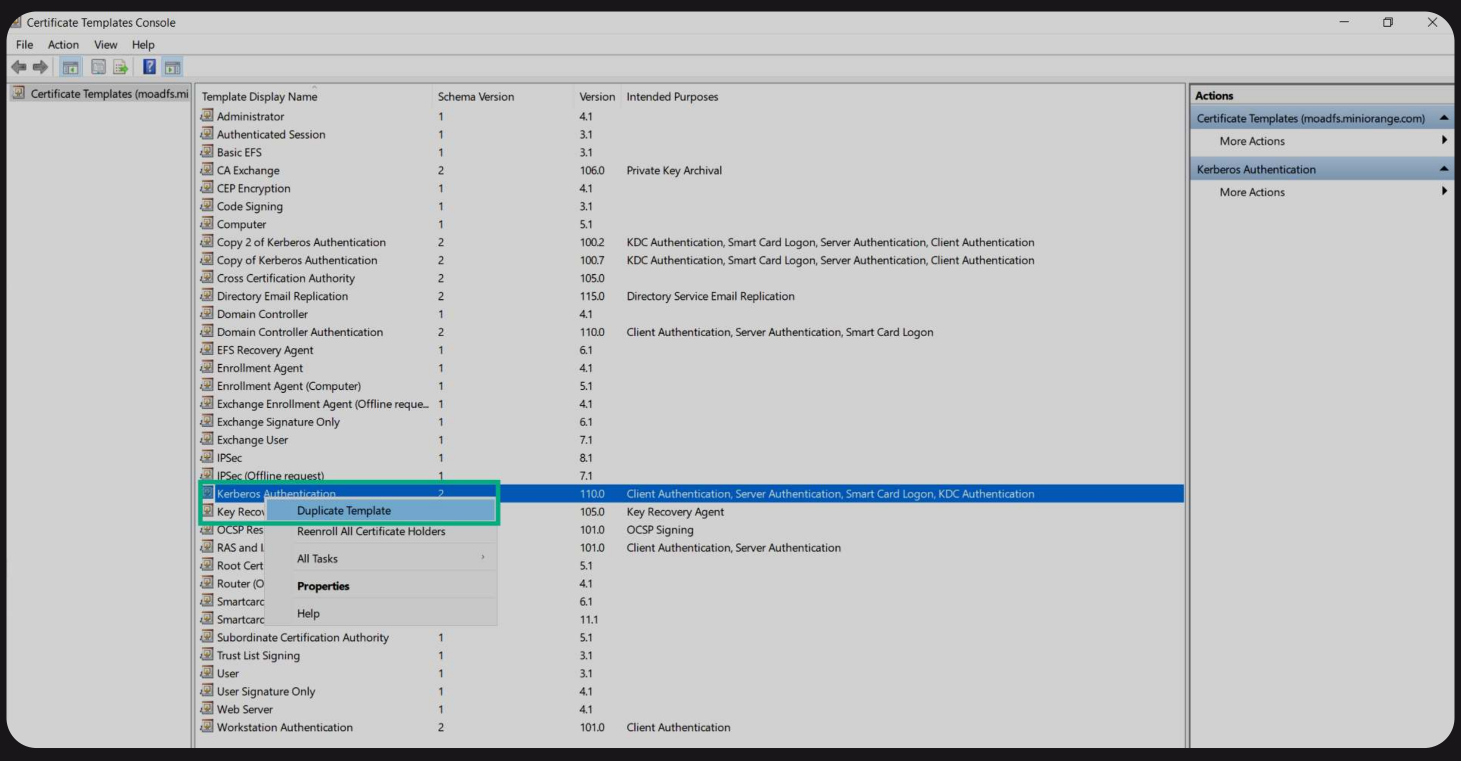
AD CS Configuration Results

Create a certificate template

Mikhail Serov

To Table of Contents

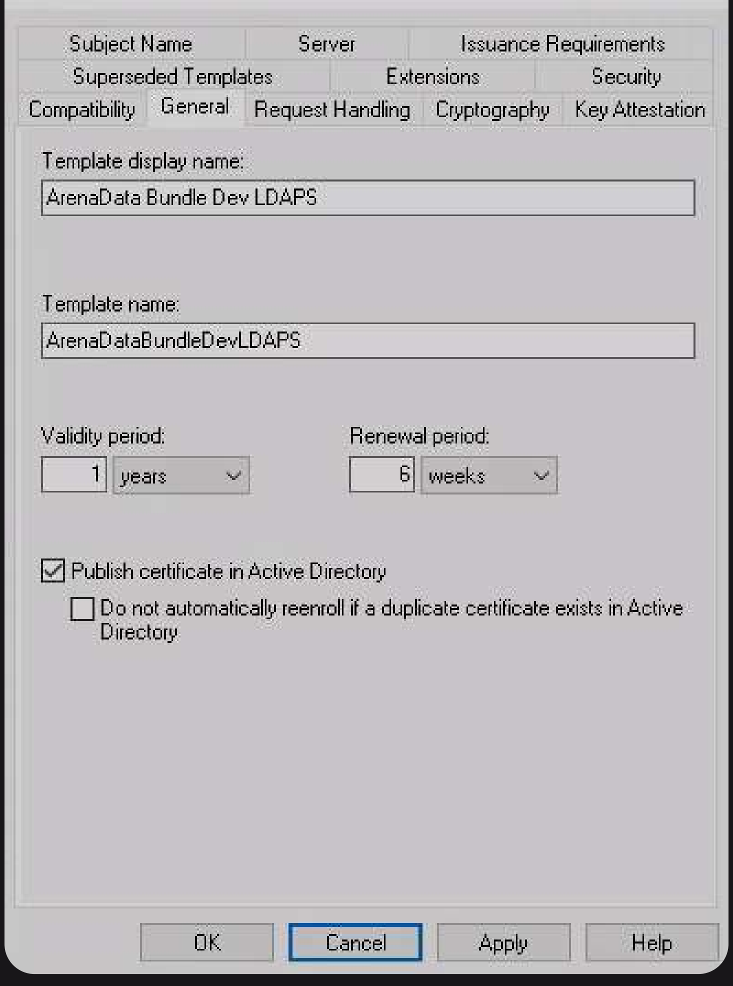
1. Press `Win+R` and run the `certtmpl.msc` command.
2. Right-click on ***Kerberos Authentication*** template and select ***Duplicate Template***.


Certificate Templates Console
File
Action
View
Help
Certificate Templates (moadfs.mi
Actions
Certificate Templates (moadfs.miniorange.com)
More Actions
Kerberos Authentication
More Actions
Duplicate Template
Reenroll All Certificate Holders
All Tasks
Properties
Help

Template Display Name	Schema Version	Version	Intended Purposes
Administrator	1	4.1	
Authenticated Session	1	3.1	
Basic EFS	1	3.1	
CA Exchange	2	106.0	Private Key Archival
CEP Encryption	1	4.1	
Code Signing	1	3.1	
Computer	1	5.1	
Copy 2 of Kerberos Authentication	2	100.2	KDC Authentication, Smart Card Logon, Server Authentication, Client Authentication
Copy of Kerberos Authentication	2	100.7	KDC Authentication, Smart Card Logon, Server Authentication, Client Authentication
Cross Certification Authority	2	105.0	
Directory Email Replication	2	115.0	Directory Service Email Replication
Domain Controller	1	4.1	
Domain Controller Authentication	2	110.0	Client Authentication, Server Authentication, Smart Card Logon
EFS Recovery Agent	1	6.1	
Enrollment Agent	1	4.1	
Enrollment Agent (Computer)	1	5.1	
Exchange Enrollment Agent (Offline reque...	1	4.1	
Exchange Signature Only	1	6.1	
Exchange User	1	7.1	
IPSec	1	8.1	
IPSec (Offline request)	1	7.1	
Kerberos Authentication	2	110.0	Client Authentication, Server Authentication, Smart Card Logon, KDC Authentication
Key Recov		105.0	Key Recovery Agent
OCSP Res		101.0	OCSP Signing
RAS and I		101.0	Client Authentication, Server Authentication
Root Cert		5.1	
Router (O		4.1	
Smartcarc		6.1	
Smartcarc		11.1	
Subordinate Certification Authority	1	5.1	
Trust List Signing	1	3.1	
User	1	3.1	
User Signature Only	1	4.1	
Web Server	1	4.1	
Workstation Authentication	2	101.0	Client Authentication

Duplicate template

3. The properties of new template appear. Go to the ***General*** tab, rename the template display name, and enable the ***Publish certificate in Active Directory*** option.


Properties of New Template
Subject Name
Server
Issuance Requirements
Superseded Templates
Extensions
Security
Compatibility
General
Request Handling
Cryptography
Key Attestation
OK
Cancel
Apply
Help

Template display name:
ArenaData Bundle Dev LDAPS

Template name:
ArenaDataBundleDevLDAPS

Validity period: 1 years

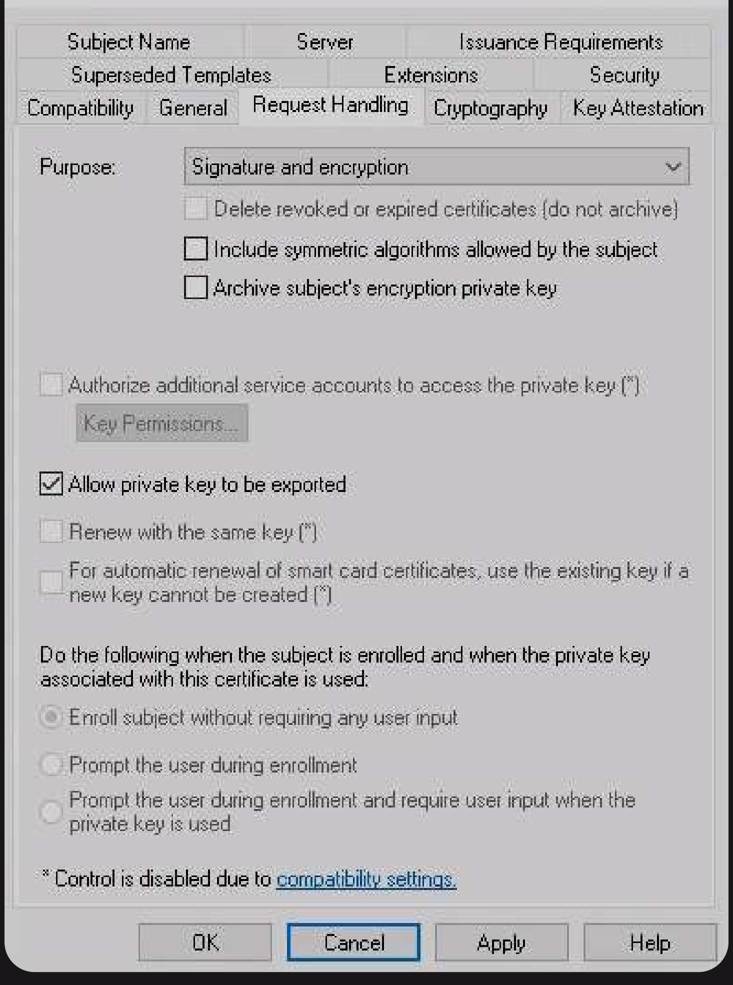
Renewal period: 6 weeks

☑ Publish certificate in Active Directory

☐ Do not automatically reenroll if a duplicate certificate exists in Active Directory

Properties of New Template

4. Go to the ***Request Handling*** tab and enable the ***Allow private key to be exported*** option.


Properties of New Template
Subject Name
Server
Issuance Requirements
Superseded Templates
Extensions
Security
Compatibility
General
Request Handling
Cryptography
Key Attestation
Key Permissions...
OK
Cancel
Apply
Help

Purpose: Signature and encryption

☐ Delete revoked or expired certificates (do not archive)

☐ Include symmetric algorithms allowed by the subject

☐ Archive subject's encryption private key

☐ Authorize additional service accounts to access the private key (*)

☑ Allow private key to be exported

☐ Renew with the same key (*)

☐ For automatic renewal of smart card certificates, use the existing key if a new key cannot be created (*)

Do the following when the subject is enrolled and when the private key associated with this certificate is used:

● Enroll subject without requiring any user input

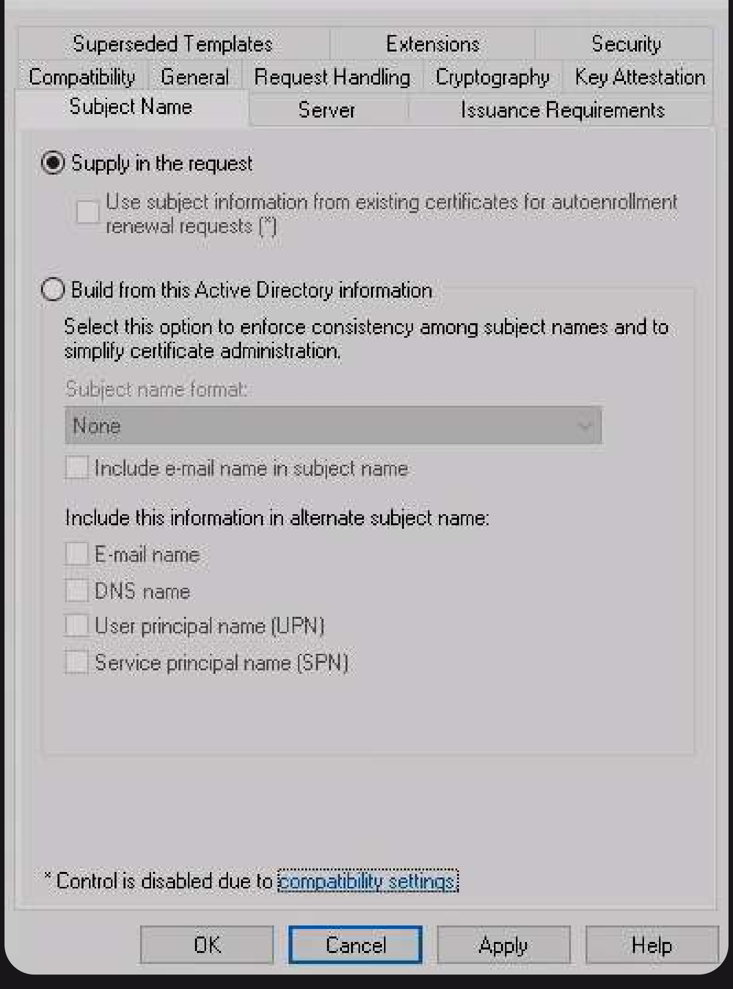
○ Prompt the user during enrollment

○ Prompt the user during enrollment and require user input when the private key is used

* Control is disabled due to compatibility settings.

Allow private key export

5. Go to the ***Subject Name*** tab and select ***Supply in the request***.


Properties of New Template
Superseded Templates
Extensions
Security
Compatibility
General
Request Handling
Cryptography
Key Attestation
Subject Name
Server
Issuance Requirements
OK
Cancel
Apply
Help

● Supply in the request

☐ Use subject information from existing certificates for autoenrollment renewal requests (*)

○ Build from this Active Directory information

Select this option to enforce consistency among subject names and to simplify certificate administration.

Subject name format:
None

☐ Include e-mail name in subject name

Include this information in alternate subject name:

☐ E-mail name

☐ DNS name

☐ User principal name (UPN)

☐ Service principal name (SPN)

* Control is disabled due to compatibility settings.

Supply subject name

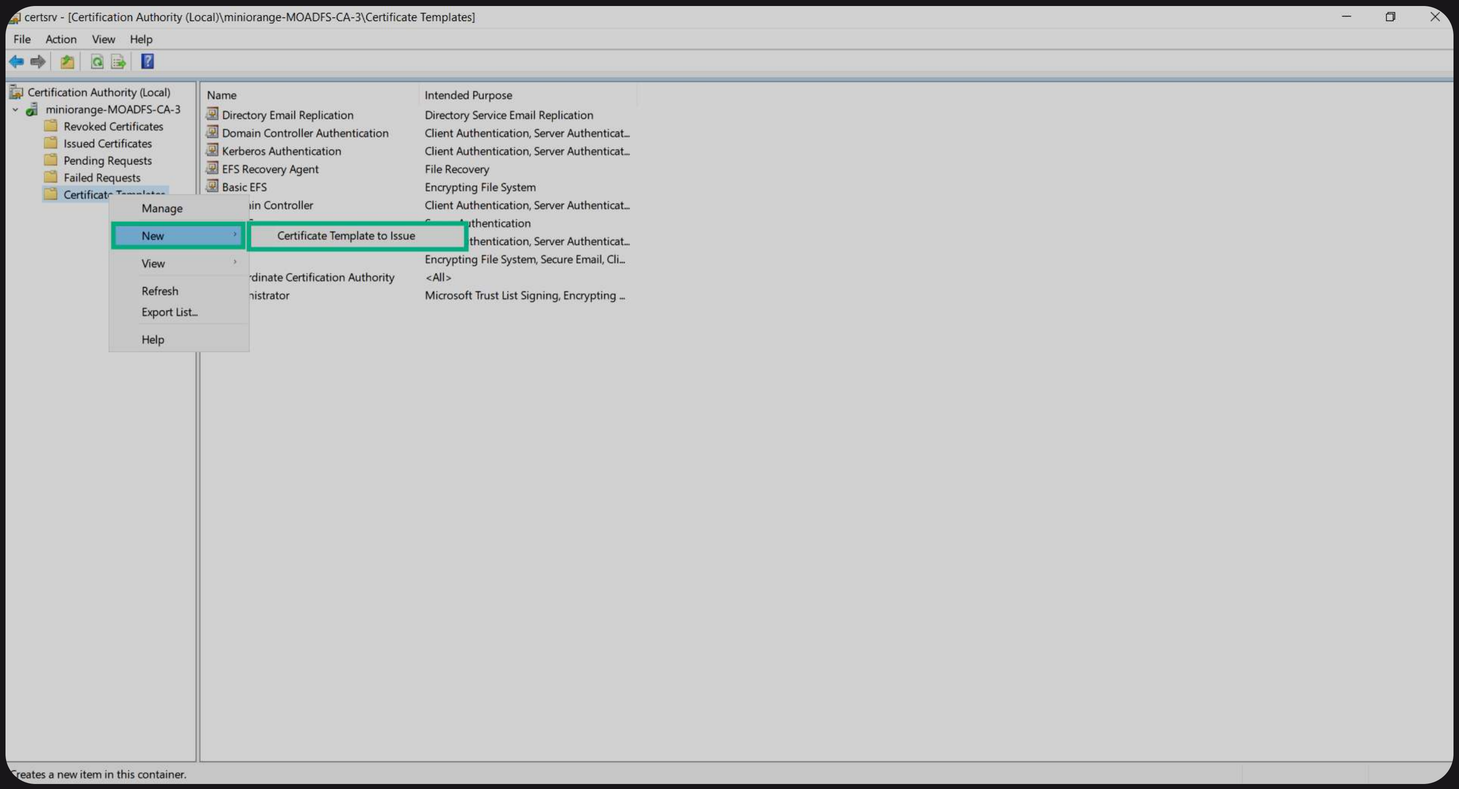
6. Click ***Apply*** and then click ***OK***.

Issue a certificate template

Mikhail Serov

To Table of Contents

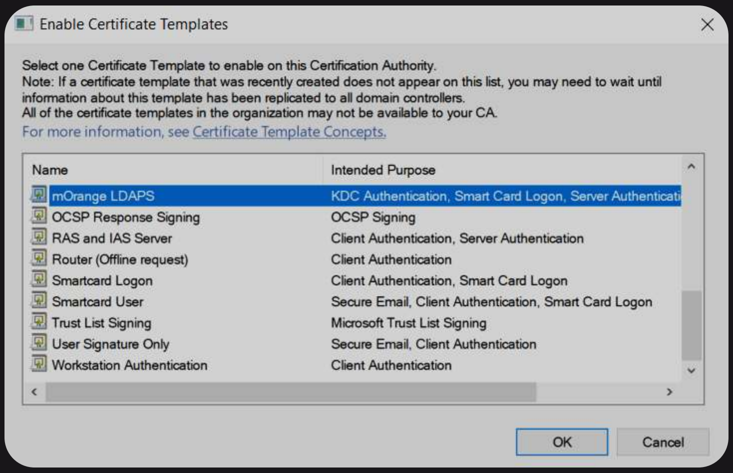
1. Go to ***Start*** → ***Certification Authority***.
2. Right-click ***Certificate Templates*** and then select ***New*** → ***Certificate Template to Issue***.


certsrv - [Certification Authority (Local)\miniorange-MOADFS-CA-3\Certificate Templates]
File
Action
View
Help
Certification Authority (Local)
miniorange-MOADFS-CA-3
Revoked Certificates
Issued Certificates
Pending Requests
Failed Requests
Manage
New
Certificate Template to Issue
View
Refresh
Export List...
Help
reates a new item in this container.

Name	Intended Purpose
Directory Email Replication	Directory Service Email Replication
Domain Controller Authentication	Client Authentication, Server Authenticat...
Kerberos Authentication	Client Authentication, Server Authenticat...
EFS Recovery Agent	File Recovery
Basic EFS	Encrypting File System
in Controller	Client Authentication, Server Authenticat...
	uthentication
	thentication, Server Authenticat...
	Encrypting File System, Secure Email, Cli...
rdinate Certification Authority	<All>
nistrator	Microsoft Trust List Signing, Encrypting ...

New certificate template to issue

3. Select your recently created certificate template and click ***OK***.


Enable Certificate Templates
OK
Cancel

Select one Certificate Template to enable on this Certification Authority.
Note: If a certificate template that was recently created does not appear on this list, you may need to wait until information about this template has been replicated to all domain controllers.
All of the certificate templates in the organization may not be available to your CA.
For more information, see Certificate Template Concepts.

Name	Intended Purpose
mOrange LDAPS	KDC Authentication, Smart Card Logon, Server Authenticati
OCSP Response Signing	OCSP Signing
RAS and IAS Server	Client Authentication, Server Authentication
Router (Offline request)	Client Authentication
Smartcard Logon	Client Authentication, Smart Card Logon
Smartcard User	Secure Email, Client Authentication, Smart Card Logon
Trust List Signing	Microsoft Trust List Signing
User Signature Only	Secure Email, Client Authentication
Workstation Authentication	Client Authentication

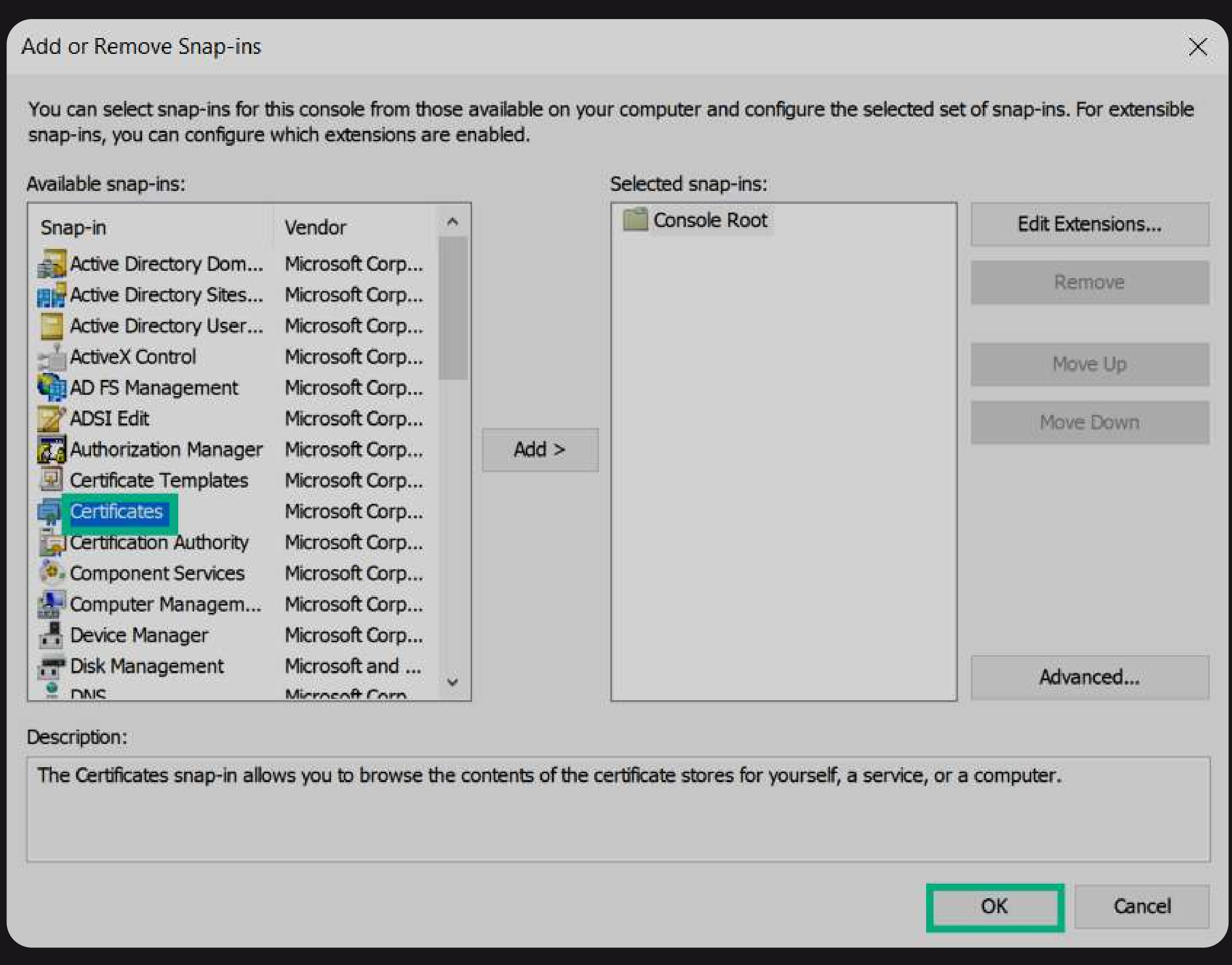
Certificate selection

Request a new certificate

Mikhail Serov

To Table of Contents

1. Press `Win+R` and run the `mmc` command.
2. Click ***File* → *Add or Remove Snap-ins***.
3. Select ***Certificates*** and click ***Add***. Then, click ***OK***.


Add or Remove Snap-ins
Available snap-ins:
Snap-in
Vendor
Active Directory Dom...
Active Directory Sites...
Active Directory User...
ActiveX Control
AD FS Management
ADSI Edit
Authorization Manager
Certificate Templates
Certificates
Certification Authority
Component Services
Computer Managem...
Device Manager
Disk Management
DNS
Microsoft Corp...
Microsoft and ...
Add >
Selected snap-ins:
Console Root
Edit Extensions...
Remove
Move Up
Move Down
Advanced...
OK
Cancel

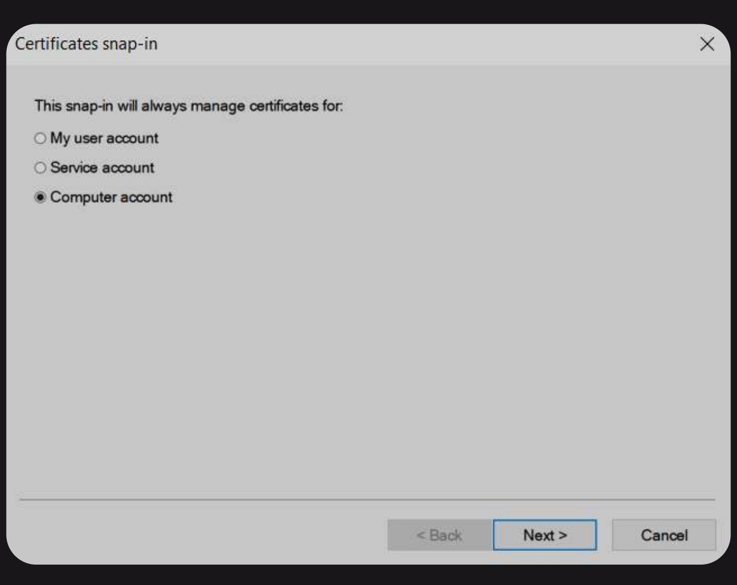
You can select snap-ins for this console from those available on your computer and configure the selected set of snap-ins. For extensible snap-ins, you can configure which extensions are enabled.

Description:

The Certificates snap-in allows you to browse the contents of the certificate stores for yourself, a service, or a computer.

Certificates snap-ins

4. Select ***Computer account*** and click ***Next***.

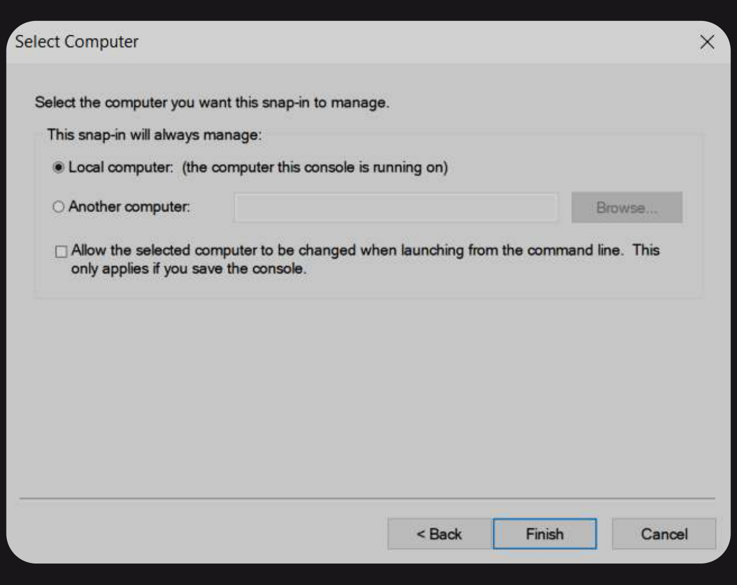

Certificates snap-in
< Back
Next >
Cancel

This snap-in will always manage certificates for:

○ My user account
○ Service account
● Computer account

Account type

5. Select ***Local computer*** and click ***Finish***.


Select Computer
Browse...
< Back
Finish
Cancel

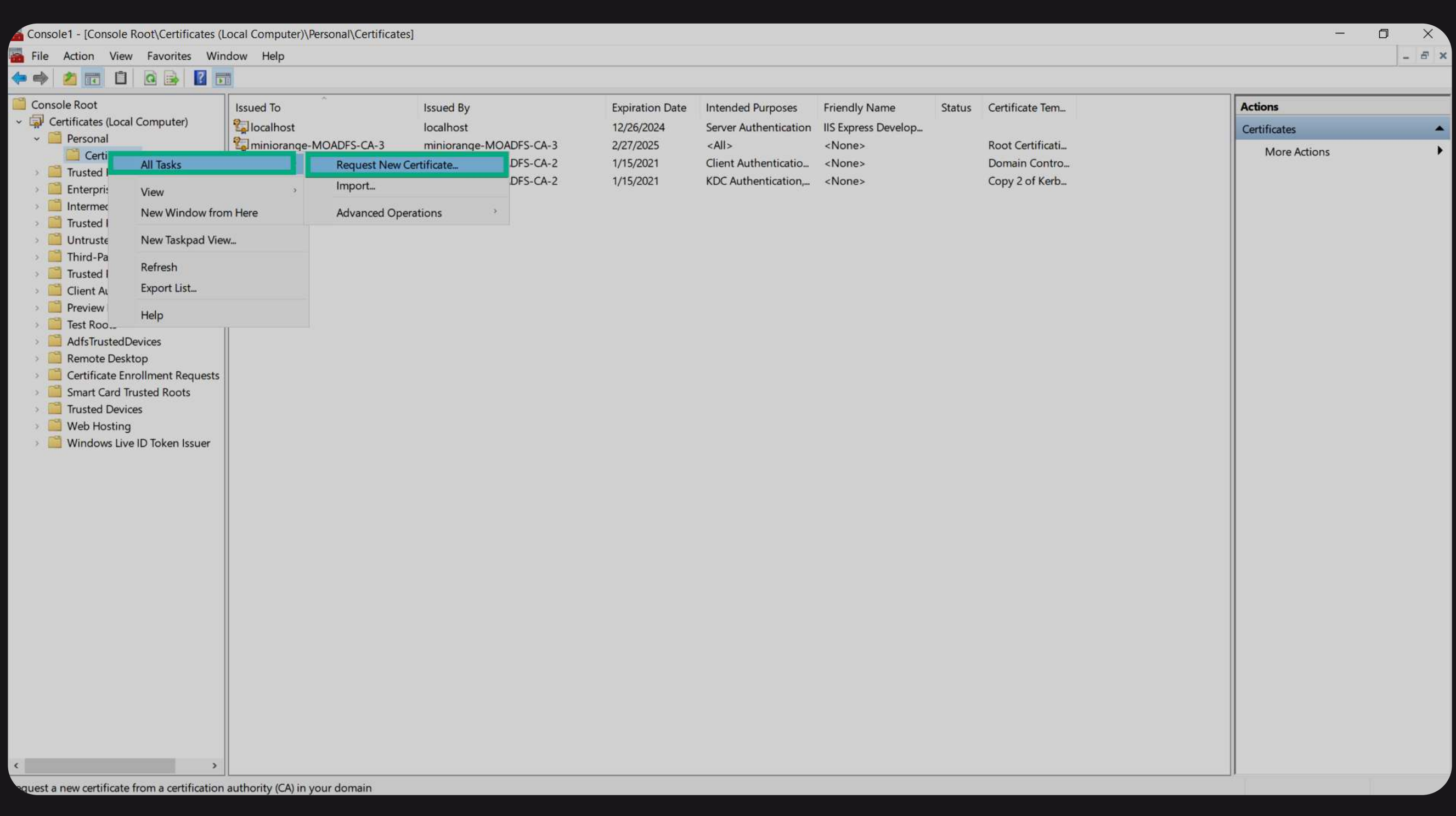
Select the computer you want this snap-in to manage.

This snap-in will always manage:

● Local computer: (the computer this console is running on)
○ Another computer:
☐ Allow the selected computer to be changed when launching from the command line. This only applies if you save the console.

Computer selection

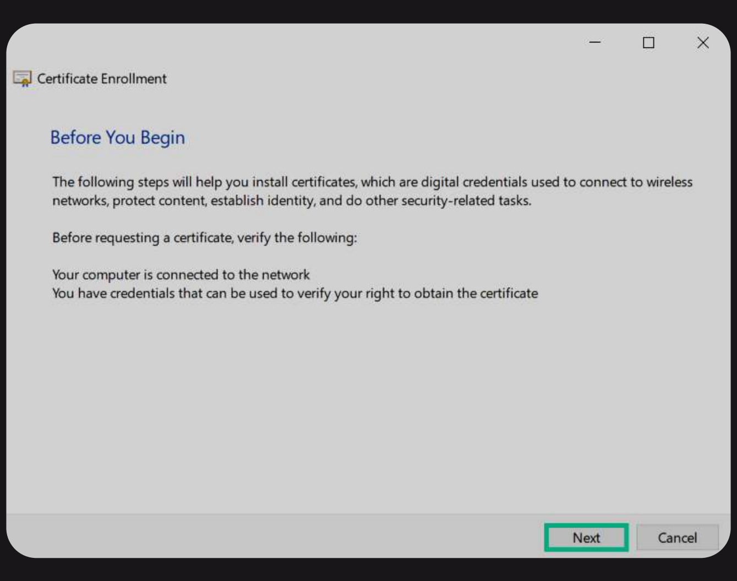
6. Right-click ***Certificates***, select ***All Tasks***, and click ***Request New Certificate***.


Console1 - [Console Root\Certificates (Local Computer)\Personal\Certificates]
File
Action
View
Favorites
Window
Help
Console Root
Certificates (Local Computer)
Personal
AdfsTrustedDevices
Remote Desktop
Certificate Enrollment Requests
Smart Card Trusted Roots
Trusted Devices
Web Hosting
Windows Live ID Token Issuer
All Tasks
View
New Window from Here
New Taskpad View...
Refresh
Export List...
Help
Request New Certificate...
Import...
Advanced Operations
Issued To
Issued By
Expiration Date
Intended Purposes
Friendly Name
Status
Certificate Tem...
Actions
Certificates
More Actions

Request a new certificate from a certification authority (CA) in your domain

Request new certificate

7. Click ***Next***.


Certificate Enrollment
Next
Cancel

Before You Begin

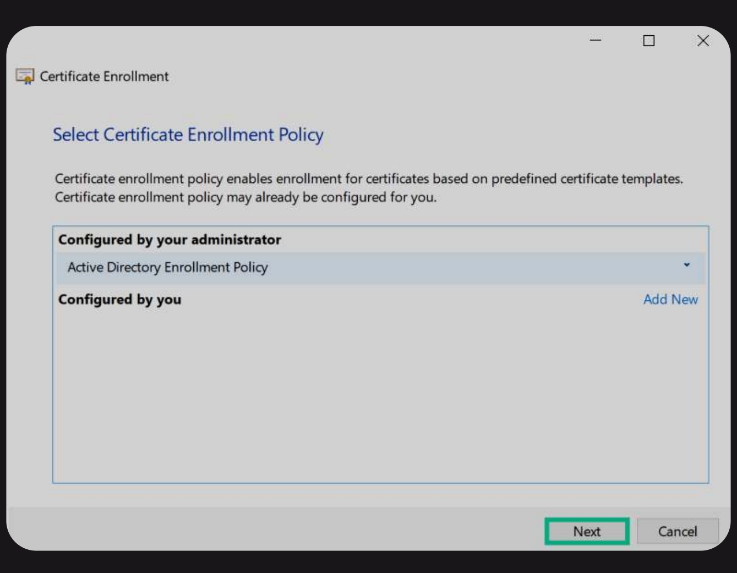
The following steps will help you install certificates, which are digital credentials used to connect to wireless networks, protect content, establish identity, and do other security-related tasks.

Before requesting a certificate, verify the following:

Your computer is connected to the network
You have credentials that can be used to verify your right to obtain the certificate

Certificate enrollment

8. Click ***Next***.


Certificate Enrollment
Add New
Next
Cancel

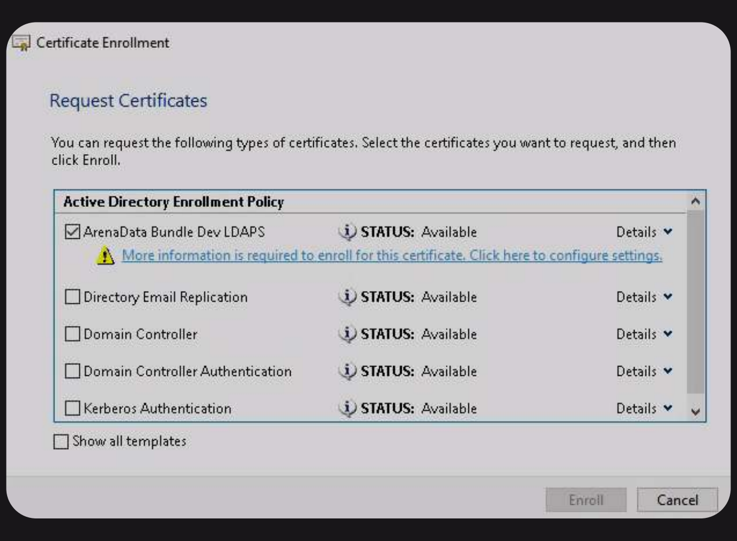
Select Certificate Enrollment Policy

Certificate enrollment policy enables enrollment for certificates based on predefined certificate templates. Certificate enrollment policy may already be configured for you.

Configured by your administrator
Active Directory Enrollment Policy
Configured by you

Select enrollment policy

9. Select the previously created template and click ***More information is required…***.


Certificate Enrollment
Details
Enroll
Cancel

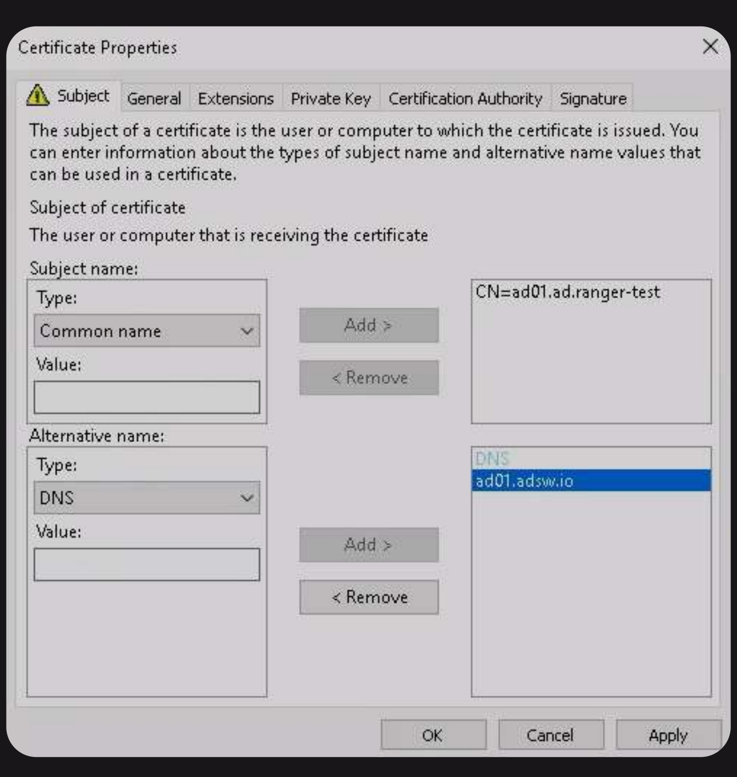
Request Certificates

You can request the following types of certificates. Select the certificates you want to request, and then click Enroll.

Active Directory Enrollment Policy
☑ ArenaData Bundle Dev LDAPS **STATUS:** Available
More information is required to enroll for this certificate. Click here to configure settings.
☐ Directory Email Replication **STATUS:** Available
☐ Domain Controller **STATUS:** Available
☐ Domain Controller Authentication **STATUS:** Available
☐ Kerberos Authentication **STATUS:** Available
☐ Show all templates

Additional certificate information

10. Enter `CN=ad01.ad.ranger-test` and `DNS=ad01.adsw.io` (or other corresponding computer names) in ***AD domain*** and ***DNS name*** fields. Click ***Apply*** and then click ***OK***.


Certificate Properties
Subject
General
Extensions
Private Key
Certification Authority
Signature
Add >
< Remove
OK
Cancel
Apply

The subject of a certificate is the user or computer to which the certificate is issued. You can enter information about the types of subject name and alternative name values that can be used in a certificate.

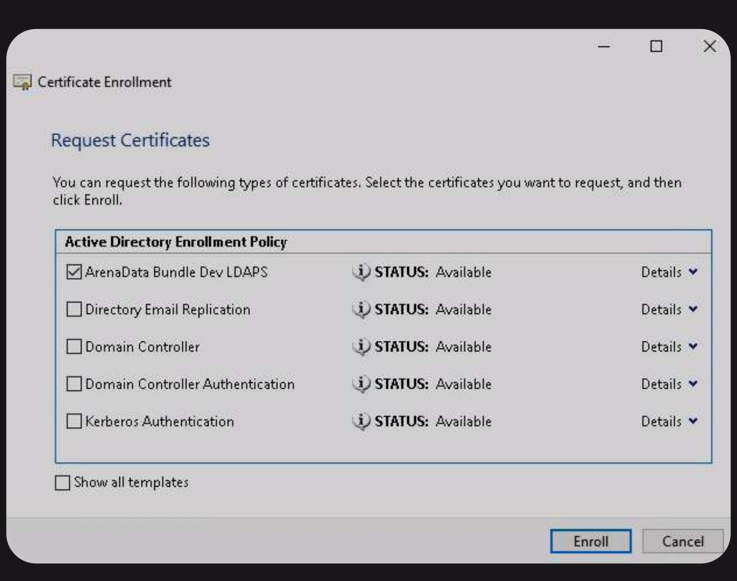
Subject of certificate
The user or computer that is receiving the certificate

Subject name:
Type: Common name
Value:
CN=ad01.ad.ranger-test

Alternative name:
Type: DNS
Value:
DNS
ad01.adsw.io

Certificate properties

11. Click ***Enroll***.


Certificate Enrollment
Details
Enroll
Cancel

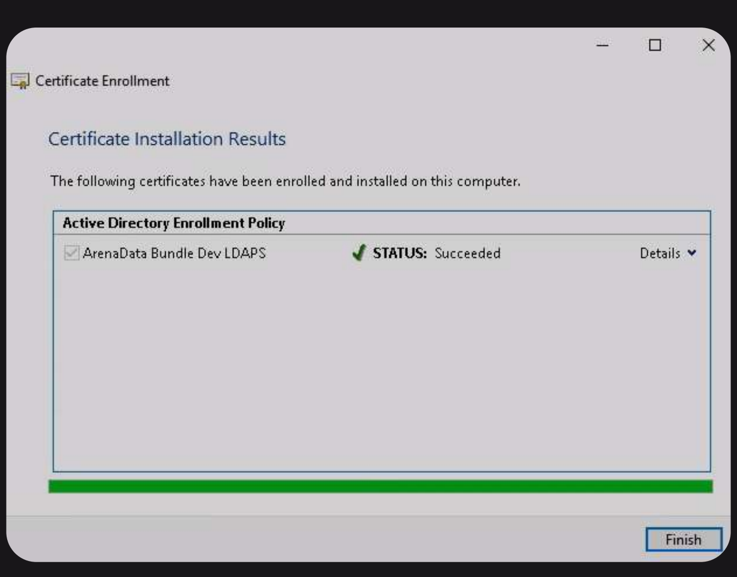
Request Certificates

You can request the following types of certificates. Select the certificates you want to request, and then click Enroll.

Active Directory Enrollment Policy
☑ ArenaData Bundle Dev LDAPS **STATUS:** Available
☐ Directory Email Replication **STATUS:** Available
☐ Domain Controller **STATUS:** Available
☐ Domain Controller Authentication **STATUS:** Available
☐ Kerberos Authentication **STATUS:** Available
☐ Show all templates

Enroll certificate

12. Click ***Finish***.


Certificate Enrollment
Details
Finish

Certificate Installation Results

The following certificates have been enrolled and installed on this computer.

Active Directory Enrollment Policy
☑ ArenaData Bundle Dev LDAPS **STATUS:** Succeeded

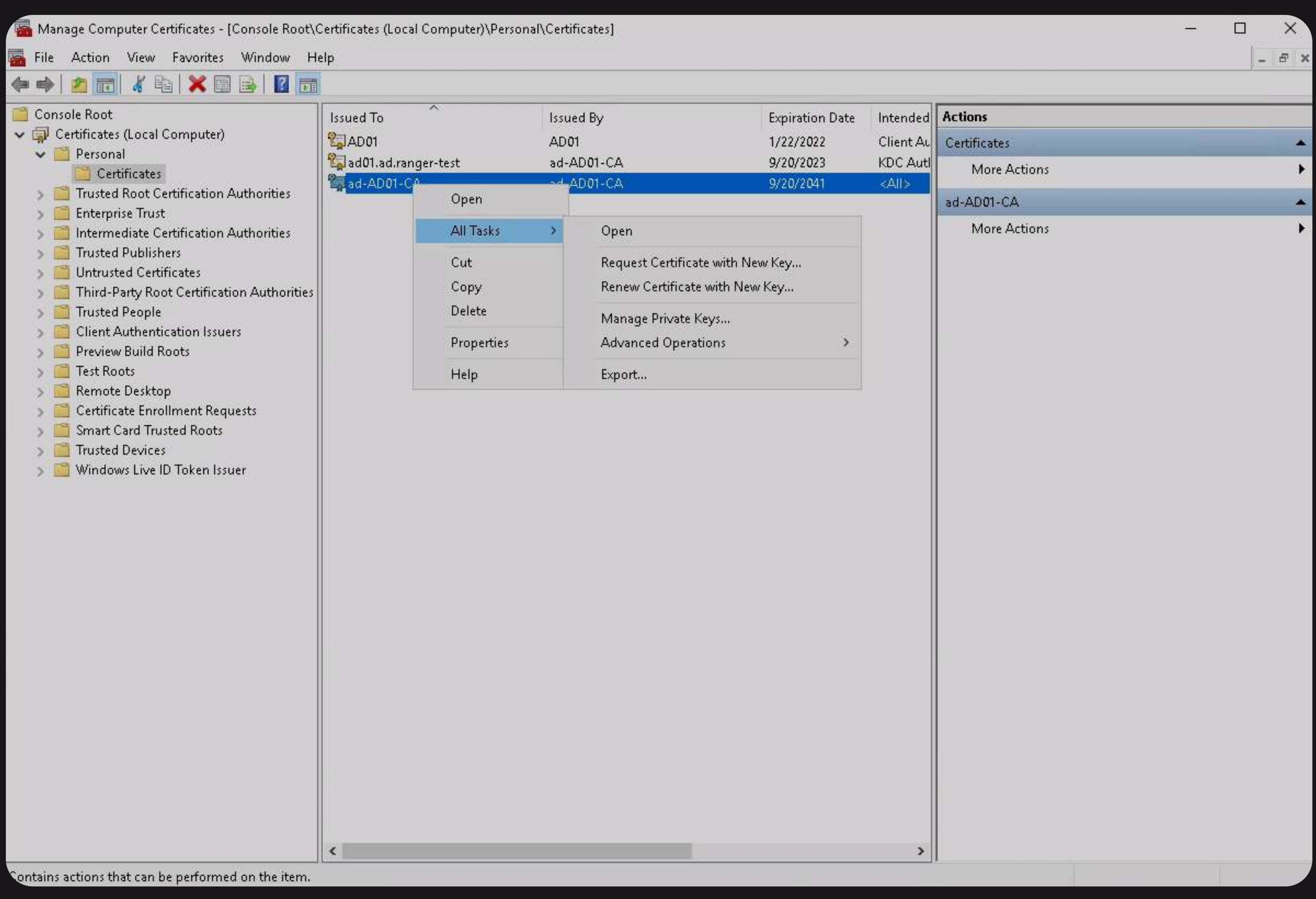
Certificate installation results

Export CA certificate

Mikhail Serov

To Table of Contents

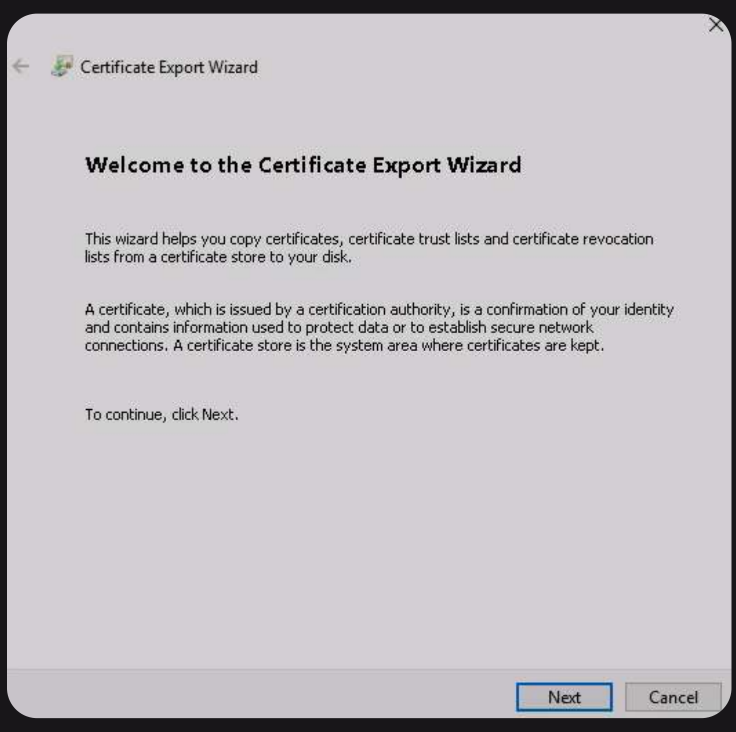
1. Open the `mmc` certificates management snap-in (`Win+R` → Run `mmc` → ***File*** → ***Add or Remove Snap-ins***).
2. Right-click your CA certificate and select ***All Tasks*** → ***Export***.


Manage Computer Certificates - [Console Root\Certificates (Local Computer)\Personal\Certificates]
File
Action
View
Favorites
Window
Help
Console Root
Certificates (Local Computer)
Personal
Certificates
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Third-Party Root Certification Authorities
Trusted People
Client Authentication Issuers
Preview Build Roots
Test Roots
Remote Desktop
Certificate Enrollment Requests
Smart Card Trusted Roots
Trusted Devices
Windows Live ID Token Issuer
Open
All Tasks
Cut
Copy
Delete
Properties
Help
Open
Request Certificate with New Key...
Renew Certificate with New Key...
Manage Private Keys...
Advanced Operations
Export...
Actions
Certificates
More Actions
ad-AD01-CA
More Actions
Contains actions that can be performed on the item.

Issued To	Issued By	Expiration Date	Intended
AD01	AD01	1/22/2022	Client Au
ad01.ad.ranger-test	ad-AD01-CA	9/20/2023	KDC Auth
ad-AD01-CA	ad-AD01-CA	9/20/2041	<All>

Export certificate

3. Click ***Next***.


Certificate Export Wizard
Next
Cancel

Welcome to the Certificate Export Wizard

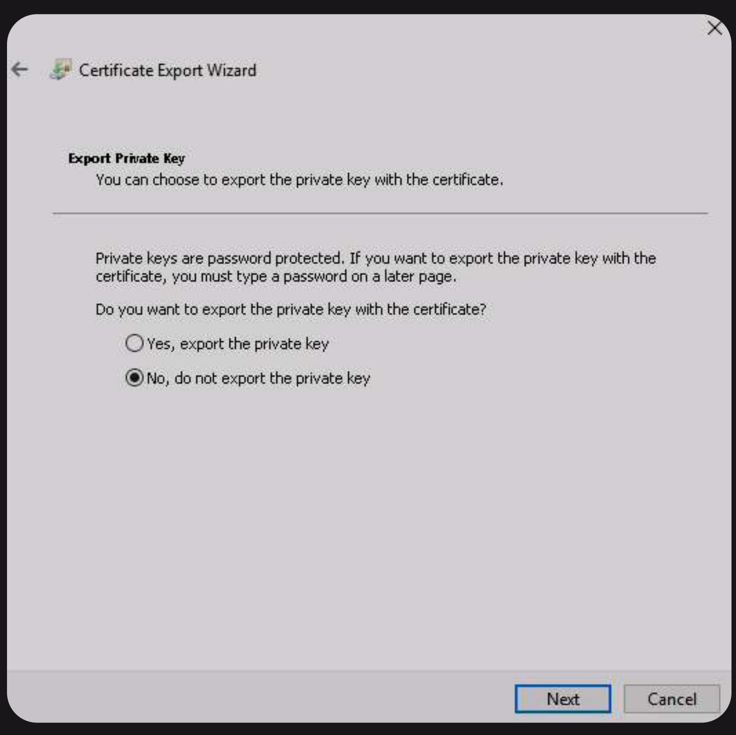
This wizard helps you copy certificates, certificate trust lists and certificate revocation lists from a certificate store to your disk.

A certificate, which is issued by a certification authority, is a confirmation of your identity and contains information used to protect data or to establish secure network connections. A certificate store is the system area where certificates are kept.

To continue, click Next.

Certificate Export Wizard

4. Select ***No, don't export the private key***, then click ***Next***.


Certificate Export Wizard
Next
Cancel

Export Private Key
You can choose to export the private key with the certificate.

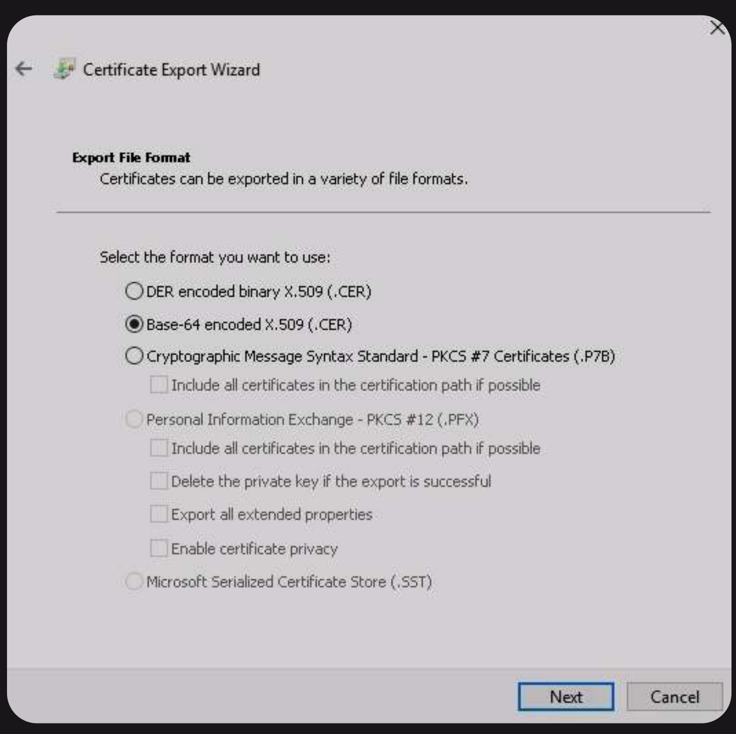
Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

Do you want to export the private key with the certificate?

○ Yes, export the private key
● No, do not export the private key

Export private key

5. Select the ***Base-64 encoded X.509*** format and then click ***Next***.

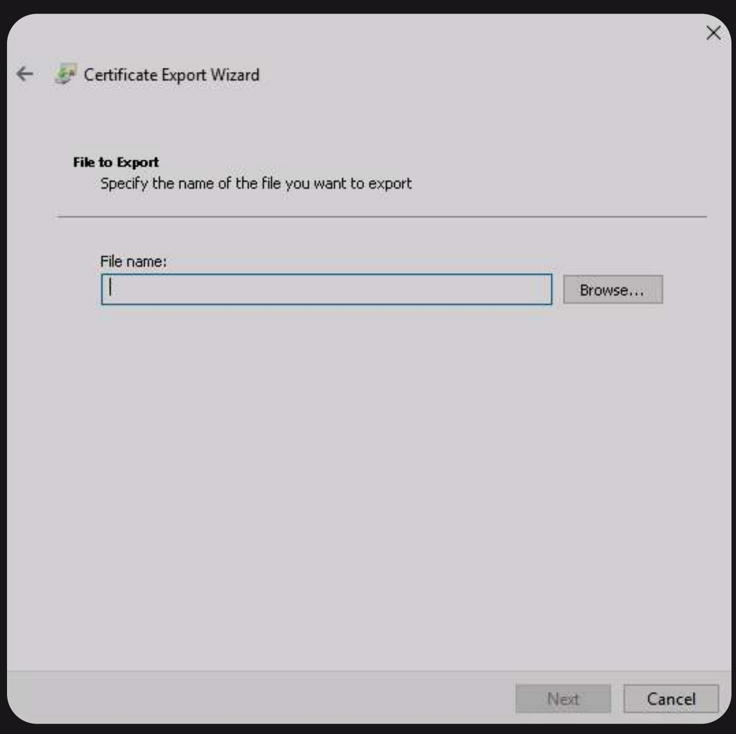

Certificate Export Wizard
Next
Cancel

Export File Format
Certificates can be exported in a variety of file formats.

Select the format you want to use:
○ DER encoded binary X.509 (.CER)
● Base-64 encoded X.509 (.CER)
○ Cryptographic Message Syntax Standard - PKCS #7 Certificates (.P7B)
  ☐ Include all certificates in the certification path if possible
○ Personal Information Exchange - PKCS #12 (.PFX)
  ☐ Include all certificates in the certification path if possible
  ☐ Delete the private key if the export is successful
  ☐ Export all extended properties
  ☐ Enable certificate privacy
○ Microsoft Serialized Certificate Store (.SST)

Export file format

6. Fill in the file name for your certificate, then click ***Next***.

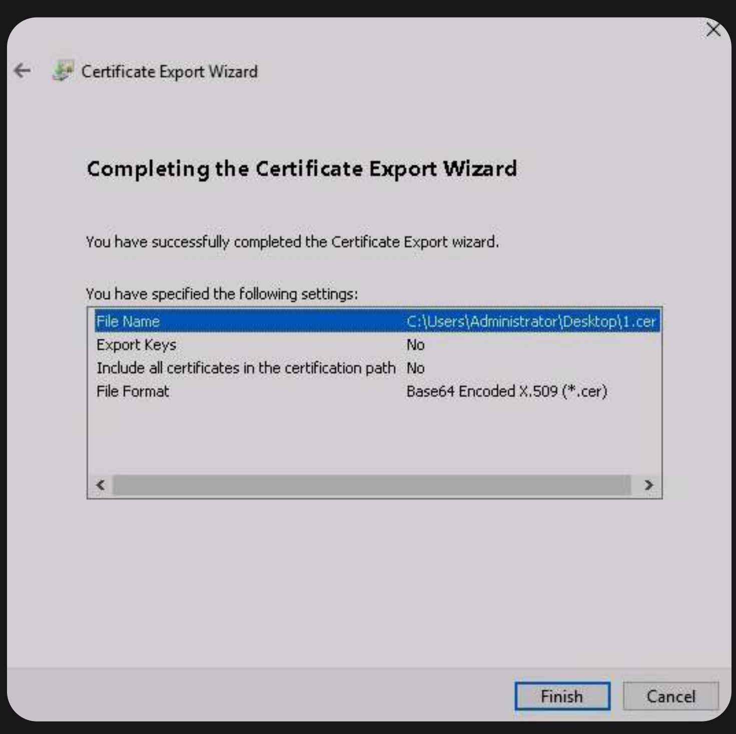

Certificate Export Wizard
Browse...
Next
Cancel

File to Export
Specify the name of the file you want to export

File name:

File name to export

7. Click ***Finish***.


Certificate Export Wizard
Finish
Cancel

Completing the Certificate Export Wizard

You have successfully completed the Certificate Export wizard.

You have specified the following settings:

File Name	C:\Users\Administrator\Desktop\1.cer
Export Keys	No
Include all certificates in the certification path	No
File Format	Base64 Encoded X.509 (*.cer)

Certificate export complete

Configure Kerberos authentication based on Active Directory via ADCM

Mikhail Serov, Sergei Tikhomirov

Contents

To Table of Contents

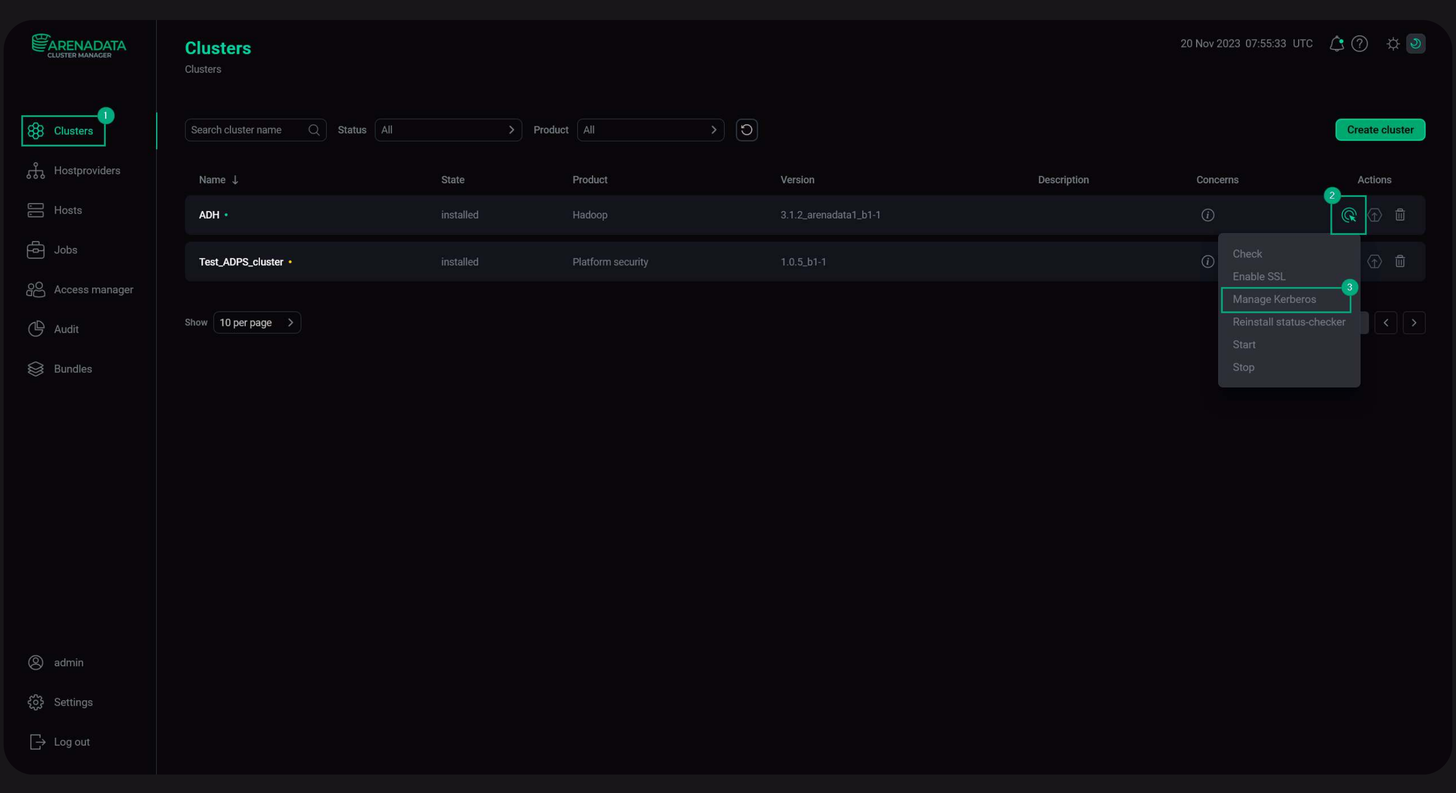
Overview

Active Directory parameters

Overview

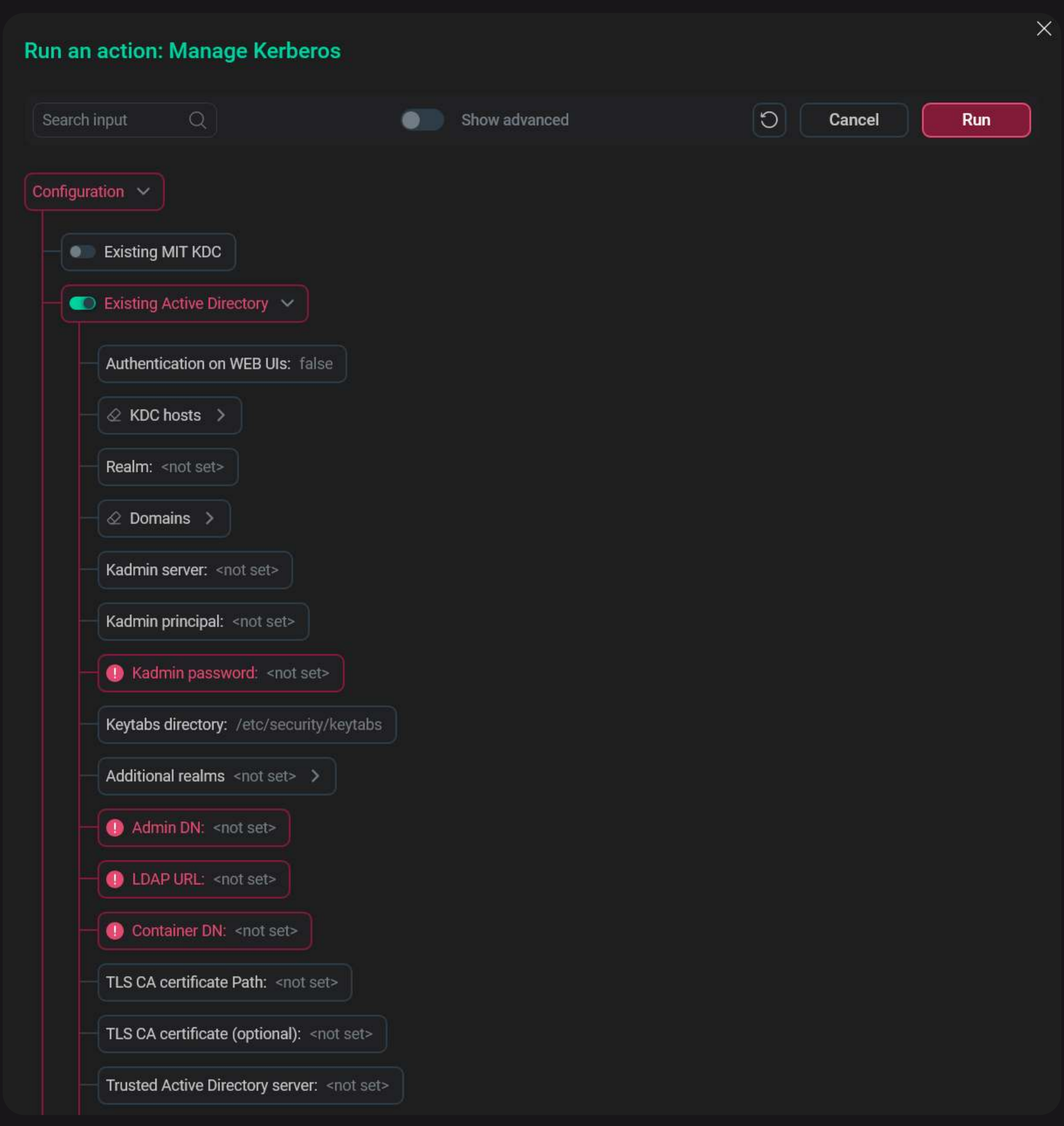
To kerberize a cluster using Active Directory, follow the steps below:

1. In ADCM web UI, go to the *Clusters* page. Select an installed and prepared ADH cluster, and run the *Manage Kerberos* action.



Manage Kerberos

2. In the pop-up window, turn on the *Existing Active Directory* option.



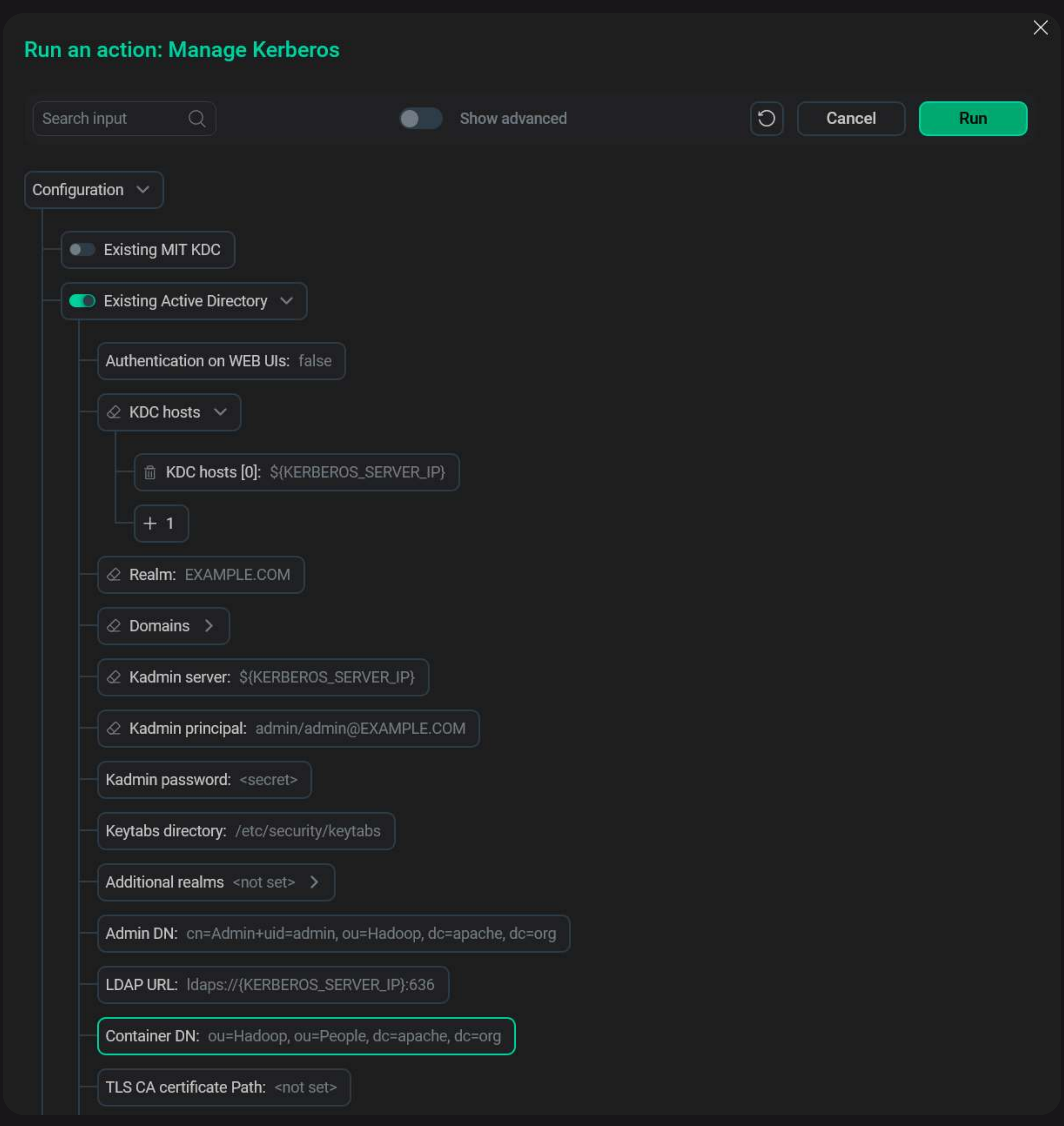
Run an action: Manage Kerberos

Configuration

- Existing MIT KDC
- Existing Active Directory
  - Authentication on WEB UIs: false
  - KDC hosts
  - Realm: <not set>
  - Domains
  - Kadmin server: <not set>
  - Kadmin principal: <not set>
  - Kadmin password: <not set>
  - Keytabs directory: /etc/security/keytabs
  - Additional realms <not set>
  - Admin DN: <not set>
  - LDAP URL: <not set>
  - Container DN: <not set>
  - TLS CA certificate Path: <not set>
  - TLS CA certificate (optional): <not set>
  - Trusted Active Directory server: <not set>

Choose the relevant option

3. Fill in the Active Directory parameters.



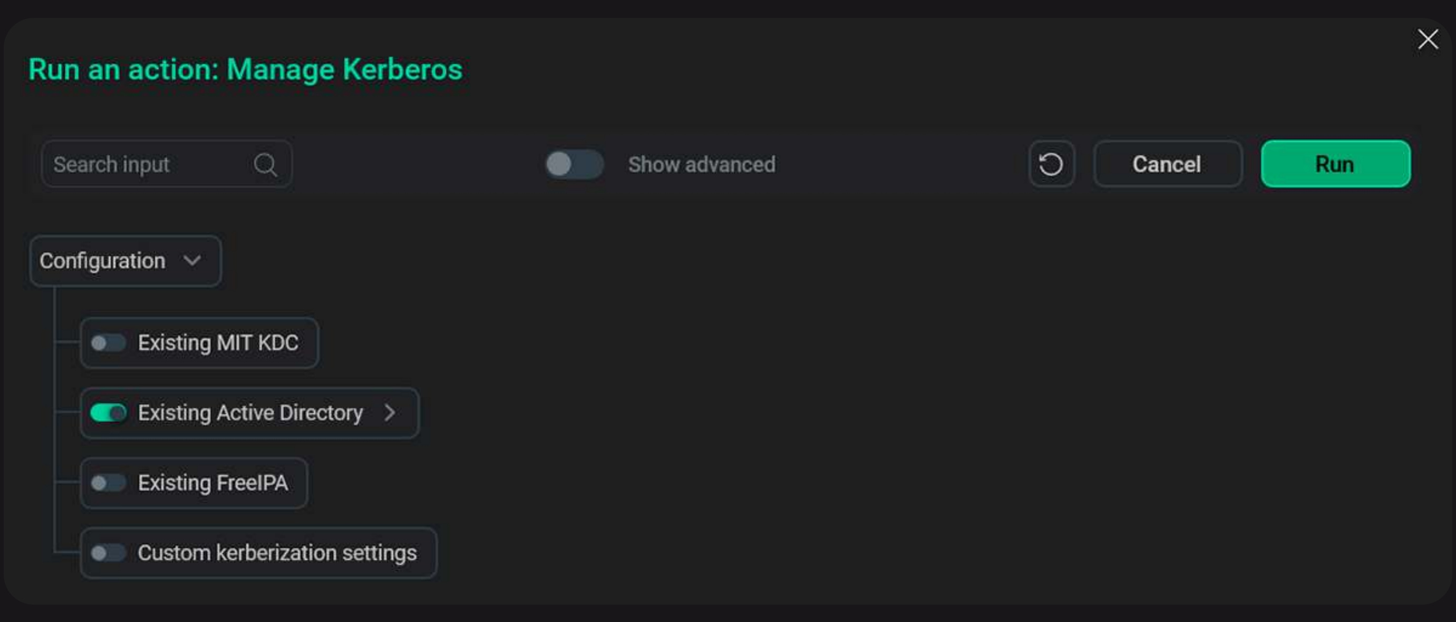
Run an action: Manage Kerberos

Configuration

- Existing MIT KDC
- Existing Active Directory
  - Authentication on WEB UIs: false
  - KDC hosts
    - KDC hosts [0]: ${KERBEROS_SERVER_IP}
  - Realm: EXAMPLE.COM
  - Domains
  - Kadmin server: ${KERBEROS_SERVER_IP}
  - Kadmin principal: admin/admin@EXAMPLE.COM
  - Kadmin password: <secret>
  - Keytabs directory: /etc/security/keytabs
  - Additional realms <not set>
  - Admin DN: cn=Admin+uid=admin, ou=Hadoop, dc=apache, dc=org
  - LDAP URL: ldaps://{KERBEROS_SERVER_IP}:636
  - Container DN: ou=Hadoop, ou=People, dc=apache, dc=org
  - TLS CA certificate Path: <not set>

Active Directory fields

4. Click *Run*, wait for the job to complete and proceed to setting up Kerberos in the cluster.



Run an action: Manage Kerberos

Configuration

- Existing MIT KDC
- Existing Active Directory
- Existing FreeIPA
- Custom kerberization settings

Run the action

Active Directory parameters

Parameter	Description
Authentication on WEB UIs	Enables Kerberos authentication on Web UIs
KDC hosts	One or more KDC hosts
Realm	A Kerberos realm
Domains	Domains associated with hosts
Kadmin server	A host where `kadmin` is running
Kadmin principal	A principal name used to connect via `kadmin`, for example `admin@RU-CENTRAL1.INTERNAL`
Kadmin password	A principal password used to connect via `kadmin`
Keytabs directory	Directory of the keytab file that contains one or several principals along with their keys
Additional realms	Additional Kerberos realms
Admin DN	Full distiguished name of admin user with rights create, modify, delete, pwdchange user accounts in the target Organizational Unit
LDAP URL	LDAP URL consists of `ldap://` or `ldaps://`, hostname or IP address, and port of the AD server
Container DN	Container distinguished name
TLS CA certificate Path	CA certificate path on the host's filesystem. Should be preplaced or will be written down from the `TLS CA certificate` field
TLS CA certificate (optional)	CA certificate to be written on a host using the path from the `TLS CA certificate Path` field
Trusted Active Directory server	An Active Directory server for one-way cross-realm trust from the MIT Kerberos KDC
Trusted Active Directory realm	An Active Directory realm for one-way cross-realm trust from the MIT Kerberos KDC

Enable Kerberos with prepared Active Directory server and no admin permissions

Sergei Tikhomirov

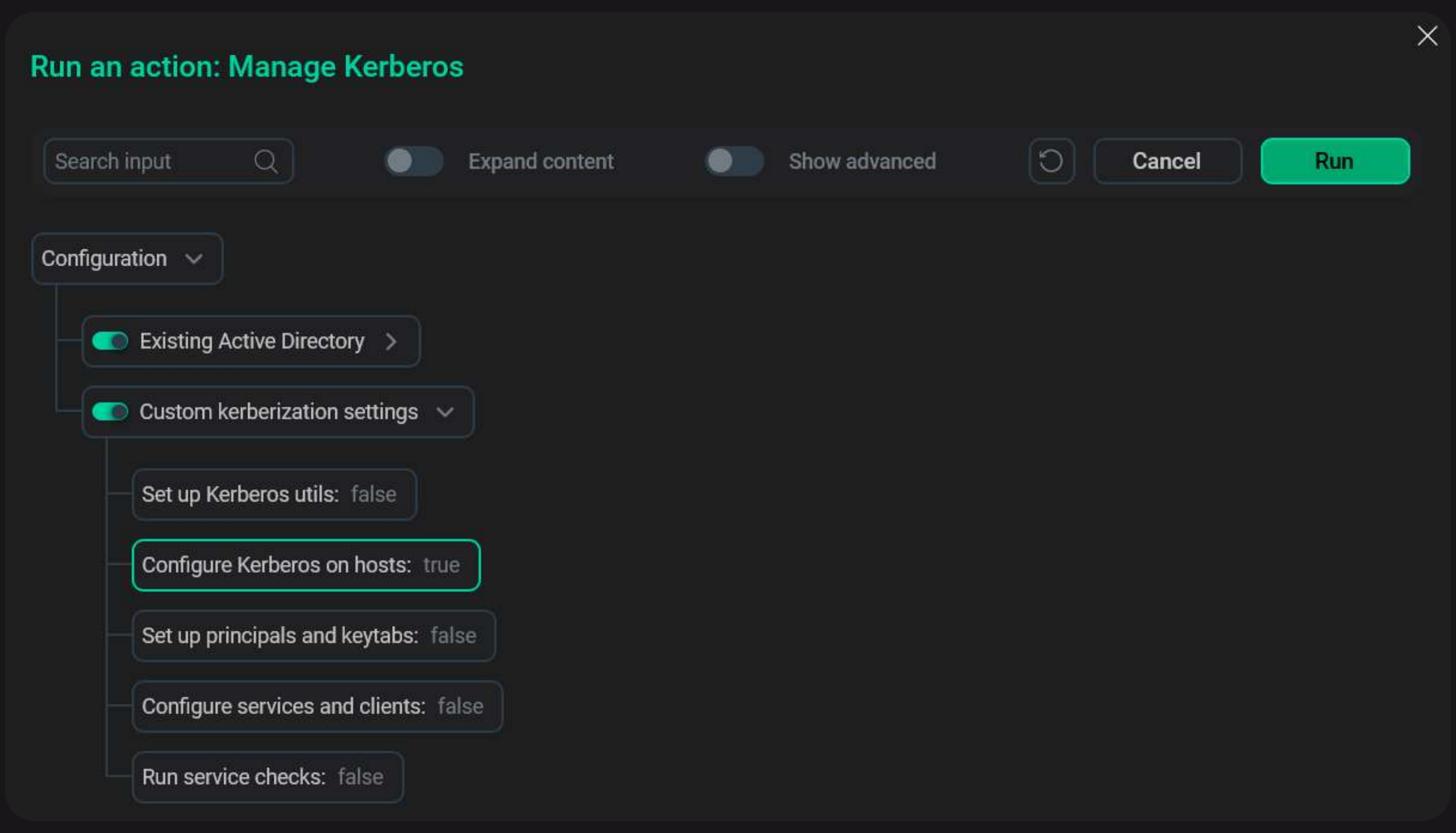
Contents

To Table of Contents

Prepare cluster hosts
Enable Kerberos

Prepare cluster hosts

1. Check that the ADH cluster is successfully installed and there are no issues.
2. Add the *krb5.conf* file to the */etc/* directory. You can do it manually, but another way is to launch the ***Manage Kerberos*** action with the ***Existing Active Directory*** option selected and the ***Configure Kerberos on hosts*** parameter enabled in the ***Custom kerberization settings*** menu. Make sure to provide non-admin credentials while filling out the ***Existing Active Directory*** form.



Run an action: Manage Kerberos

Configuration

Existing Active Directory

Custom kerberization settings

Set up Kerberos utils: false

Configure Kerberos on hosts: true

Set up principals and keytabs: false

Configure services and clients: false

Run service checks: false

Config file creation

3. Manually add principals for each service. You can use the ***Keytab*** column in the owner-keytab mapping table as a guide for which services you need to add a principal.

```
$ ldapadd -H <ldap_link> -x -D <username>@<realm> -w <password> -f <ldif_file>.ldif
```

where:

- `<ldap_link>` — link to the LDAP server.
- `<username>` — name of the LDAP user.
- `<realm>` — realm of the user.
- `<password>` — password for `<username>`.
- `<ldif_file>` — name of the file with principal information. An example of contents is presented below.

```
DN: CN=HTTP/<principal>,OU=Hadoop,DC=fresh,DC=com
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
accountExpires: 0
userPrincipalName: HTTP/<principal>@<realm>
servicePrincipalName: HTTP/<principal>
distinguishedName: CN=HTTP/<principal>,OU=Hadoop,DC=fresh,DC=com
userAccountControl: 514
```

where `<principal>` is the principal name for a service.

A successful principal addition should result in a message like the one below:

```
adding new entry "CN=HTTP/<principal>,OU=kerberos,ou=adh,DC=ad,DC=ranger-test"
```

To see a more detailed output, use the `-v` option with ldapadd which enables the verbose mode.

You can check the correctness of your principals using ldapsearch:

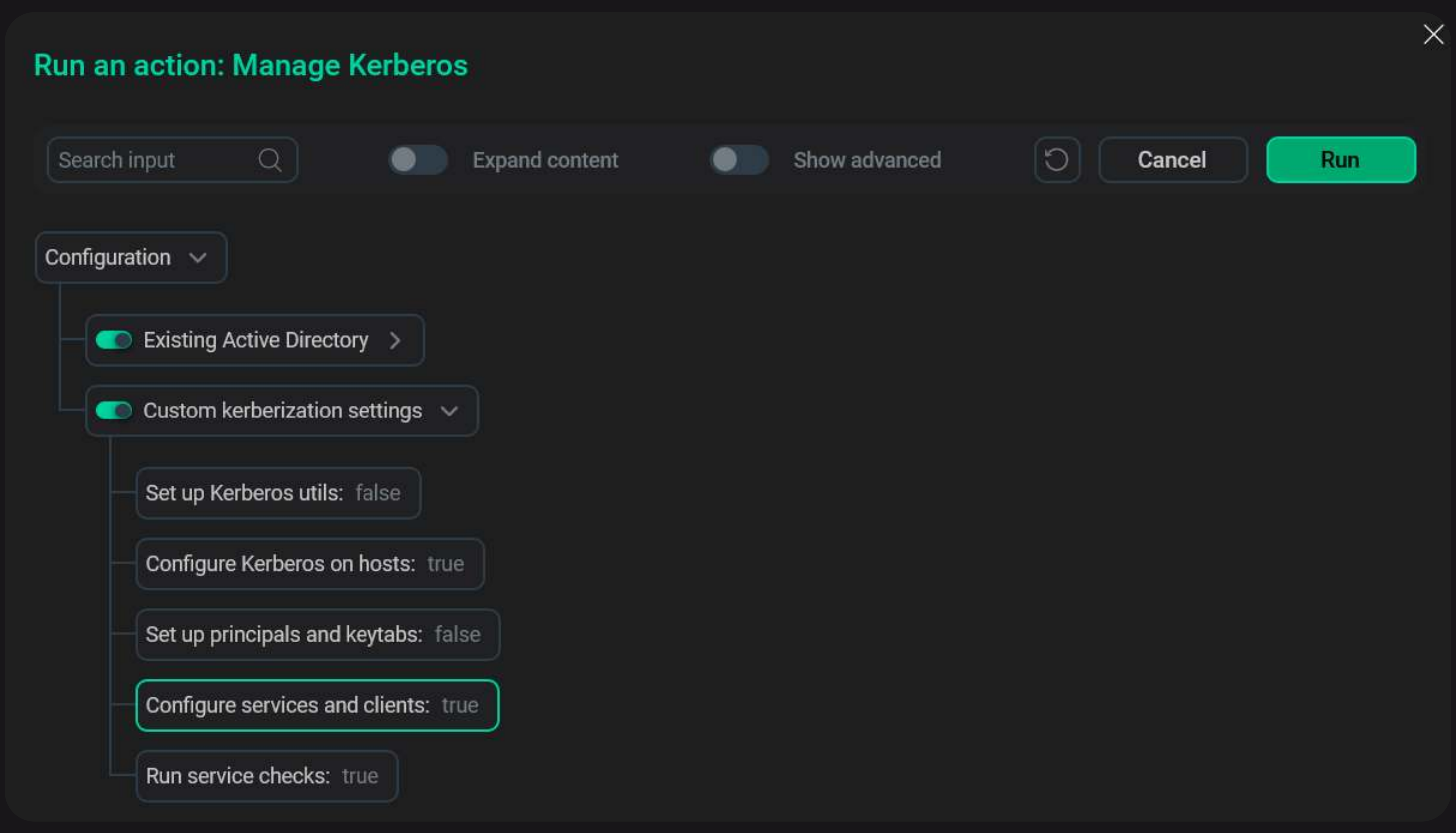
```
$ ldapsearch -x -H <ldap_link> -D <username>@<realm> -w <password> -b <search_base> <search_filter>
```

where:

- `<search_base>` — base for the search prompt.
- `<search_filter>` — search prompt.

Enable Kerberos

Run the ***Manage Kerberos*** action with the ***Existing Active Directory*** option selected and the ***Configure Kerberos on hosts***, ***Configure services and clients***, and ***Run Service checks*** parameters enabled.



Run an action: Manage Kerberos

Configuration

Existing Active Directory

Custom kerberization settings

Set up Kerberos utils: false

Configure Kerberos on hosts: true

Set up principals and keytabs: false

Configure services and clients: true

Run service checks: true

Finish kerberization

AD Kerberos common problems

Konstantin Alpashkin, Sergei Tikhomirov

Contents

To Table of Contents

- Error creating HTTP principal on host in AD domain
  - How to solve
  - Create an HTTP principal
- Principal already exists in another container
  - How to solve
  - Example
- Administrator has no permissions for the base_dn in the custom ldap.conf
  - How to solve

This article describes common issues that you may encounter during AD Kerberos configuration and their workarounds. For convenient visualization of an LDAP server contents, you can use Apache Directory Studio or any other suitable tool.

> **NOTE**
>
> In this section, a **container** denotes an LDAP entry called the base DN (search base) for service principals managed by the current user, that may contain the Organization Unit (OU), etc. In the Manage Kerberos window, this entry is specified in the ***Container DN*** field.

Some cases with possible solutions are listed below.

Error creating HTTP principal on host in AD domain

Such an error occurs during the Manage Kerberos action if an AD host has a security update KB5008382, given that the cluster hosts are in AD domain. You can find the typical error message below.

```
######################## STDOUT ########################
adding new entry "CN=HTTP/dl-dev-h3-kdc.adp.local,OU=svc,OU=EIAP,DC=adp,DC=local"

######################## STDERR ########################
ldap_add: Constraint violation (19)
        additional info: 000021C7: AtrErr: DSID-03200E89, #1:
        0: 000021C7: DSID-03200E89, problem 1005 (CONSTRAINT_ATT_TYPE), data 0, Att 90303 (servicePrincipalName)
```

How to solve

There are two workarounds:

- Remove hosts from the AD domain and return them after kerberization.
- Create an HTTP principal by AD Administrator before kerberization.

Create an HTTP principal

1. Authorize in AD server as an AD administrator with permissions to manage AD hosts. There are two ways to do that:
   - Log in to Windows AD server and go to ***AD Users and Computers***. Then, create a principal `HTTP/<fqdn>@<realm>` for each host.
   - Prepare the environment (see Useful tips), then log in the console with ldap-client and run the following command:

   ```
   ldapadd -H "ldaps://kru-win-new.fresh.com" -x -D administrator@FRESH.COM -w "mypassword" -f adduser.ldif
   ```

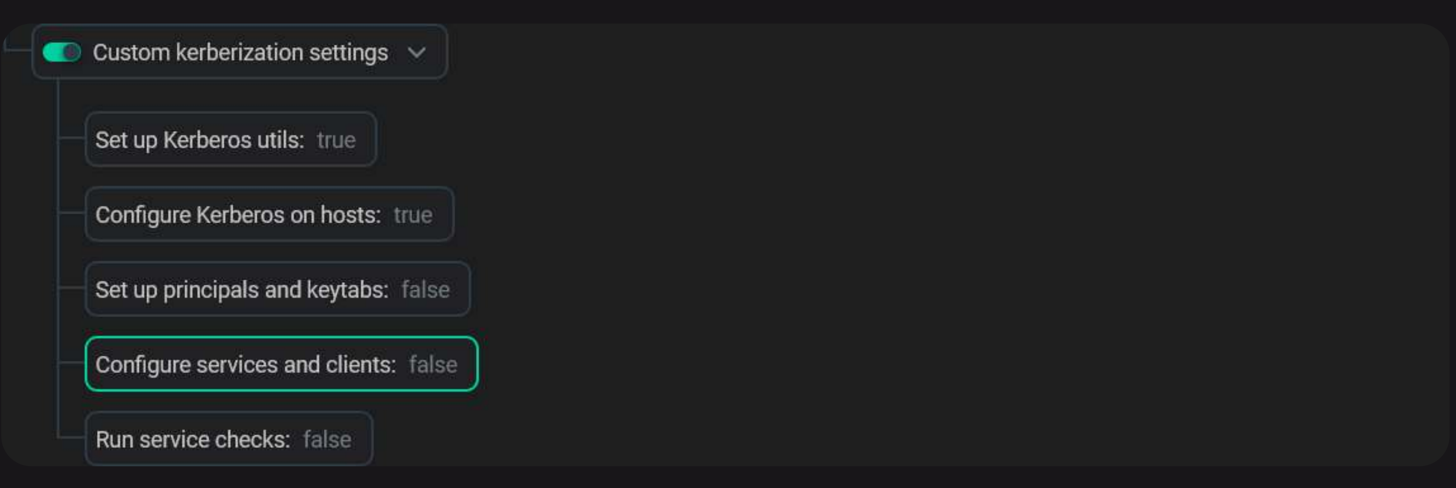
   Example contents of the *adduser.ldif* config file is presented below.

   ```
   DN: CN=HTTP/kru-test-adps.ru-central1.internal,OU=Hadoop,DC=fresh,DC=com
   changetype: add
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: user
   accountExpires: 0
   userPrincipalName: HTTP/kru-test-adps.ru-central1.internal@FRESH.COM
   servicePrincipalName: HTTP/kru-test-adps.ru-central1.internal
   distinguishedName: CN=HTTP/kru-test-adps.ru-central1.internal,OU=Hadoop,DC=fresh,DC=com
   userAccountControl: 514
   ```

2. Run the Manage Kerberos action. In case you don't have local admin permissions — disable the ***Set up principals and keytabs*** flag to skip their creation and make sure to provide all service principals and keytabs yourself.

Useful tips

- To prepare Kerberos environment on host, you can install ldap client with all cluster settings by using ***Manage Kerberos*** with the ***Custom kerberization settings*** option enabled, and ***Setup Kerberos Clients*** and ***Cluster configuration*** flags activated.



Custom kerberization settings
- Set up Kerberos utils: true
- Configure Kerberos on hosts: true
- Set up principals and keytabs: false
- Configure services and clients: false
- Run service checks: false

Customize action

- It's important to place HTTP principal in the same directory that you input into the ***Container DN*** field of the ***Manage Kerberos*** modal window, because the local admin should have permissions to control this principal.

This way, Kerberos will be enabled using the existing HTTP principal. The password of this principal will be generated and keytab will be written to all the hosts.

Principal already exists in another container

ADCM can not search/remove/re-create service principals in the containers that differ from the Container DN value specified while enabling/disabling Kerberos. Since the current administrator may not have permissions for all the AD domain, such principals that exist in different containers with the same name lead to a conflict:

```
######################## STDERR ########################
ldap_add: Constraint violation (19)
    additional info: 000021C8: AtrErr: DSID-03200BE8, #1:
    0: 000021C8: DSID-03200BE8, problem 1005 (CONSTRAINT_ATT_TYPE), data 0, Att 90290 (userPrincipalName).
```

How to solve

Remove existing service principals from the old container manually. An example on how to do this is demonstrated below.

Example

1. Search for existing principals in all the domain's containers, using the filter as shown below:

```
ldapsearch -H ldap://<ldap-server-url> -o ldif-wrap=no -x -W -D 'cn=userA,ou=kerberos,ou=adh,dc=foo,dc=bar' -b 'dc=foo,dc=bar'
name=hdfs/test-principal-name.ru-central1.internal | grep dn
```

   Where:

   - `-b 'dc=foo,dc=bar'` consists of the actual domain parts (e.g. 'dc=com,dc=hostname') corresponding to the Kerberos REALM.
   - `-o ldif-wrap=no` disables the ldapsearch results wrapping. Omitting this option truncates the results returned by ldapsearch so `grep` may not extract the full DN value required for removing principals.

   The ldapsearch returns container DN(s) like the ones below:

```
# Example 1:
dn: CN=hdfs/test-principal-name.ru-central1.internal,OU=e_ivanov_krb1_ou,OU=kerberos,OU=adh,DC=ad,DC=ranger-test

# Example 2:
dn: CN=del_me,CN=Users,DC=adh-sec,DC=com
```

2. Delete the existing conflicting principals. ADCM will re-create principals with an appropriate Container DN value. For example:

```
ldapdelete -H ldap://<ldap-server-url> -x -W -D 'cn=userA,ou=kerberos,ou=adh,dc=ad,dc=ranger-test' CN=hive/test-principal-name2.
ru-central1.internal,OU=userB,OU=kerberos,OU=adh,DC=ad,DC=ranger-test
```

3. After removing all the service principals from the container, run ***(Re)Enable Kerberos***. The action should complete without conflicts.

Administrator has no permissions for the base_dn in the custom ldap.conf

In this case, while expanding a component on the new host or while adding a new service, you may get the following error:

```
<timestamp> DEBUG adcm_check ansible adcm_check: title: Hdfs-Datanode: Add principal, title: <principal-name>,
message:######################## STDOUT ########################

######################## STDERR ########################
ldap_sasl_bind(SIMPLE): Can't contact LDAP server (-1)
```

How to solve

Check if `base_dn` in the custom *ldap.conf* corresponds to the actual Container DN value for the current user. If `base_dn` does not match, change the `base_dn` value in the custom *ldap.conf* or re-enable Kerberos with an appropriate container DN.

FreeIPA overview

Konstantin Alpashkin

To Table of Contents

FreeIPA is a free open source identity management system for Linux/UNIX environments. FreeIPA relies on Linux (Fedora), 389 Directory Server, MIT Kerberos, NTP, DNS, Dogtag (Certificate System) and its main purpose is to provide centralized account management and authentication.

To ensure security, Arenadata Hyperwave makes use of Kerberos, LDAP, SSL certificates/keys, and other related technologies. For such a toolkit, FreeIPA is a good fit for the identity management system role.

NOTE

FreeIPA is supported starting ADH 2.1.6.b1.

To enable FreeIPA for an ADH cluster you should use ADCM. During the enablement, ADCM installs IPA clients on each cluster node and connects the clients to the FreeIPA server using the provided FreeIPA Kerberos parameters.

In this tutorial, a sample single-noded FreeIPA server is used to demonstrate examples. This is a basic FreeIPA installation on CentOS (`yum install ipa-server`) with default installation parameters.

Configure Kerberos authentication based on FreeIPA via ADCM

Konstantin Alpashkin, Sergei Tikhomirov

Contents

To Table of Contents

Overview

FreeIPA Kerberos parameters

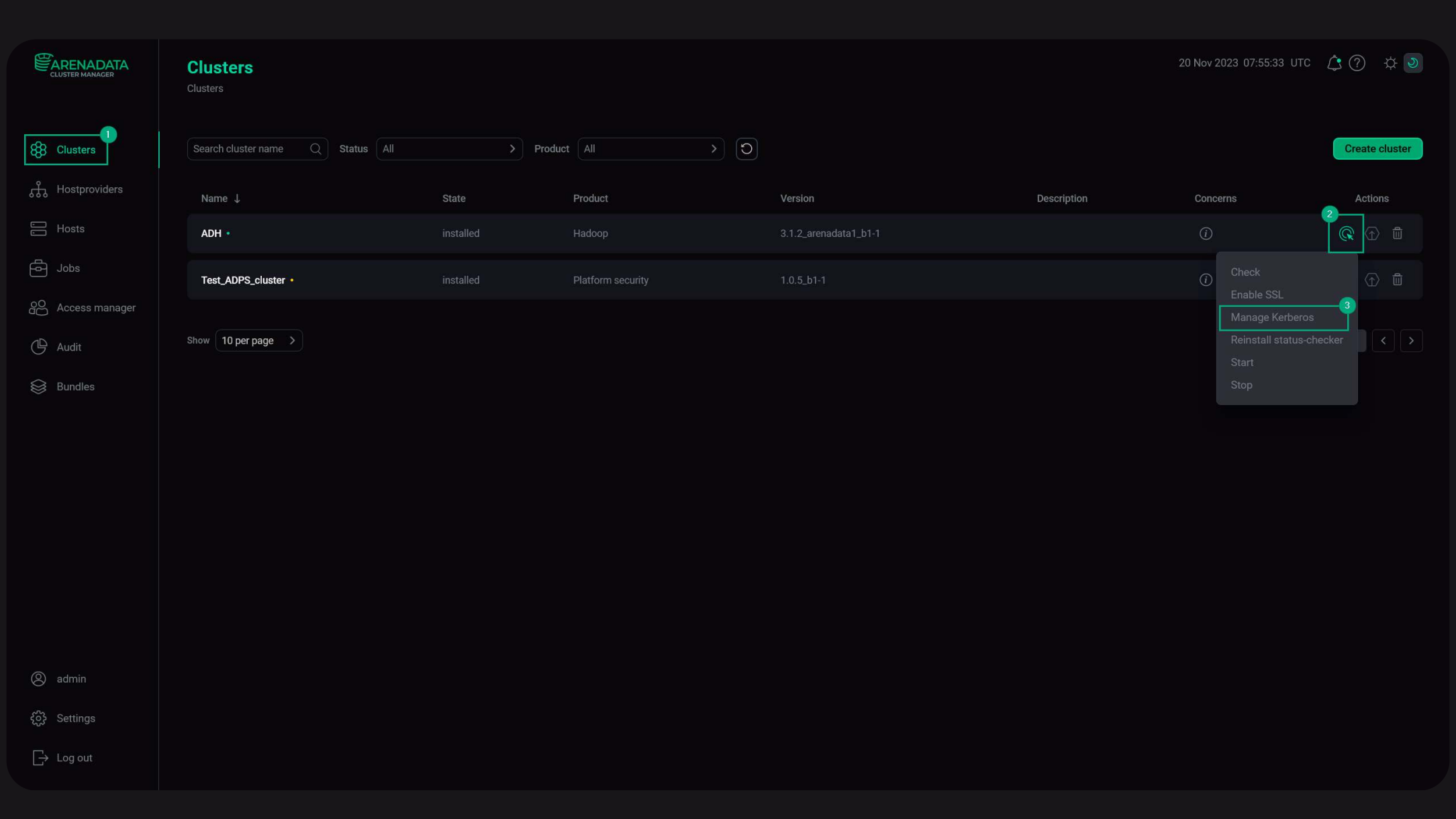
Check FreeIPA integration

- HDFS access before the cluster is kerberized
- HDFS access after enrolling FreeIPA

Overview

To kerberize a cluster using FreeIPA, follow the steps below:

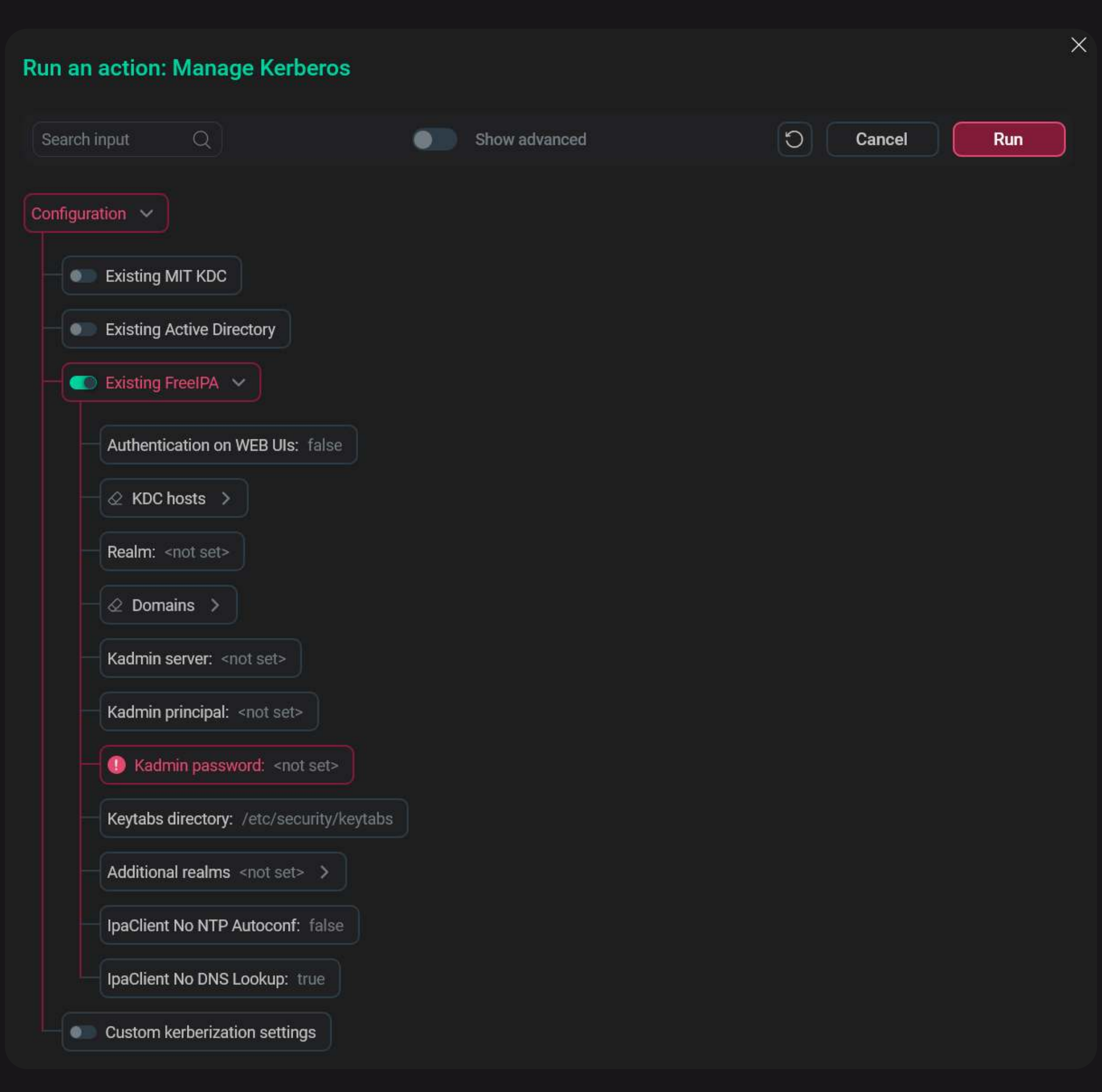
1. In ADCM web UI, go to the ***Clusters*** page. Select an installed and prepared ADH cluster, and run the ***Manage Kerberos*** action.



Name	State	Product	Version	Description	Concerns	Actions
ADH	installed	Hadoop	3.1.2_arenadata1_b1-1			
Test_ADPS_cluster	installed	Platform security	1.0.5_b1-1			

Manage Kerberos

2. In the pop-up window, turn on the ***Existing FreeIPA*** option.

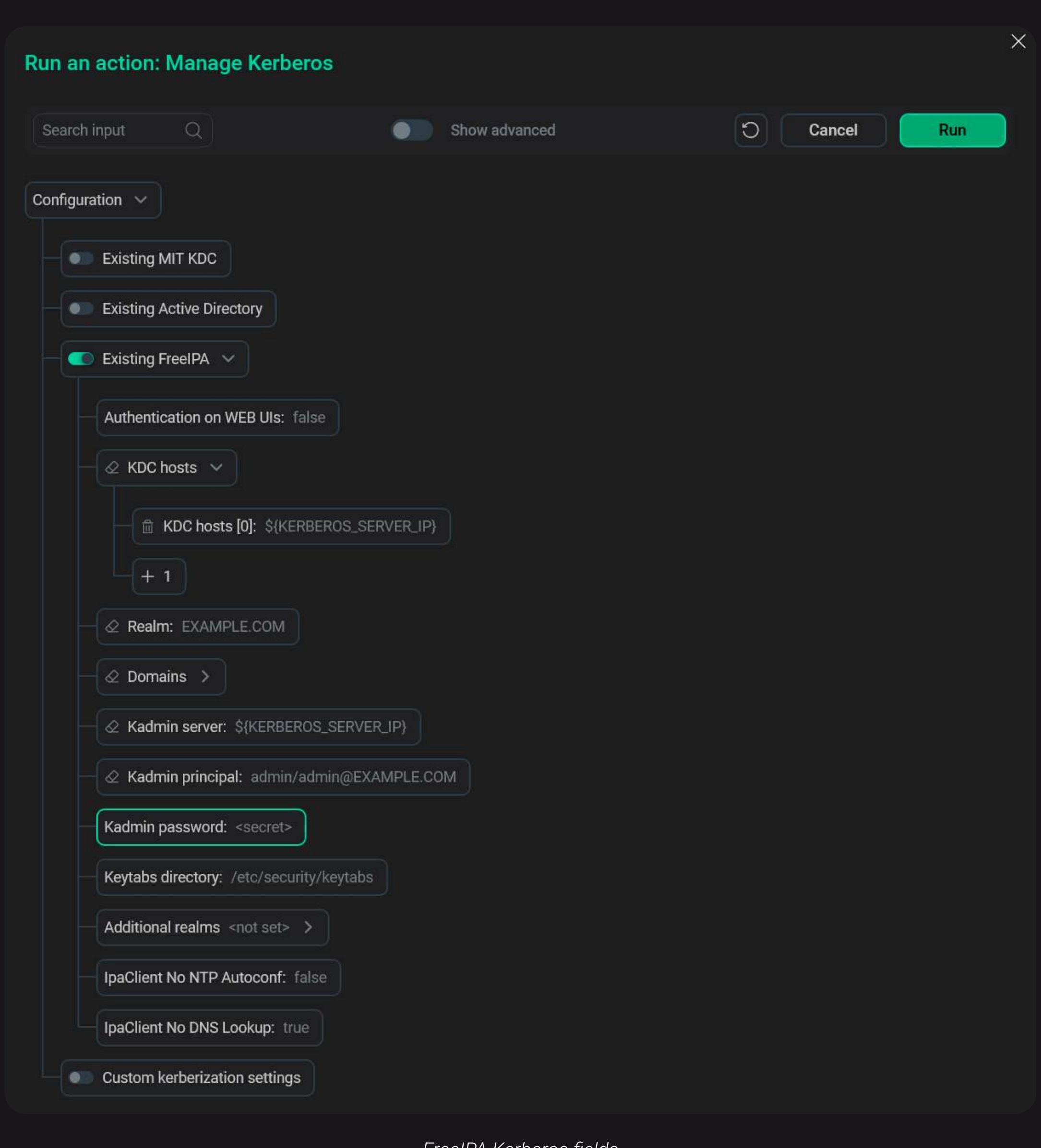


Run an action: Manage Kerberos

Configuration
- Existing MIT KDC
- Existing Active Directory
- Existing FreeIPA
  - Authentication on WEB UIs: false
  - KDC hosts
  - Realm: <not set>
  - Domains
  - Kadmin server: <not set>
  - Kadmin principal: <not set>
  - Kadmin password: <not set>
  - Keytabs directory: /etc/security/keytabs
  - Additional realms <not set>
  - IpaClient No NTP Autoconf: false
  - IpaClient No DNS Lookup: true
- Custom kerberization settings

Choose the relevant option

3. Fill in the FreeIPA Kerberos parameters.

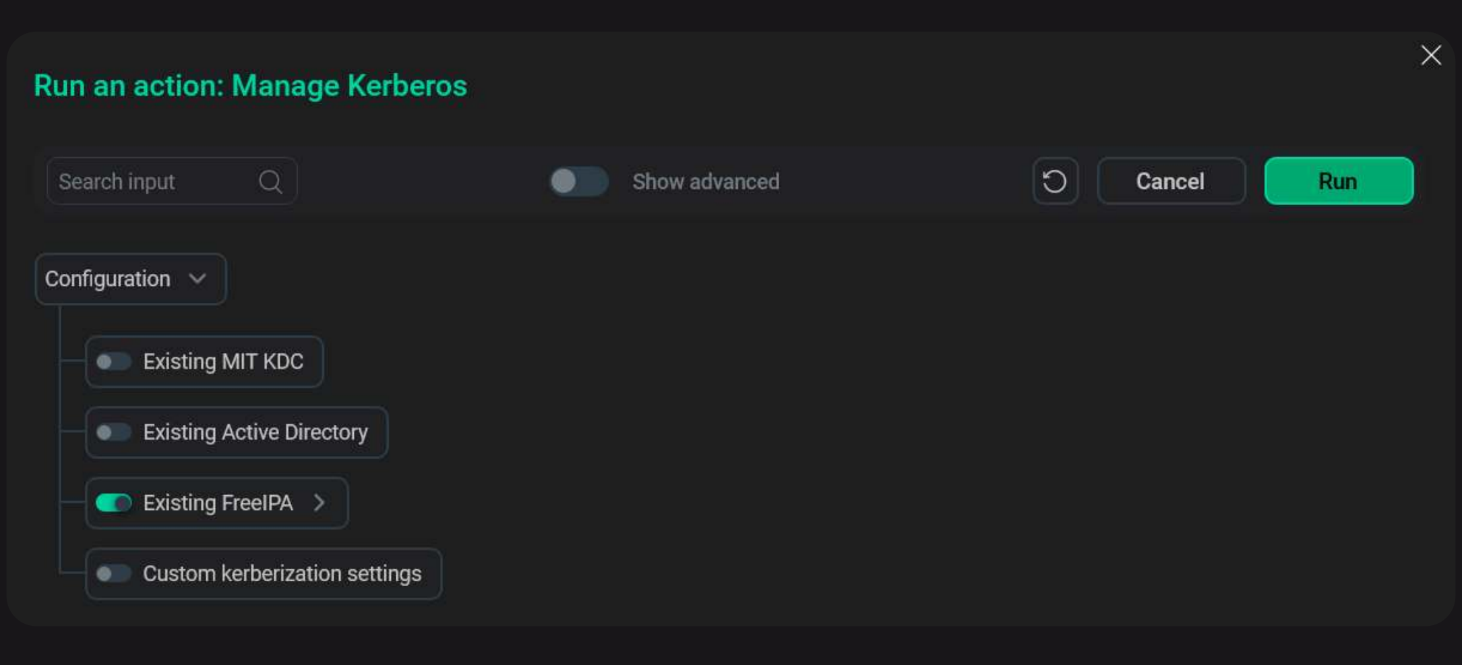


Run an action: Manage Kerberos

Configuration
- Existing MIT KDC
- Existing Active Directory
- Existing FreeIPA
  - Authentication on WEB UIs: false
  - KDC hosts
    - KDC hosts [0]: ${KERBEROS_SERVER_IP}
    - + 1
  - Realm: EXAMPLE.COM
  - Domains
  - Kadmin server: ${KERBEROS_SERVER_IP}
  - Kadmin principal: admin/admin@EXAMPLE.COM
  - Kadmin password: <secret>
  - Keytabs directory: /etc/security/keytabs
  - Additional realms <not set>
  - IpaClient No NTP Autoconf: false
  - IpaClient No DNS Lookup: true
- Custom kerberization settings

FreeIPA Kerberos fields

4. Click ***Run***, wait for the job to complete and proceed to setting up Kerberos in the cluster.



Run an action: Manage Kerberos

Configuration
- Existing MIT KDC
- Existing Active Directory
- Existing FreeIPA
- Custom kerberization settings

Run the action

FreeIPA Kerberos parameters

The following parameters are required to kerberize your ADH cluster with FreeIPA. You can get values for these by running `ipa user-find <ipa_admin>` on your FreeIPA server, where `<ipa_admin>` is your IPA Admin user.

Parameter	Description
Authentication on WEB UIs	Enables Kerberos authentication on Web UIs
KDC hosts	One or more KDC hosts with running FreeIPA server(s). Only FQDN is acceptable
Realm	A Kerberos realm to connect to the FreeIPA server
Domains	One or more domains associated with FreeIPA
Kadmin server	A host where `kadmin` is running. Only FQDN is acceptable
Kadmin principal	A principal name used to connect via `kadmin`, for example `admin@RU-CENTRAL1.INTERNAL`
Kadmin password	An IPA Admin password
Keytabs directory	Directory of the keytab file that contains one or several principals along with their keys
Additional realms	Additional Kerberos realms
IpaClient No NTP Autoconf	Disables the NTP configuration during the IPA client installation
IpaClient No DNS Lookup	Disables the DNS lookup for FreeIPA server during the IPA client installation
Number of retries for kinit invocation attempts	Number of retries for kinit invocation attempts

Check FreeIPA integration

After a successful kerberization, the hadoop commands can only run on cluster nodes after getting a Kerberos ticket. Below are examples that demonstrate cluster access attempts for an unsecured cluster and for a cluster kerberized with FreeIPA.

HDFS access before the cluster is kerberized

HDFS CLI

List the HDFS root contents:

```
$ hdfs dfs -ls /
```

The output is as follows:

```
Found 4 items
drwxrwxrwt   - yarn hadoop          0 2023-01-31 16:46 /logs
drwxr-xr-x   - hdfs hadoop          0 2023-01-31 16:43 /system
drwxrwxrwx   - hdfs hadoop          0 2023-01-31 21:46 /tmp
drwxr-xr-x   - hdfs hadoop          0 2023-01-31 17:00 /user
```

HDFS HttpFS

The curl way to list HDFS contents via HttpFS:

```
$ curl "http://ka-adh-2.ru-central1.internal:14000/webhdfs/v1/?op=LISTSTATUS&user=admin"
```

The response is as follows:

```
{"FileStatuses":{"FileStatus":
[{"pathSuffix":"logs","type":"DIRECTORY","length":0,"owner":"yarn","group":"hadoop","permission":"1777","accessTime":0,"modificationTime":1675183594965,"blockSize":0,"replication":0},...
```

HDFS access after enrolling FreeIPA

HDFS CLI

List the HDFS root contents:

```
$ hdfs dfs -ls /
```

Sample output (access denied):

```
2023-02-01 21:15:51,819 WARN ipc.Client: Exception encountered while connecting to the server : org.apache.hadoop.security.AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS]
2023-02-01 21:15:51,831 WARN ipc.Client: Exception encountered while connecting to the server : org.apache.hadoop.security.AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS]
2023-02-01 21:15:51,832 INFO retry.RetryInvocationHandler: java.io.IOException: DestHost:destPort ka-adh-3.ru-central1.internal:8020 , LocalHost:localPort ka-adh-1.ru-central1.internal/10.92.17.141:0. Failed on local exception: java.io.IOException: org.apache.hadoop.security.AccessControlException: Client cannot authenticate via:[TOKEN, KERBEROS], while invoking ClientNamenodeProtocolTranslatorPB.getFileInfo over ka-adh-3.ru-central1.internal/10.92.16.90:8020 after 1 failover attempts. Trying to failover after sleeping for 518ms
...
```

To pass the authentication, the UNIX user who submits the command `hdfs dfs …` (`admin` in this scenario) should obtain a corresponding Kerberos ticket.

```
$ kinit admin
```

After getting a ticket, the command `hdfs dfs -ls /` prints the HDFS root contents indicating that the authentication was successful:

```
Found 4 items
drwxrwxrwt   - yarn hadoop          0 2023-01-31 16:46 /logs
drwxr-xr-x   - hdfs hadoop          0 2023-01-31 16:43 /system
drwxrwxrwx   - hdfs hadoop          0 2023-01-31 21:46 /tmp
drwxr-xr-x   - hdfs hadoop          0 2023-01-31 17:00 /user
```

HDFS HttpFS

HDFS request:

```
$ curl "http://ka-adh-2.ru-central1.internal:14000/webhdfs/v1/?op=LISTSTATUS&user=admin"
```

Response (access denied):

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8"/>
<title>Error 401 Authentication required</title>
</head>
<body><h2>HTTP ERROR 401</h2>
<p>Problem accessing /webhdfs/v1/. Reason:
<pre>    Authentication required</pre></p>
</body>
</html>
```

To pass the authentication, you have to get a Kerberos ticket for the user who runs curl (`admin` in this scenario):

```
$ kinit admin
```

HDFS request:

```
$ curl --negotiate -u : "http://ka-adh-2.ru-central1.internal:14000/webhdfs/v1/?op=LISTSTATUS&user=admin"
```

> TIP
>
> The `--negotiate` flag enables SPNEGO to allow curl access to a Kerberos-protected resource.

Response (successful):

```
{"FileStatuses":{"FileStatus":
[{"pathSuffix":"logs","type":"DIRECTORY","length":0,"owner":"yarn","group":"hadoop","permission":"1777","accessTime":0,"modificationTime":1675183594965,"blockSize":0,"replication":0},
{"pathSuffix":"system","type":"DIRECTORY","length":0,"owner":"hdfs","group":"hadoop","permission":"755","accessTime":0,"modificationTime":1675183407542,"blockSize":0,"replication":0},...
```

Enable Kerberos with prepared FreeIPA server and no admin permissions

Sergei Tikhomirov

Contents

To Table of Contents

Create an IPA user with no admin permissions

Prepare cluster hosts

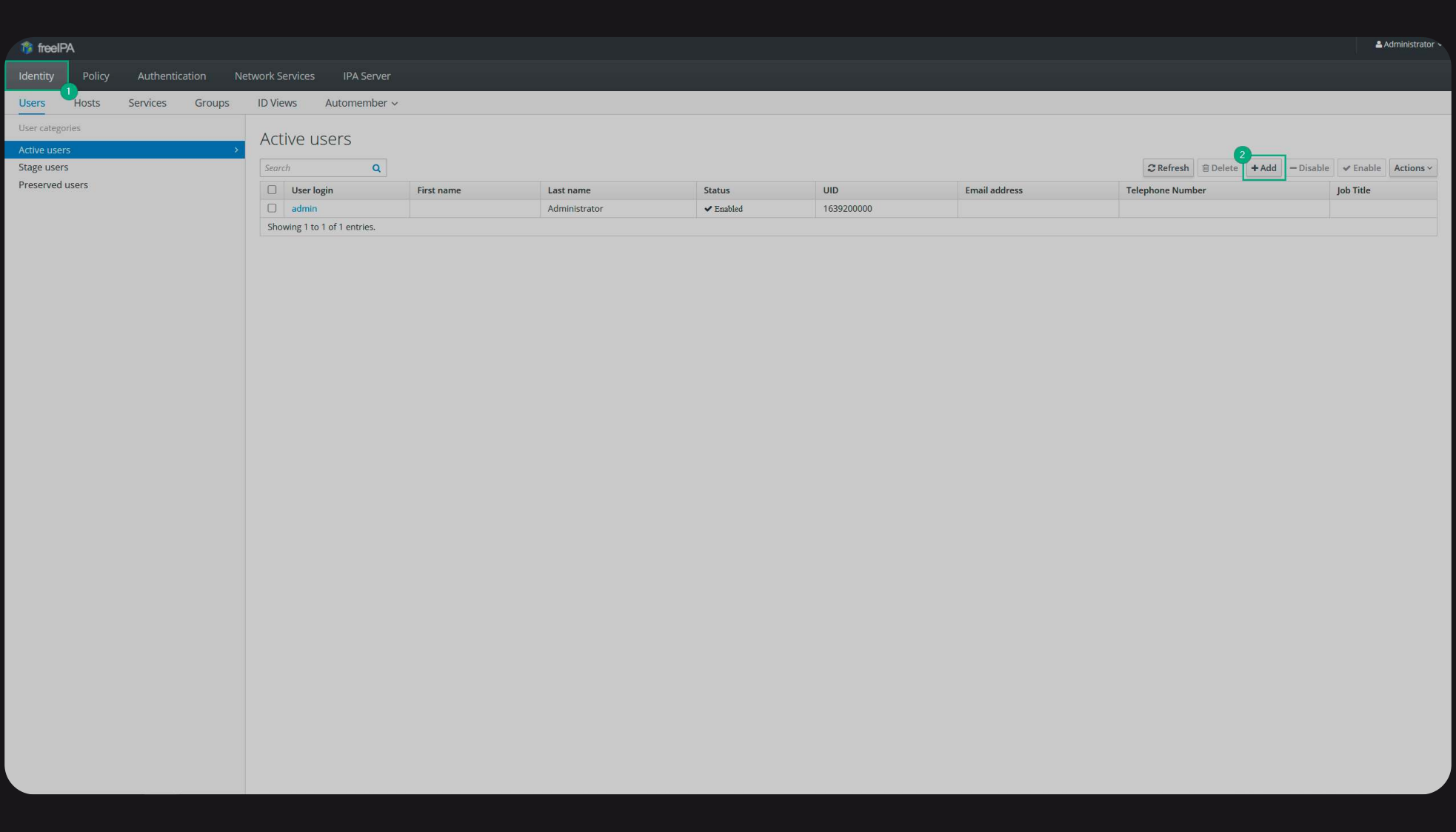
Enable Kerberos

Additional information

Create an IPA user with no admin permissions

If you don't have a non-admin IPA user, you can create one in FreeIPA UI. To do that, follow the steps below:

1. Login to your IPA server as admin, go to the ***Identity*** tab, and click ***Add***.

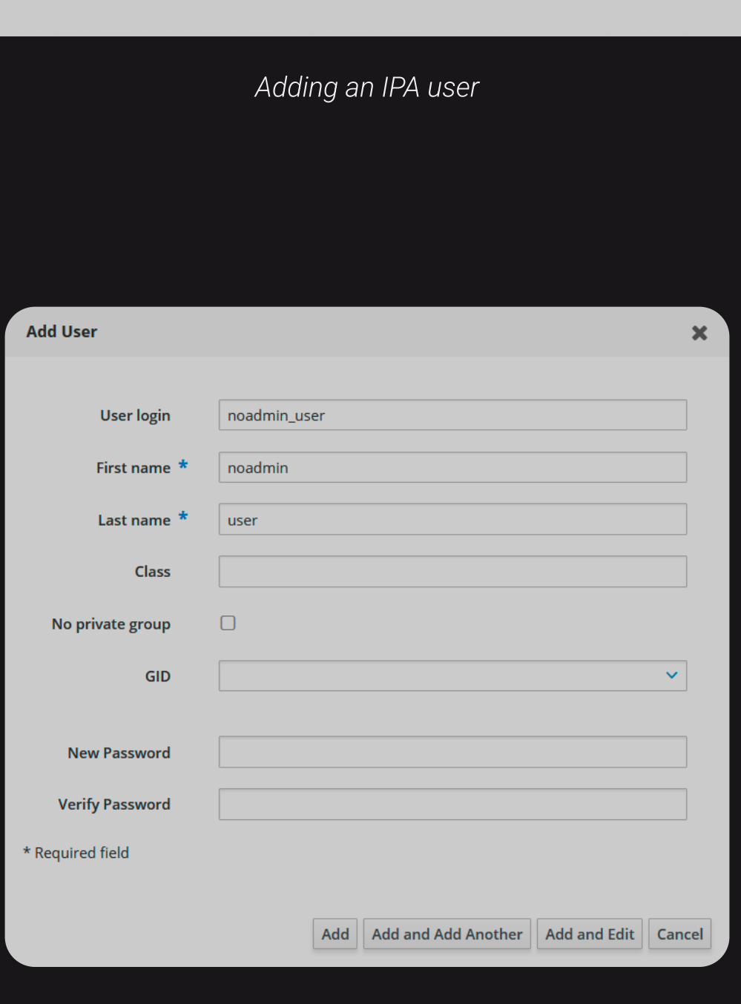

freeIPA
Identity
Policy
Authentication
Network Services
IPA Server
Users
Hosts
Services
Groups
ID Views
Automember
User categories
Active users
Stage users
Preserved users
Active users
Refresh
Delete
Add
Disable
Enable
Actions
1
2

User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
admin		Administrator	✓ Enabled	1639200000			

Showing 1 to 1 of 1 entries.

Adding an IPA user

2. Fill in the form in the pop-up window.



Add User

User login: noadmin_user

First name *: noadmin

Last name *: user

Class:

No private group: ☐

GID:

New Password:

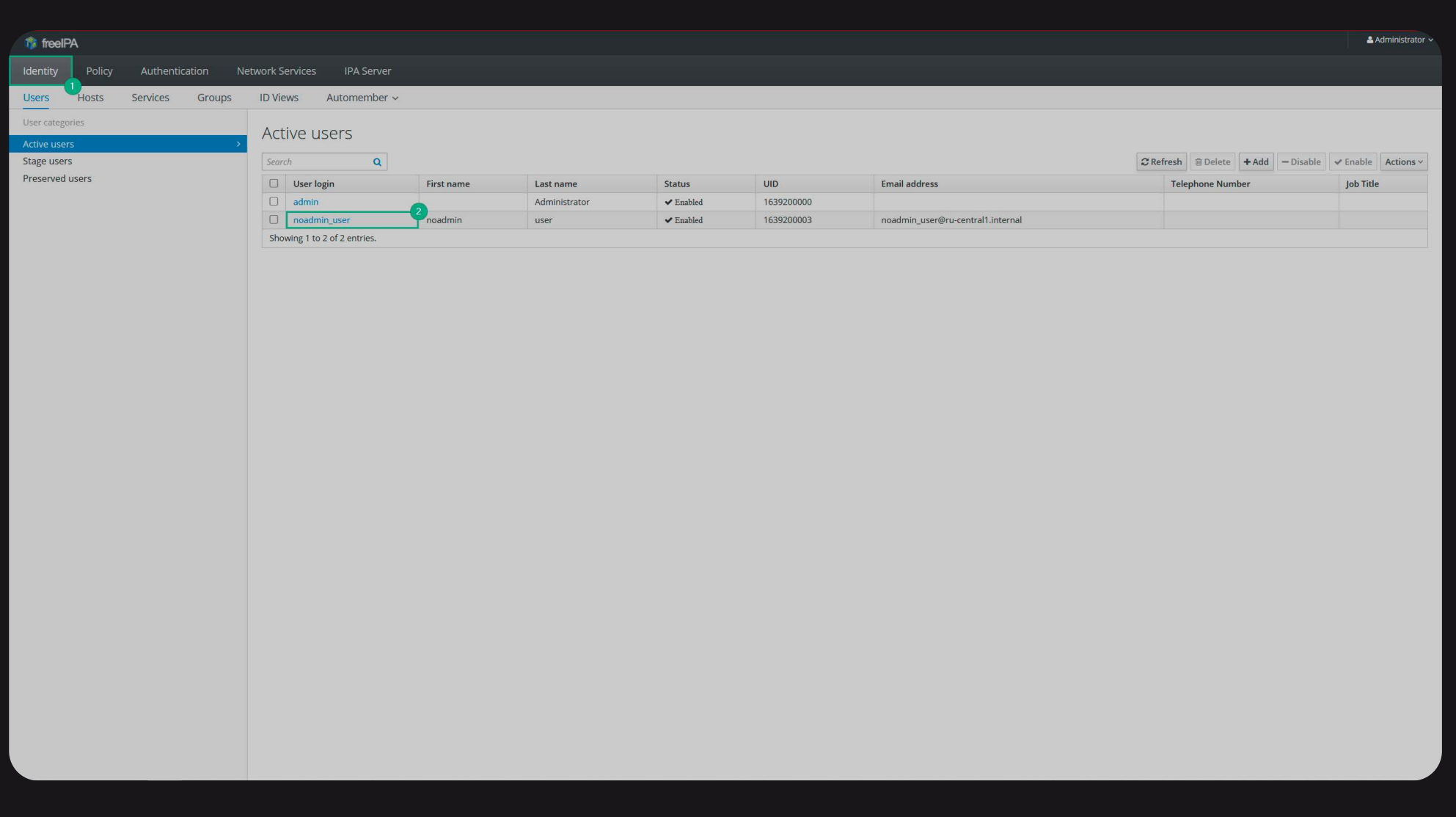
Verify Password:

* Required field

Add
Add and Add Another
Add and Edit
Cancel

IPA user credentials

3. Set a password for the user:
   - On the ***Identity*** tab, click on the new user in the ***User login*** column.

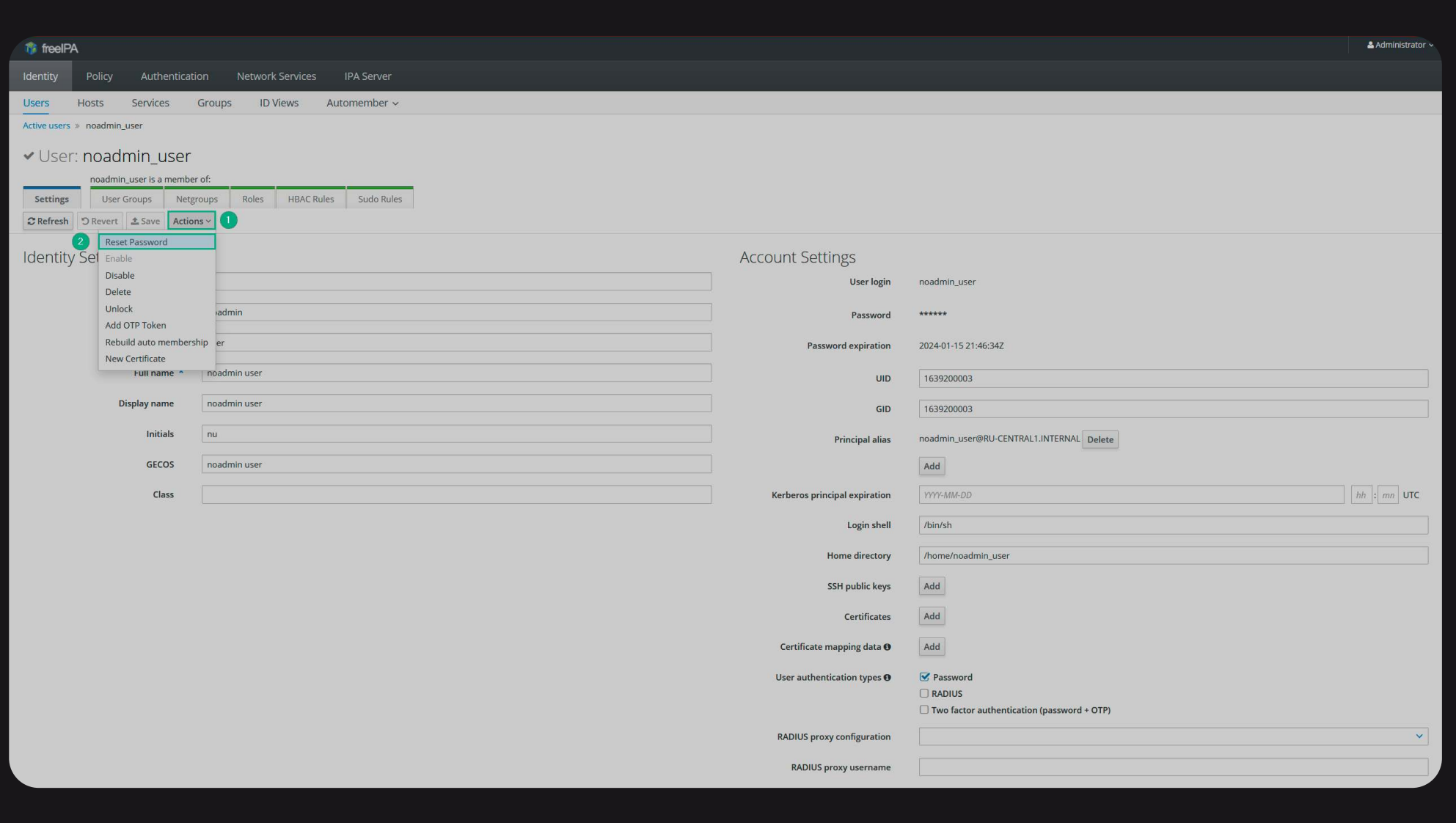

freeIPA
Identity
Policy
Authentication
Network Services
IPA Server
Users
Hosts
Services
Groups
ID Views
Automember
Active users
Stage users
Preserved users
Refresh
Delete
Add
Disable
Enable
Actions
1
2

User login	First name	Last name	Status	UID	Email address	Telephone Number	Job Title
admin		Administrator	✓ Enabled	1639200000			
noadmin_user	noadmin	user	✓ Enabled	1639200003	noadmin_user@ru-central1.internal		

Showing 1 to 2 of 2 entries.

IPA users

   - Click ***Actions*** → ***Reset password***.


freeIPA
Identity
Policy
Authentication
Network Services
IPA Server
Users
Hosts
Services
Groups
ID Views
Automember
Active users
User: noadmin_user
Settings
User Groups
Netgroups
Roles
HBAC Rules
Sudo Rules
Refresh
Revert
Save
Actions
1
2
Reset Password
Enable
Disable
Delete
Unlock
Add OTP Token
Rebuild auto membership
New Certificate
Identity Settings
Account Settings

User: noadmin_user

noadmin_user is a member of:

Full name: noadmin user

Display name: noadmin user

Initials: nu

GECOS: noadmin user

Class:

User login: noadmin_user

Password: ******

Password expiration: 2024-01-15 21:46:34Z

UID: 1639200003

GID: 1639200003

Principal alias: noadmin_user@RU-CENTRAL1.INTERNAL

Kerberos principal expiration: YYYY-MM-DD UTC

Login shell: /bin/sh

Home directory: /home/noadmin_user

User authentication types: ☑ Password ☐ RADIUS ☐ Two factor authentication (password + OTP)

RADIUS proxy configuration:

RADIUS proxy username:

Setting a password

In the pop-up window, enter the new password and confirm it.

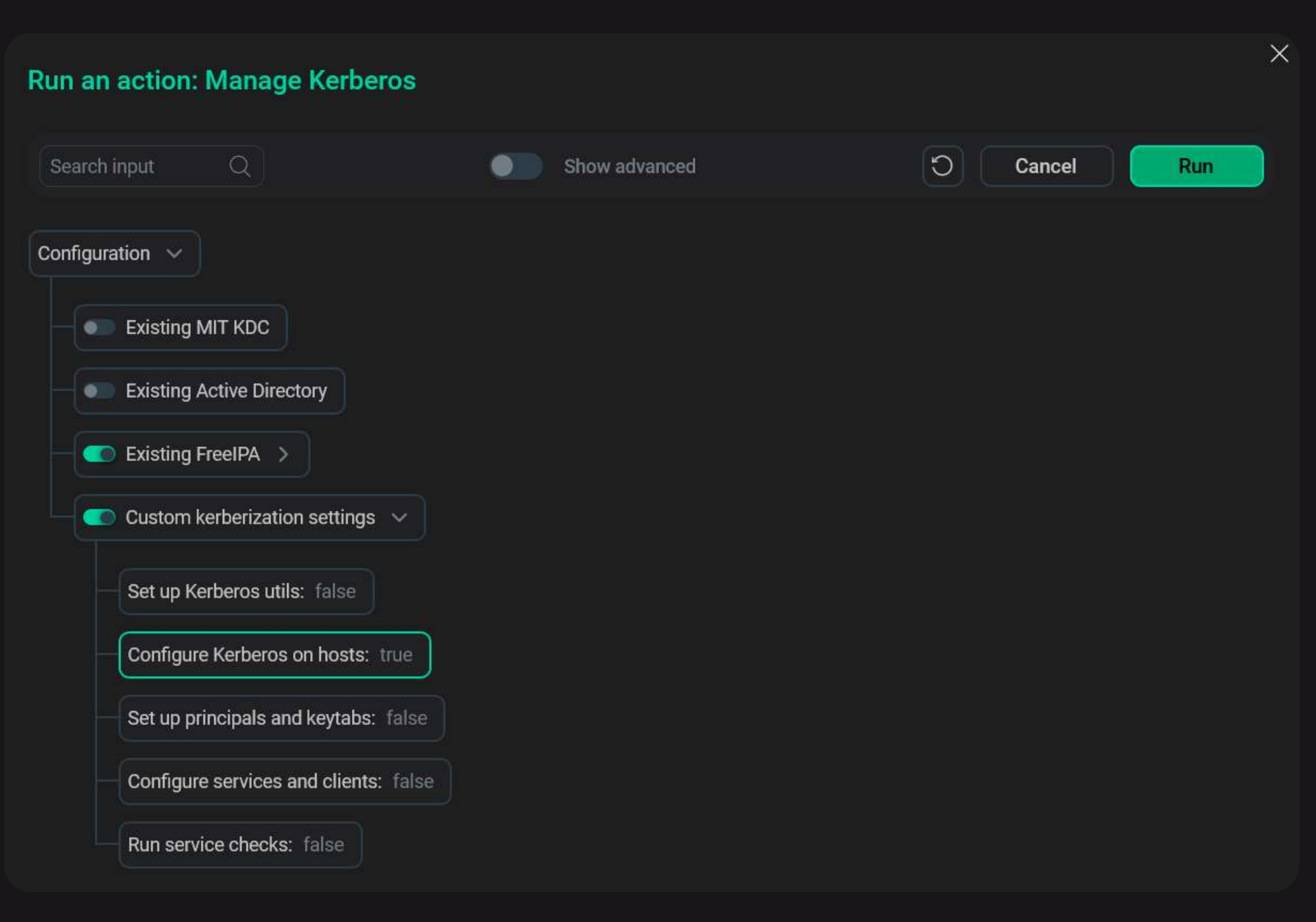
Prepare cluster hosts

IMPORTANT

Note that the steps 2—8 can be done manually, but it's highly recommended to use the script below to complete them automatically.

Do the following actions on each cluster host:

1. Add the *krb5.conf* file to the */etc/* directory. You can do it manually, but another way is to launch the Manage Kerberos action with ***Existing FreeIPA*** option and the ***Configure Kerberos on hosts*** parameter enabled in the ***Custom kerberization settings*** menu. Make sure to provide non-admin credentials while filling out the ***Existing FreeIPA*** form.


Run an action: Manage Kerberos
Search input
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
Existing Active Directory
Existing FreeIPA
Custom kerberization settings

Set up Kerberos utils: false

Configure Kerberos on hosts: true

Set up principals and keytabs: false

Configure services and clients: false

Run service checks: false

Config file creation

2. Install the FreeIPA client by calling the command below:

```
$ sudo ipa-client-install -U -p <ipa_user> -w <ipa_pass> --server=<ipa_server> --domain=<domain> --realm=<realm>
```

where:

- `<ipa_user>` — non-admin username.
- `<ipa_pass>` — password for `<ipa_user>`.
- `<ipa_server>` — FQDN of a FreeIPA server.
- `<domain>` — FreeIPA domain.
- `<realm>` — FreeIPA realm.

3. Call `kinit` with non-admin credentials:

```
$ kinit <username>
```

where `<username>` is a username for IPA user with no admin permissions.

4. Add services to IPA. You can use the ***Keytab*** column in the owner-keytab mapping as a list of services you need to add.

```
$ ipa service-add <service_name>/<fqdn>
```

where:

- `<service_name>` — name of a service component.
- `<fqdn>` — FQDN of the host you launch the command on.

5. Add keytabs for each service and locate them in */etc/security/keytabs*. You can use the ***Keytab*** column in the owner-keytab mapping as a list of keytabs you need to add.

```
$ ipa-getkeytab -p <service_name>/<fqdn> -k <service_name>.service.keytab
```

where:

- `<service_name>` — name of a service component.
- `<fqdn>` — FQDN of the host you launch the command on.

6. Add a user for each serivce if it doesn't exist yet. You can use the ***Owner user*** column in the owner-keytab mapping as a list of services you need to add.

```
$ adduser <service_name>
```

where `<service_name>` is a name of a service.

7. Change the owner for each keytab. You can use the owner-keytab mapping as guidance.

```
$ chown <user>:<group> <keytab_name>
```

where:

- `<user>` — owner username.
- `<group>` — owner group.
- `<keytab_name>` — name of a keytab file.

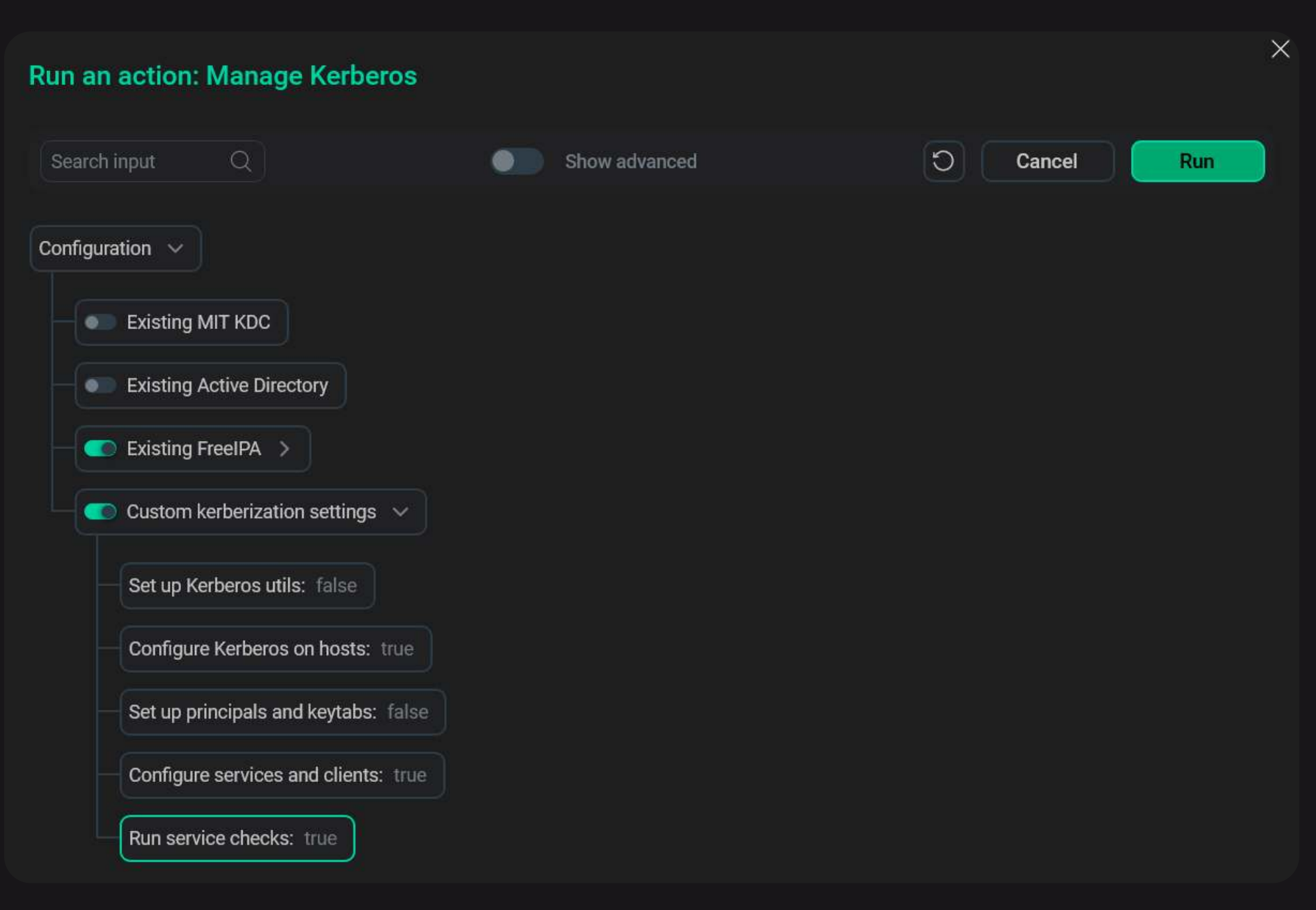
8. Set additional permissions (640) for Spark, HDFS, HBase, Hive, HTTP, and YARN service keytabs:

```
$ chmod 640 <keytab_name>
```

where `<keytab_name>` is a name of a keytab file.

Enable Kerberos

Run the ***Manager Kerberos*** action with FreeIPA user credentials. Also, disable the ***Set up Kerberos utils*** and ***Set up principals and keytabs*** parameters in the ***Custom kerberization settings*** menu.


Run an action: Manage Kerberos
Search input
Show advanced
Cancel
Run
Configuration
Existing MIT KDC
Existing Active Directory
Existing FreeIPA
Custom kerberization settings

Set up Kerberos utils: false

Configure Kerberos on hosts: true

Set up principals and keytabs: false

Configure services and clients: true

Run service checks: true

Enable Kerberos

Additional information

Owner-keytab mapping

Owner user	Owner group	Keytab	Permissions
airflow	airflow	airflow.service.keytab	600
flink	flink	flink.service.keytab	600
hbase	hadoop	hbase.service.keytab	640
	hbase	hbase-thrift2.service.keytab	600
hdfs	hadoop	HTTP.service.keytab	640
		hdfs-datanode.service.keytab	600
		hdfs-journalnode.service.keytab	600
		hdfs-namenode.service.keytab	600
		hdfs-zkfc.service.keytab	600
		hdfs.service.keytab	640
hive	hive	hive.service.keytab	640
httpfs	httpfs	httpfs.service.keytab	600
livy	hadoop	livy.service.keytab	600
mapred	hadoop	mapreduce-historyserver.service.keytab	600
phoenix	phoenix	hbase-phoenix_queryserver.service.keytab	600
solr	solr	solr.service.keytab	600
spark	spark	spark.service.keytab	640
yarn	hadoop	yarn-nodemanager.service.keytab	600
		yarn-resourcemanager.service.keytab	600
		yarn-timelineserver.service.keytab	600
		yarn.service.keytab	640
zeppelin	zeppelin	zeppelin.service.keytab	600
zookeeper	zookeeper	zookeeper.service.keytab	600
impala	impala	impala.service.keytab	600

Host preparation script

```
#!/bin/bash

# Script for adding services and keytabs to Cluster hosts
# Edit NUMHOSTS and HOSTS as it will be suitable for your case.
# EDIT your IPA parameters
# Select services according to the bundle

declare -a NUMHOSTS
declare -a HOSTS

SSH_OPTS='-o GlobalKnownHostsFile=/dev/null -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'
NUMHOSTS=(adh3-1 adh3-2 adh3-3)
HOSTS=$(for i in "${NUMHOSTS[@]}"; do echo "stikhomirov-$i.ru-central1.internal"; done)

#IPA credentials
ipa_user='noadmin_username'
ipa_pass='12345678'
ipa_server='stikhomirov-freeipa.ru-central1.internal'
domain='ru-central1.internal'
realm='RU-CENTRAL1.INTERNAL'

#ADH services
services=(airflow flink hbase-phoenix_queryserver hbase-thrift2 hbase hdfs-datanode hdfs-journalnode hdfs-namenode hdfs-zkfc hdfs hive httpfs livy mapreduce-historyserver solr spark sqoop yarn-nodemanager yarn-resourcemanager yarn-timelineserver yarn zeppelin zookeeper impala)
user-groups=(airflow:airflow flink:flink hbase:hadoop hbase:hbase hdfs:hadoop hdfs:hadoop hdfs:hadoop hdfs:hadoop hdfs:hadoop hdfs:hadoop hive:hive httpfs:httpfs livy:hadoop mapred:hadoop phoenix:phoenix solr:solr spark:spark sqoop:sqoop yarn:hadoop yarn:hadoop yarn:hadoop yarn:hadoop zeppelin:zeppelin zookeeper:zookeeper impala:impala)
users=(airflow flink hbase hdfs hive httpfs impala livy mapred phoenix solr spark sqoop yarn zeppelin zookeeper)
rwr_services=(spark hdfs hbase hive yarn HTTP)

for HOST in $HOSTS; do
    echo "Installing FreeIPA client"
    ssh $SSH_OPTS $HOST sudo ipa-client-install -U -p $ipa_user -w $ipa_pass --server=$ipa_server --domain=$domain --realm=$realm

    echo "Creating keytab directory"
          ssh $SSH_OPTS $HOST sudo mkdir /etc/security/keytabs

        echo "Launching kinit on $HOST"
          ssh $SSH_OPTS $HOST sudo kinit $ipa_user <<< $ipa_pass

    for user in ${users[@]}; do
        echo "Adding $user user"
            ssh $SSH_OPTS $HOST sudo adduser $user
    done

    for service in ${!services[@]}; do
        echo "Adding $service service to IPA"
            ssh $SSH_OPTS $HOST sudo ipa service-add ${services[i]}/$HOST
        echo "Get $service keytab"
            ssh $SSH_OPTS $HOST sudo ipa-getkeytab -p ${services[i]}/$HOST -k /etc/security/keytabs/${services[i]}.service.keytab
        echo "Changing owner for $service keytab"
            ssh $SSH_OPTS $HOST sudo chowner ${user-groups[i]} /etc/security/keytabs/${services[i]}.service.keytab
    done

    for service in ${rwr_services[@]}; do
        echo "Setting 640 permissions for $service"
            ssh $SSH_OPTS $HOST sudo chmod 640 /etc/security/keytabs/$service.service.keytab
    done
done
```

Configure Kerberos authentication based on Samba via ADCM

Sergei Tikhomirov

Contents

To Table of Contents

Overview

Kerberization in ADCM

Samba Kerberos parameters

Overview

Samba acts as an Active Directory domain controller, so its configuration is very similar to that of Active Directory.

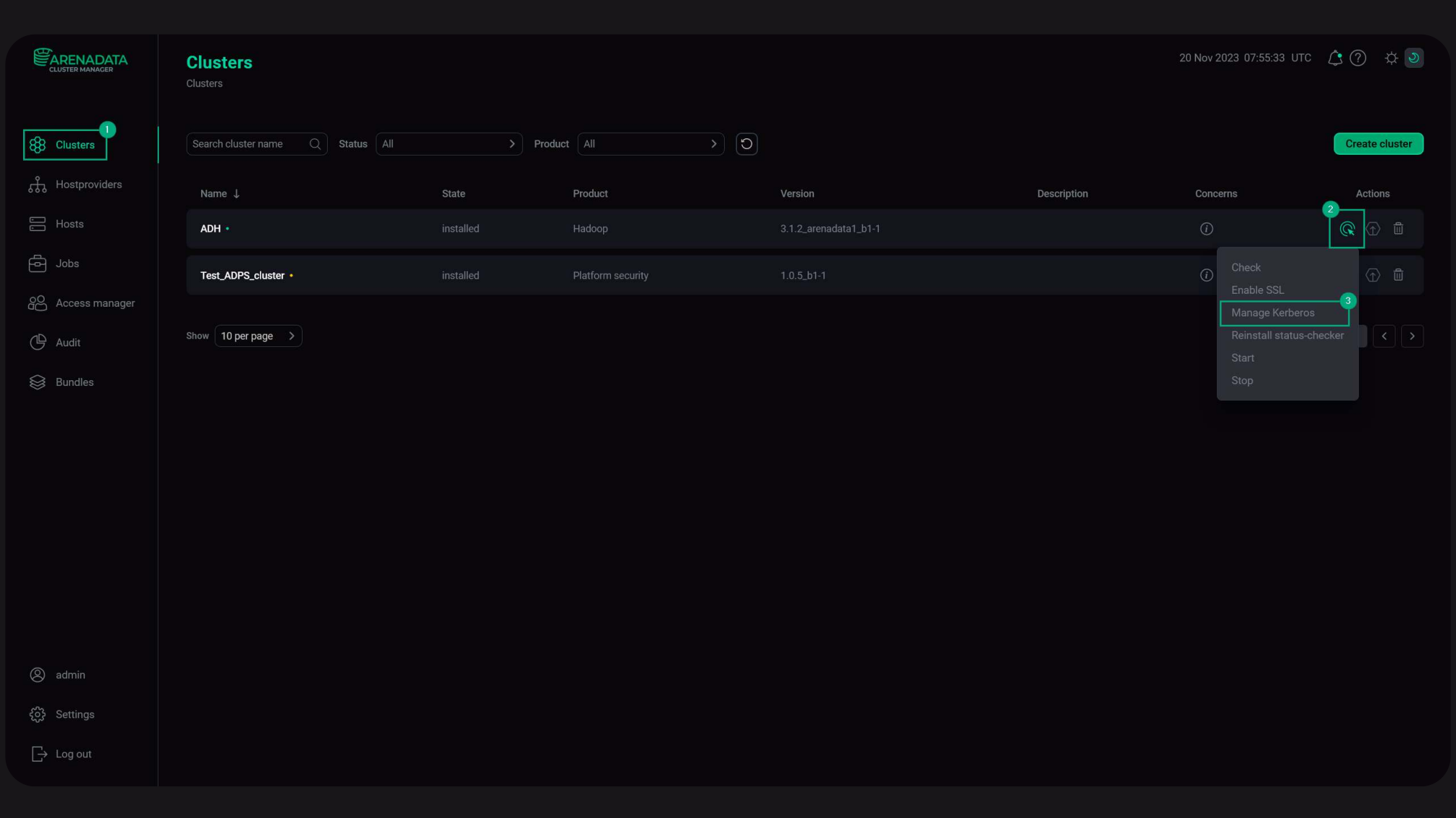
CentOS 7 specifics

On CentOS 7, the bundle doesn't handle the keytab generation because a standard Samba package for CentOS 7 doesn't support creating principals on a remote domain controller. Hence, you need to prepare the keytabs yourself. For that, you can edit the variables in the sample script provided below and use it. On other systems, the generation is automatic.

Kerberization in ADCM

To kerberize a cluster using Samba, follow the steps below:

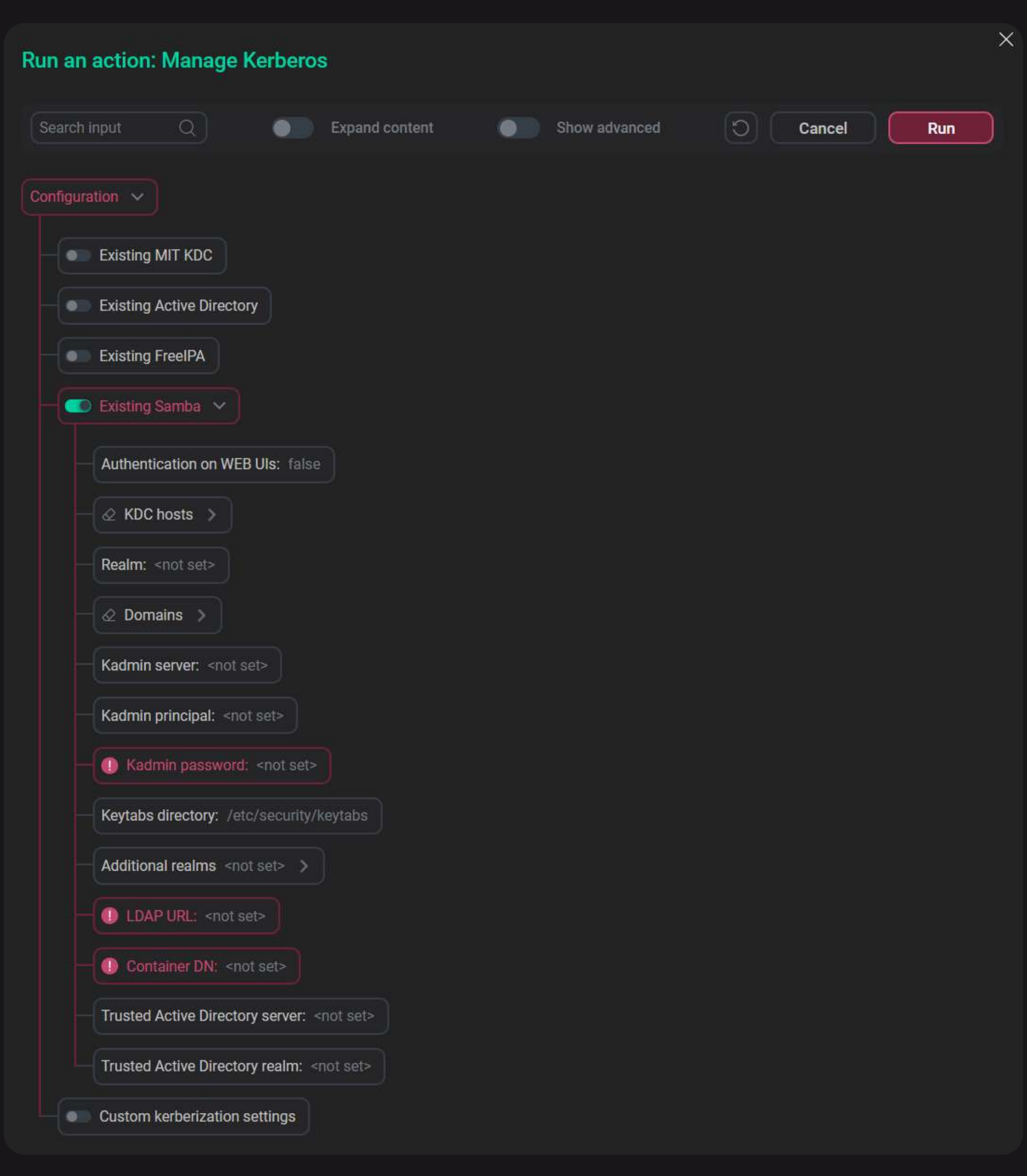
1. In the ADCM web UI, go to the ***Clusters*** page. Select an installed and prepared ADH cluster, and run the ***Manage Kerberos*** action.



Name	State	Product	Version	Description	Concerns	Actions
ADH	installed	Hadoop	3.1.2_arenadata1_b1-1			
Test_ADPS_cluster	installed	Platform security	1.0.5_b1-1			

Manage Kerberos

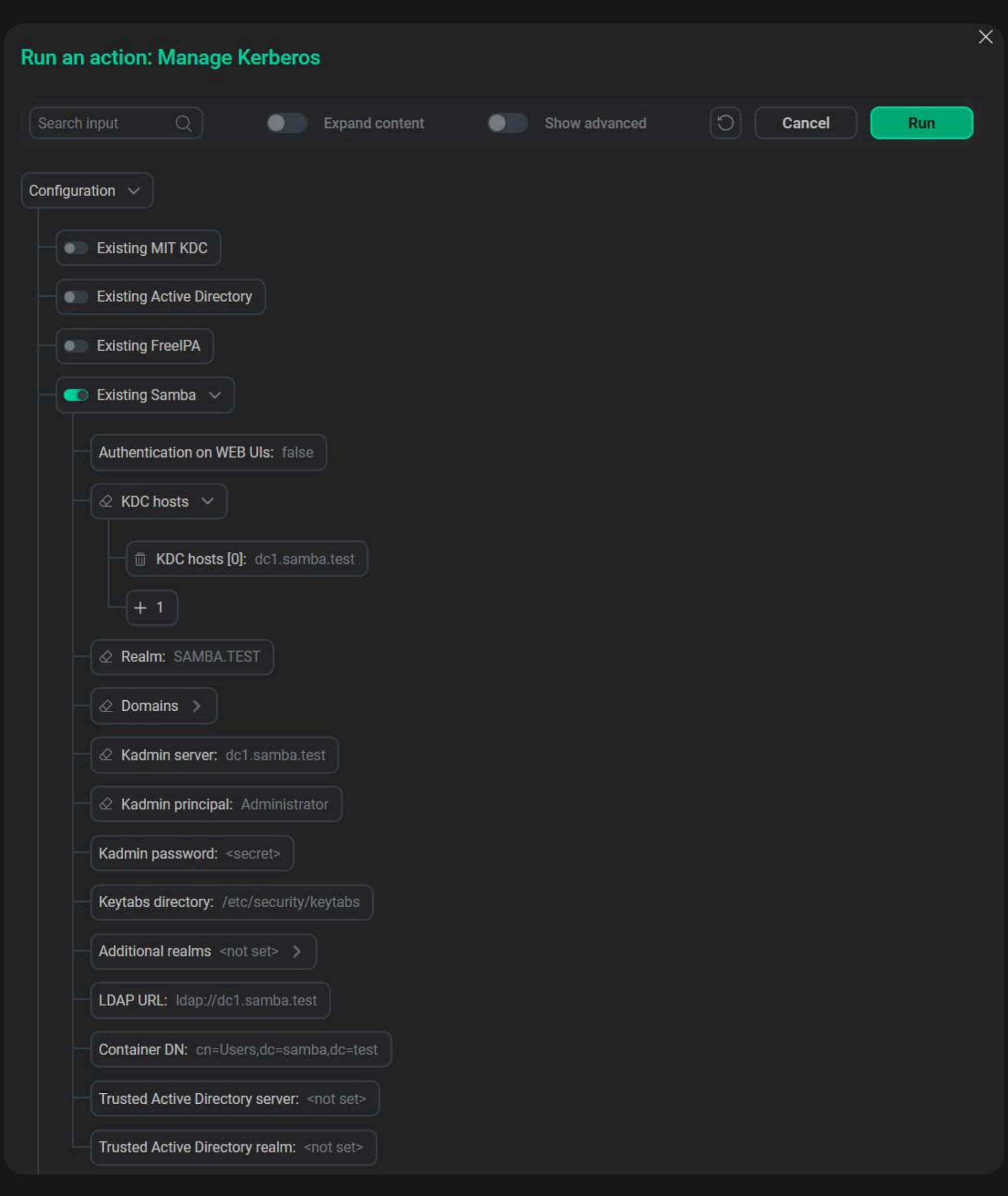
2. In the pop-up window, turn on the ***Existing Samba*** option.



Authentication on WEB UIs: false
KDC hosts
Realm: <not set>
Domains
Kadmin server: <not set>
Kadmin principal: <not set>
Kadmin password: <not set>
Keytabs directory: /etc/security/keytabs
Additional realms <not set>
LDAP URL: <not set>
Container DN: <not set>
Trusted Active Directory server: <not set>
Trusted Active Directory realm: <not set>

Choose the relevant option

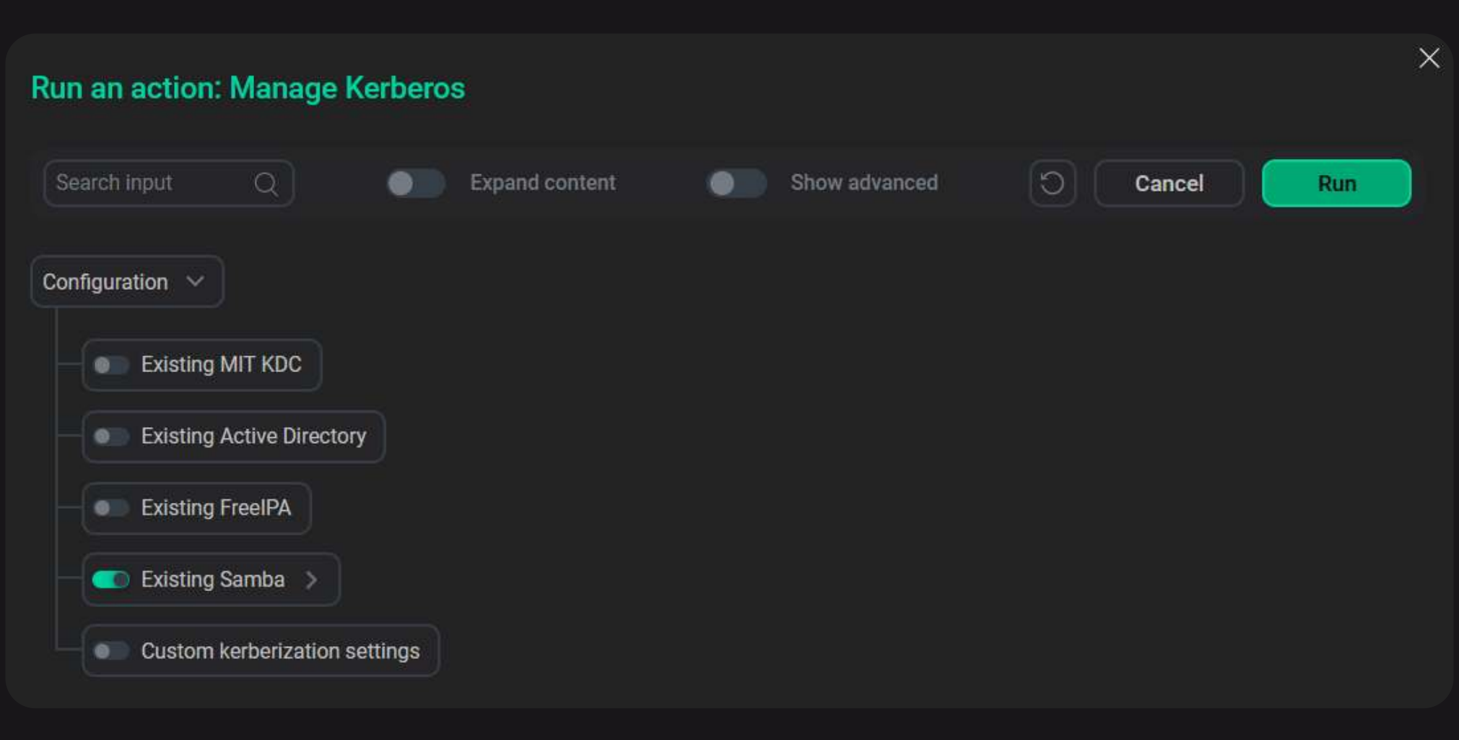
3. Fill in the Samba Kerberos parameters.



Authentication on WEB UIs: false
KDC hosts
KDC hosts [0]: dc1.samba.test
Realm: SAMBA.TEST
Domains
Kadmin server: dc1.samba.test
Kadmin principal: Administrator
Kadmin password: <secret>
Keytabs directory: /etc/security/keytabs
Additional realms <not set>
LDAP URL: ldap://dc1.samba.test
Container DN: cn=Users,dc=samba,dc=test
Trusted Active Directory server: <not set>
Trusted Active Directory realm: <not set>

Samba Kerberos fields

4. Click ***Run***, wait for the job to complete and proceed to setting up Kerberos in the cluster.



Run the action

To check, run the `kinit` command. If it doesn't work with the generated keytabs on your system, it might be useful to add the following password encryption options to the `libdefaults` section of your custom `krb5.conf` cluster parameter in ADCM:

```
[libdefaults]
  permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 arcfour-hmac rc4-hmac arcfour-hmac-md5
  default_tkt_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 rc4-hmac
  default_tgs_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96 rc4-hmac
```

Implementation specifics

In the script, the principals are created in a very specific way (same as in the bundle) for the following reasons:

- In order for the keytab authentication to work, a UPN (User Principal Name) has to be an SPN (Service Principal Name). This is required because Samba doesn't allow searching for principals by a UPN during `kinit`.
- Since a UPN can hold only one record, SPNs have to be created for each host.
- By default, when working with principals and keytabs, Samba consults with a local database, not a remote domain controller. To make it run an action on the actual domain controller, the `-H ldap://$SAMBA_SERVER` argument has to be used.

This approach also has an advantage: there is no need to add a host into a domain and set up local replicas of the Samba databases to create service users and principals. Also, the principal authentication happens with a different keytab for each user, not with a single keytab per host.

Keytab generation script

```
#!/bin/bash
# Script for adding services and keytabs to Cluster hosts
# Edit  HOSTS as it will be suitable for your case.
# EDIT your Samba parameters
# Select services according to the bundle

# Samba credentials
SAMBA_USER='Administrator'
SAMBA_PASSWORD='adminPassword'
SAMBA_SERVER='<samba_server>'
REALM='<samba_realm>'
SAMBA_HOST='<samba_host>'
USER_PASSWORD='userPassword'
OU_NAME='CN=Users,DC=samba,DC=test'

declare -a HOSTS

SSH_OPTS='-o GlobalKnownHostsFile=/dev/null -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'

HOSTS=(
    "List your hosts here"
)
encoded_password=$(echo -n "\"$USER_PASSWORD\"" | iconv -f UTF-8 -t UTF-16LE | base64)

# ADH services
SERVICES=(HTTP airflow flink hbase-phoenix_queryserver hbase-thrift2 hbase hdfs-datanode hdfs-journalnode hdfs-namenode hdfs-zkfc hdfs hive httpfs livy mapreduce-historyserver solr spark sqoop yarn-nodemanager yarn-resourcemanager yarn-timelineserver yarn zeppelin zookeeper impala ssm ssm-agent kyuubi)

mkdir /tmp/keytabs

for HOST in ${HOSTS[@]}; do
    echo "Set hostname"
        ssh $SSH_OPTS $HOST "sudo hostnamectl --static set-hostname $HOST"
    echo "Create keytab dir"
            ssh $SSH_OPTS $HOST "sudo mkdir -p /etc/security/keytabs"
    for SERVICE in ${SERVICES[@]}; do
        echo "Create service user"
        ssh $SSH_OPTS $SAMBA_HOST "sudo useradd $SERVICE"
        echo "Create service principal"
        echo "Create add.ldif"
        ssh $SSH_OPTS $SAMBA_HOST "cat <<EOF > /tmp/add.ldif
DN: CN=${SERVICE}/${HOST},${OU_NAME}
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
accountExpires: 0
sAMAccountName: ${SERVICE}-${HOST}
userPrincipalName: ${SERVICE}/${HOST}@${REALM}
servicePrincipalName: ${SERVICE}/${HOST}
distinguishedName: CN=${SERVICE}/${HOST},${OU_NAME}
userAccountControl: 514

DN: CN=${SERVICE}/${HOST},${OU_NAME}
changetype: modify
replace: unicodePwd
unicodePwd:: $encoded_password

DN: CN=${SERVICE}/${HOST},${OU_NAME}
changetype: modify
replace: userAccountControl
userAccountControl: 66048

EOF"

        echo "sudo ldbmodify -H ldap://$SAMBA_SERVER /tmp/add.ldif -U $SAMBA_USER --password=$SAMBA_PASSWORD"
        ssh $SSH_OPTS $SAMBA_HOST "sudo ldbmodify -H ldap://$SAMBA_SERVER /tmp/add.ldif -U $SAMBA_USER --password=$SAMBA_PASSWORD"

        echo "Delete keytab"
        ssh $SSH_OPTS $SAMBA_HOST "sudo rm -f /tmp/$SERVICE.service.keytab"

        echo "Extract keytab"
        TMP_SCRIPT="/tmp/extract_keytab_${SERVICE}_${HOST}.sh"
        KTUTIL_SCRIPT="/tmp/ktutil_commands_${SERVICE}_${HOST}.txt"
        cat <<EOT > $TMP_SCRIPT
#!/bin/bash
cd /tmp
sudo ktutil < $KTUTIL_SCRIPT
EOT

        cat <<EOT > $KTUTIL_SCRIPT
add_entry -password -p ${SERVICE}/${HOST}@${REALM} -k 1 -e aes256-cts-hmac-sha1-96
${USER_PASSWORD}
add_entry -password -p ${SERVICE}/${HOST}@${REALM} -k 1 -e aes128-cts-hmac-sha1-96
${USER_PASSWORD}
add_entry -password -p ${SERVICE}/${HOST}@${REALM} -k 1 -e arcfour-hmac
${USER_PASSWORD}
wkt ${SERVICE}.service.keytab
quit
EOT

        scp $SSH_OPTS $TMP_SCRIPT $SAMBA_HOST:$TMP_SCRIPT
        scp $SSH_OPTS $KTUTIL_SCRIPT $SAMBA_HOST:$KTUTIL_SCRIPT
        ssh $SSH_OPTS $SAMBA_HOST "chmod +x $TMP_SCRIPT && $TMP_SCRIPT && rm $TMP_SCRIPT && rm $KTUTIL_SCRIPT"
        rm $TMP_SCRIPT
        rm $KTUTIL_SCRIPT

        echo "Try kinit"
        ssh $SSH_OPTS $SAMBA_HOST "sudo klist -k /tmp/$SERVICE.service.keytab"
        echo "Change permission"
        ssh $SSH_OPTS $SAMBA_HOST "sudo chmod 777 /tmp/${SERVICE}.service.keytab"
        echo "Export keytab to local"
        scp $SAMBA_HOST:/tmp/${SERVICE}.service.keytab /tmp/keytabs/${SERVICE}.service.keytab
        echo "Export keytab to host"
        scp /tmp/keytabs/${SERVICE}.service.keytab $HOST:/tmp/${SERVICE}.service.keytab
        ssh $SSH_OPTS $HOST "sudo mv /tmp/${SERVICE}.service.keytab /etc/security/keytabs/${SERVICE}.service.keytab"
        echo "Clean up local temporary keytab"
        rm /tmp/keytabs/$SERVICE.service.keytab
    done

    echo "Change owner and group"

    ssh $SSH_OPTS $HOST sudo chown zookeeper:zookeeper /etc/security/keytabs/zookeeper*
    ssh $SSH_OPTS $HOST sudo chown hdfs:hadoop /etc/security/keytabs/hdfs*
    ssh $SSH_OPTS $HOST sudo chown hdfs:hadoop /etc/security/keytabs/HTTP*
    ssh $SSH_OPTS $HOST sudo chown yarn:hadoop /etc/security/keytabs/yarn*
    ssh $SSH_OPTS $HOST sudo chown hbase:hadoop /etc/security/keytabs/hbase*
    ssh $SSH_OPTS $HOST sudo chown flink:flink /etc/security/keytabs/flink*
    ssh $SSH_OPTS $HOST sudo chown sqoop:sqoop /etc/security/keytabs/sqoop*
    ssh $SSH_OPTS $HOST sudo chown zeppelin:zeppelin /etc/security/keytabs/zeppelin*
    ssh $SSH_OPTS $HOST sudo chown spark:spark /etc/security/keytabs/spark*
    ssh $SSH_OPTS $HOST sudo chown hive:hive /etc/security/keytabs/hive*
    ssh $SSH_OPTS $HOST sudo chown livy:hadoop /etc/security/keytabs/livy*
    ssh $SSH_OPTS $HOST sudo chown httpfs:httpfs /etc/security/keytabs/httpfs*
    ssh $SSH_OPTS $HOST sudo chown mapred:hadoop /etc/security/keytabs/mapreduce-historyserver*
    ssh $SSH_OPTS $HOST sudo chown solr:solr /etc/security/keytabs/solr*
    ssh $SSH_OPTS $HOST sudo chown airflow:airflow /etc/security/keytabs/airflow*
    ssh $SSH_OPTS $HOST sudo chown impala:impala /etc/security/keytabs/impala*

    echo "Change permissions"

    ssh $SSH_OPTS $HOST "sudo chmod -R 777 /etc/security/keytabs/"

done
```

Samba Kerberos parameters

Parameter	Description
Authentication on WEB UIs	Enables Kerberos authentication on Web UIs
KDC hosts	One or more domain controller hosts
Realm	A Kerberos realm
Domains	Domains associated with hosts
Kadmin server	A host where `kadmin` is running
Kadmin principal	A principal name used to connect via `kadmin`, for example `admin@RU-CENTRAL1.INTERNAL`
Kadmin password	A principal password used to connect via `kadmin`
Keytabs directory	Directory of the keytab file that contains one or several principals along with their keys
Additional realms	Additional Kerberos realms
LDAP URL	LDAP URL consists of `ldap://` or `ldaps://`, hostname or IP address, and port of the LDAP server
Container DN	Container distinguished name
Trusted Active Directory server	A trusted DC server
Trusted Active Directory realm	A realm for cross-realm trust

Connect to kerberized Hive using DBeaver

Mikhail Serov

To Table of Contents

To connect to a Hive instance from DBeaver with Kerberos, complete the following steps.

1. You should start with installing DBeaver CE. In this guide, DBeaver 21.3.4 is used.

> **NOTE**
>
> Use DBeaver CE 21.3.4 or later.

2. After DBeaver CE is installed, copy the */etc/krb5.conf* file from the Kerberos server to the *C:\Program Files\DBeaver\jre\conf\security* folder of your Windows machine.
3. Edit the *krb5.conf* file. You must remove all symbols before `[libdefaults]` section and comment the `renew_lifetime = 7d` string.
4. Transfer the keytab principal file. To create a keytab file, you should start the `kadmin` utility at Kerberos server and run the following command:

```
xst -norandkey -k <file_name> <principal_name>
```

For example:

```
xst -norandkey -k hive.keytab hive/hive-kerberos.ru-central1.internal@ADREALM.IO
```

Your *krb5.conf* file should look similar the one below:

krb5.conf

```
[libdefaults]
default_realm = ADREALM.IO
dns_lookup_realm = false
dns_lookup_kdc = false
ticket_lifetime = 24h
#renew_lifetime = 7d
forwardable = true

[realms]
ADREALM.IO = {
kdc = kerberos-server.ru-central1.internal
admin_server = kerberos-server.ru-central1.internal
}

[domain_realm]
.adrealm.io = ADREALM.IO
adrealm.io = ADREALM.IO
.adrealm.io = ADREALM.IO
```

5. Create the *jaas.conf* file inside the DBeaver folder. In this file, you should specify the path to the principal keytab file used for connecting to Hive. You also have to specify the principal name.
   Your *jaas.conf* file should look similarly to this:

jaas.conf

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
doNotPrompt=true
useKeyTab=true
keyTab="C:/Users/Administrator/Downloads/hive.keytab"
useTicketCache=false
renewTGT=false
principal="hive/hive-kerberos.ru-central1.internal@ADREALM.IO";
};
```

6. Create the *dbeaver.ini* file located in the DBeaver folder. You have to add the paths to the *jaas.conf* and *krb5.conf* files that were created earlier. For now we assume that *jaas.conf* and *dbeaver.ini* are in the same folder.

dbeaver.ini

```
-startup
plugins/org.eclipse.equinox.launcher_1.6.200.v20210416-2027.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.2.200.v20210429-1609
-vmargs
-XX:+IgnoreUnrecognizedVMOptions
--add-modules=ALL-SYSTEM
-Dosgi.requiredJavaVersion=11
-Xms64m
-Xmx1024m
-Djava.security.krb5.conf=C:\Program Files\DBeaver\jre\conf\security\krb5.conf
-Djava.security.auth.login.config=jaas.conf
-Djavax.security.auth.useSubjectCredsOnly=false
```

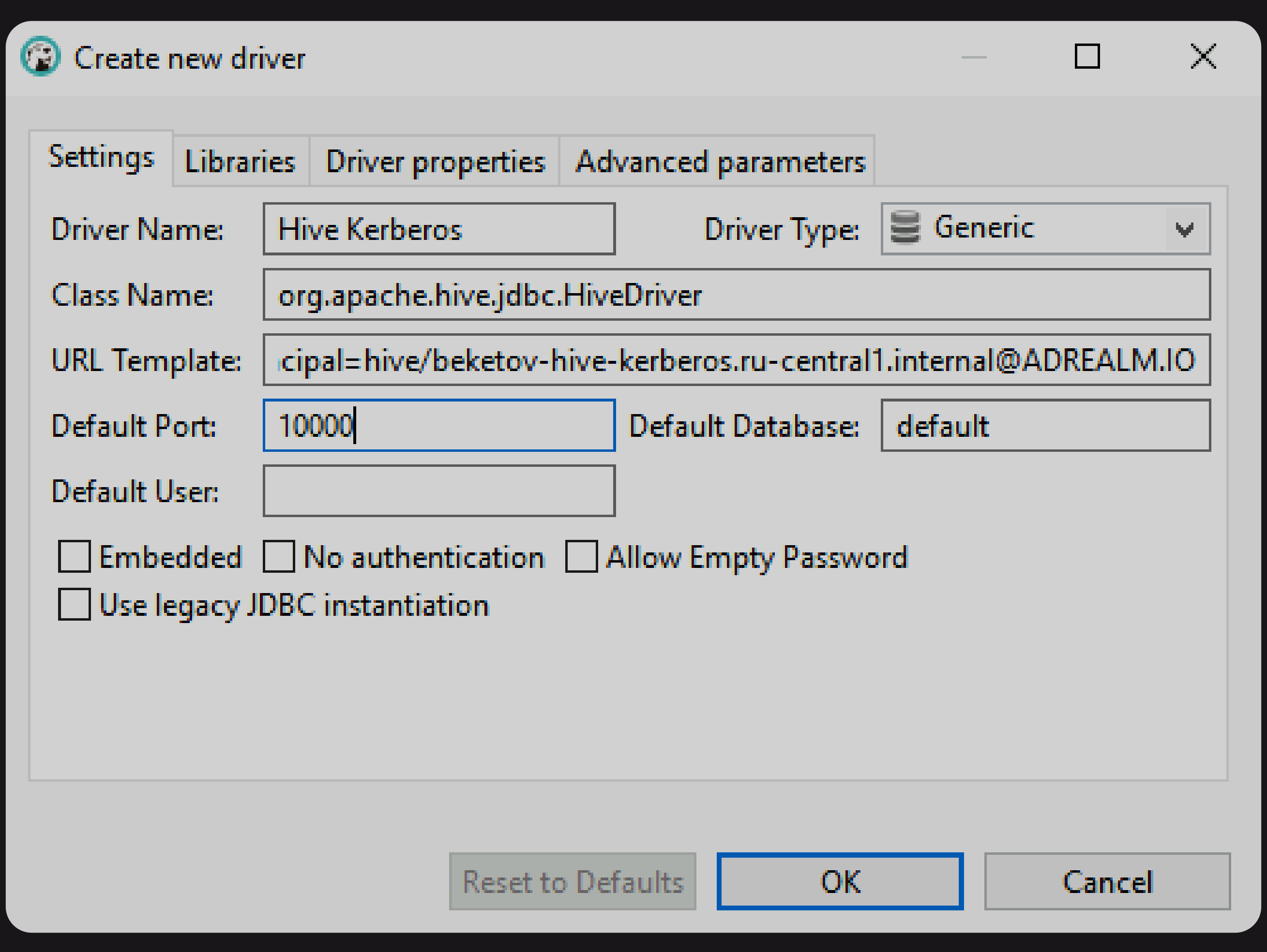
7. Get the authentication ticket using the keytab file. To do this, you should switch to the *C:\Program Files\DBeaver\jre\bin* folder and run the following command:

```
./kinit.exe -k -t keytab-file-path principal_name
```

The output of the command above is similar to this:

```
New ticket is stored in cache file C:\Users\Administrator\krb5cc_Administrator
```

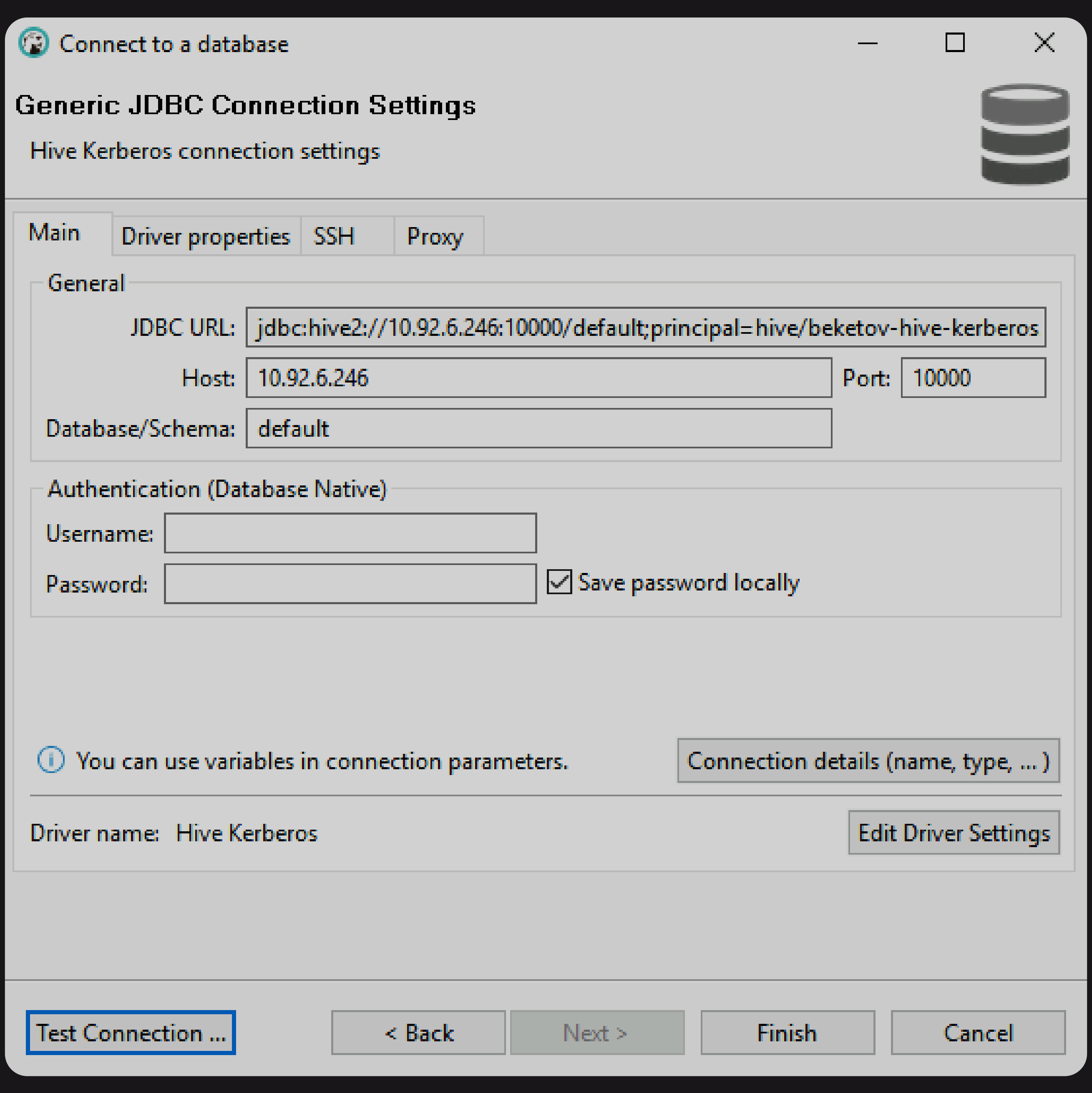
8. Download the driver JAR and unpack it into the DBeaver folder.
9. Now you can run DBeaver. Go to the ***Database*** menu and select ***Driver Manager***. Choose ***New***.
10. Fill in the necessary fields:
    - ***Driver name***: Hive Kerberos;
    - ***URL Template***: jdbc:hive2://{host}:{port}/{database};principal=hive/hive-kerberos.ru-central1.internal@ADREALM.IO;
    - ***Default port***: 10000;
    - ***Default Database***: default.
11. Go to the ***Libraries*** tab and select the `hive-jdbc-3.1.1-arenadata-standalone.jar` driver file that you unpacked earlier. Choose ***FindClass*** and select `org.apache.hive.jdbc.HiveDriver`. Click ***Ok***.



Driver Name: Hive Kerberos
Driver Type: Generic
Class Name: org.apache.hive.jdbc.HiveDriver
URL Template: cipal=hive/beketov-hive-kerberos.ru-central1.internal@ADREALM.IO
Default Port: 10000
Default Database: default
Default User:
☐ Embedded ☐ No authentication ☐ Allow Empty Password
☐ Use legacy JDBC instantiation

Create a driver

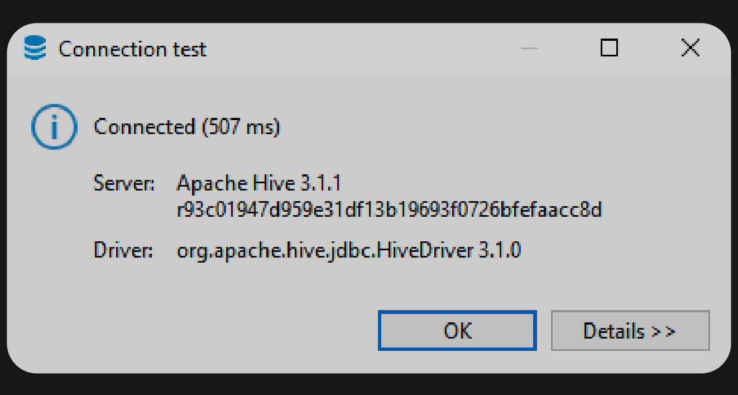
12. Create a new connection. For this, go to ***Database*** → ***New Database Connection*** tab. Choose ***Hive Kerberos***. Fill in the ***Host*** field with the IP address of the Hive server.


Connect to a database
Generic JDBC Connection Settings
Hive Kerberos connection settings
Main
Driver properties
SSH
Proxy
General
Authentication (Database Native)
Connection details (name, type, ...)
Edit Driver Settings
Test Connection ...
< Back
Next >
Finish
Cancel

JDBC URL: jdbc:hive2://10.92.6.246:10000/default;principal=hive/beketov-hive-kerberos
Host: 10.92.6.246
Port: 10000
Database/Schema: default
Username:
Password:
☑ Save password locally
You can use variables in connection parameters.
Driver name: Hive Kerberos

Create a new connection

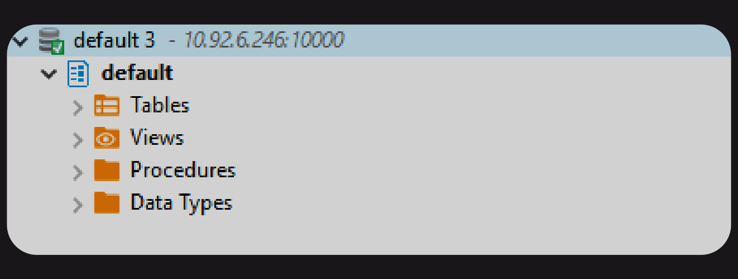
13. Click ***Test Connection*** to verify the connection.



Connected (507 ms)
Server: Apache Hive 3.1.1
r93c01947d959e31df13b19693f0726bfefaacc8d
Driver: org.apache.hive.jdbc.HiveDriver 3.1.0

Connection test

14. Click ***Finish***. The setup is complete.



Hive instance

SPNEGO authentication

Sergei Tikhomirov

Contents

To Table of Contents

[Browser configuration](#)

[Supported services](#)

SPNEGO (Simple and Protected GSS API Negotiation Mechanism) is a mechanism that is used to negotiate the security technology to be used for a safe connection. In particular, it is used when neither client nor server knows what security protocols the other end supports. SPNEGO uses a protocol to find out what GSS API mechanisms are available, selects one, and delegates all the security operations to it.

When a server receives a request from a browser, it can ask the latter to use SPNEGO for authentication. This protocol performs a Kerberos authentication over HTTP. In detail, the process goes as follows:

1. A server returns the `401 Unauthorized` code to a request with the `WWW-Authorization: Negotiate` header.
2. A browser communicates with a TGS to get a ticket.
3. An obtained ticket is converted to a base64 SPNEGO token and sent to a server with an authentication request.
4. A token is unpacked and a server authenticates a ticket.

Browser configuration

Some browsers need additional tweaking for the SPNEGO Kerberos authentication. You can find instructions for the most popular browsers below.

Mozilla Firefox

1. In the search bar, enter the `about:config` command and run it.
2. Change the following parameters with the according values:
   - `network.negotiate-auth.trusted-uris` — your Kerberos-protected domain (e.g. `.ru-central1.internal`).
   - `network.automatic-ntlm-auth.trusted-uris` — your Kerberos-protected domain (e.g. `.ru-central1.internal`).
   - `network.auth.use-sspi` — `false`.
3. Reopen the browser and continue to a protected resource.

Google Chrome

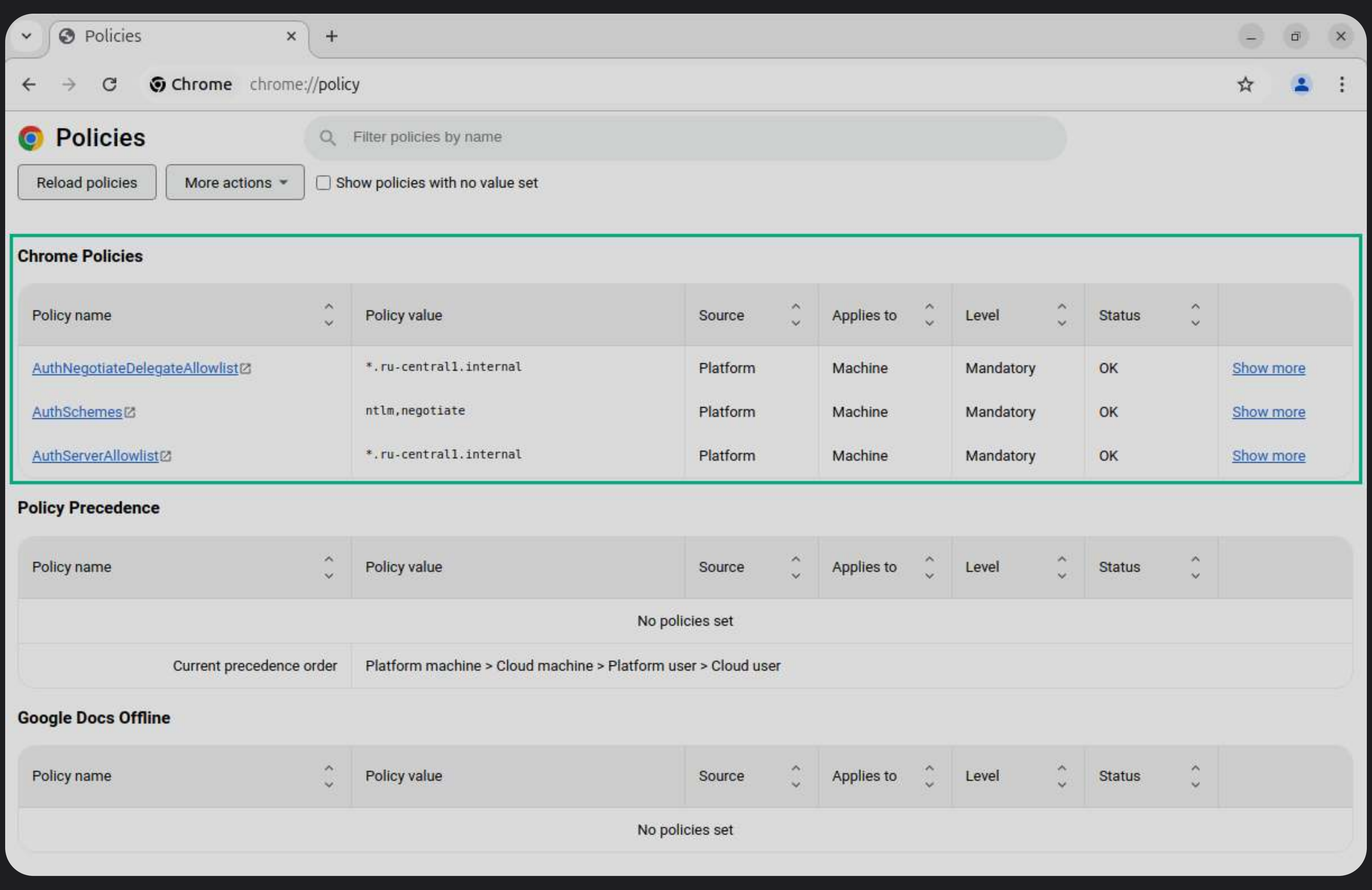
You need to add several browser policies in order for the authentication to function correctly.

- In Windows, open registry editor and go to ***HKEY_LOCAL_MACHINE*** → ***SOFTWARE*** → ***Google*** → ***Chrome***. To add these policies, right click the empty space, select ***New*** → ***String value*** and enter the policy name and value for each policy:
  - `AuthSchemes` — `digest,ntlm,negotiate`.
  - `AuthServerAllowlist` — `<domain>`. If this setting is skipped, Google Chrome will check if the server is in intranet. If it is, the authentication will continue, and if it's not — Google Chrome will ignore the authentication requests from that server.
  - `AuthNegotiateDelegateAllowlist` — `<domain>`. If this setting is skipped, Google Chrome will not pass user credentials to servers even if they are in intranet. In the descriptions above, `<domain>` is your Kerberos-protected domain (e.g. `.ru-central1.internal`).

    Note that the Windows host must be within the domain, while for Linux — it's not a necessity.
- In Linux, go to */etc/opt/chrome/policies/managed* (create missing folders if any) and create a policy file (e.g. *test_policy.json*) with the following contents:

  ```
  {
    "AuthServerAllowlist": "<domain>",
    "AuthNegotiateDelegateAllowlist": "<domain>",
    "AuthSchemes": "digest,ntlm,negotiate"
  }
  ```

You can check if the policies are added correctly by running the `chrome://policy` command in the search bar.



Chrome Policies

Policy name	Policy value	Source	Applies to	Level	Status	
AuthNegotiateDelegateAllowlist	`*.ru-central1.internal`	Platform	Machine	Mandatory	OK	Show more
AuthSchemes	`ntlm,negotiate`	Platform	Machine	Mandatory	OK	Show more
AuthServerAllowlist	`*.ru-central1.internal`	Platform	Machine	Mandatory	OK	Show more

Policy Precedence

Policy name	Policy value	Source	Applies to	Level	Status
No policies set					
Current precedence order	Platform machine > Cloud machine > Platform user > Cloud user				

Google Docs Offline

Policy name	Policy value	Source	Applies to	Level	Status
No policies set					

Google Chrome policies for SPNEGO

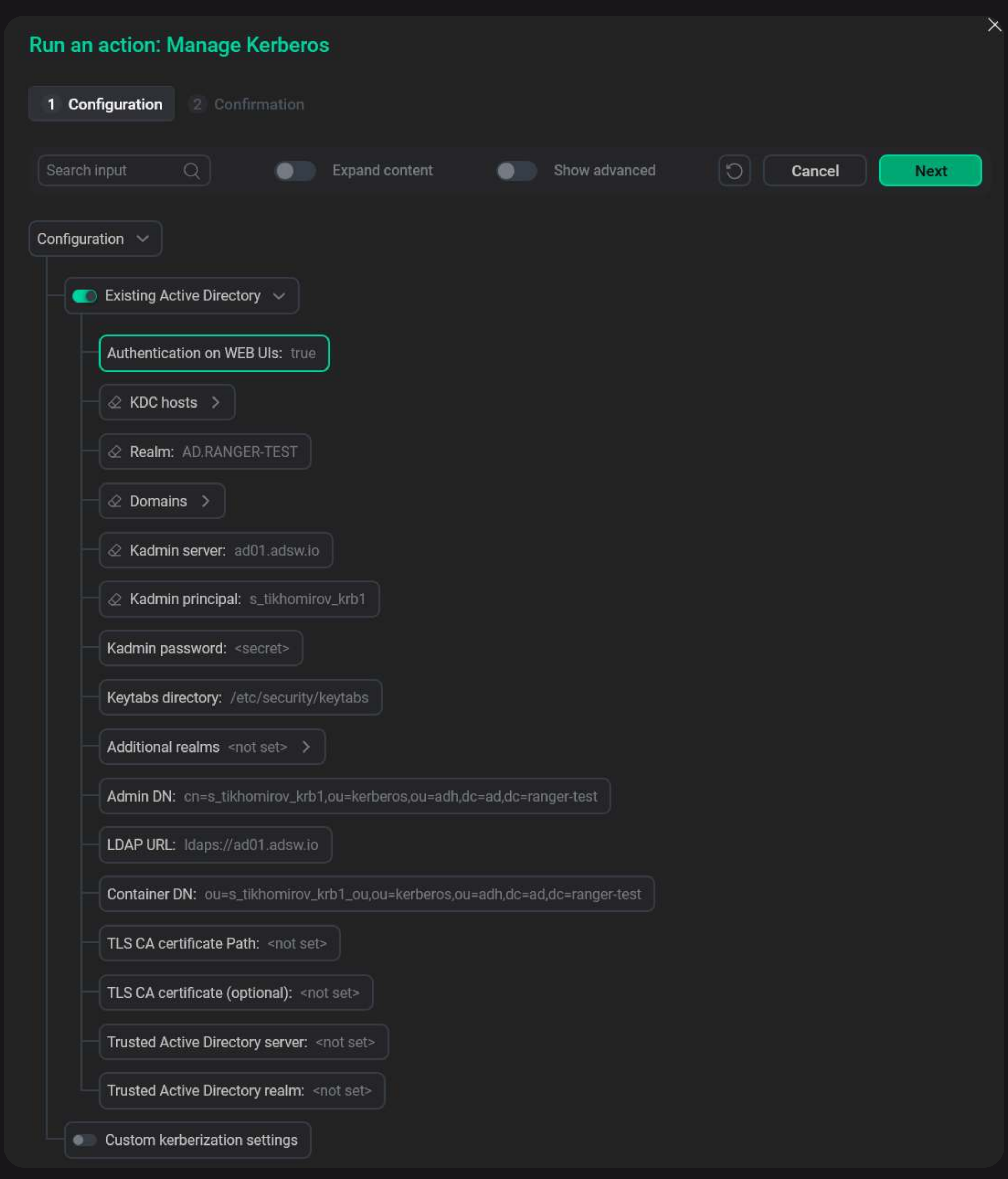
If the tweaking is correct, you will be able to authenticate with a Kerberos ticket that was issued on your system. Otherwise, you will keep getting the `401 Unauthorized` error.

Supported services

The following ADH services support the authentication on web UIs with SPNEGO:

- Flink
- HDFS
- Hive
- HUE
- Impala
- Kyuubi
- Spark3
- SSM
- YARN

To activate the SPNEGO mechanism for them (unless a separate instruction is provided), you need to enable the ***Authentication on WEB UIs*** flag during the Kerberos activation.



Authentication on WEB UIs: true
Realm: AD.RANGER-TEST
Kadmin server: ad01.adsw.io
Kadmin principal: s_tikhomirov_krb1
Kadmin password: <secret>
Keytabs directory: /etc/security/keytabs
Additional realms <not set>
Admin DN: cn=s_tikhomirov_krb1,ou=kerberos,ou=adh,dc=ad,dc=ranger-test
LDAP URL: ldaps://ad01.adsw.io
Container DN: ou=s_tikhomirov_krb1_ou,ou=kerberos,ou=adh,dc=ad,dc=ranger-test
TLS CA certificate Path: <not set>
TLS CA certificate (optional): <not set>
Trusted Active Directory server: <not set>
Trusted Active Directory realm: <not set>

The "Authentication on WEB UIs" parameter

Requirements for SSL certificates

Mikhail Serov

To Table of Contents

To implement encryption between the nodes the following is required:

1. Create a certificate for each node using the RSA algorithm. The key length should be 2048 bit, CN should be the FQDN of the corresponding node. It is not recommended to use the DSA algorithm. It is also not recommended to use the ECDSA algorithm due to this algorithm not being widely supported by certification centers. The key length that equals 2048 bits is necessary and sufficient to implement secure encryption.

2. Create the *trustore.jks* keystore file that will store the certificates. The path to this file should be specified for SSL to be started.

NOTE

- Certificates should be signed by the key that belongs to one of the trusted root certificates included into the Java certificate storage.
- Self-signed certificates are allowed. In this case, the root certificate should be added to the *trustore.jks* storage.
- Nginx requires OpenSSL certificates. For a cluster that uses Nginx (with Hive TezUI or Airflow), generate an OpenSSL key and a certificate.
- The OpenSSL certificate for your host name should be added to the *ca-bundle.crt* storage.
- The account used for the installation of certificates should have the rights to write to the following paths:
  - */etc/pki/tls/certs/*
  - */etc/pki/java/*
- Make sure to use an FQDN as a key alias in *keystore.jks*, not a short name.

3. Enable the traffic between the nodes according to the table below.
   The script below creates all the necessary entities and can be used as a reference.

```
#!/bin/bash

# Script for generating and import self-signed certificates to java keystore and openssl ca-bundle
# Edit NUMHOSTS and HOSTS as it will be suitable for your case.


declare -a NUMHOSTS
declare -a HOSTS

SSH_OPTS='-o GlobalKnownHostsFile=/dev/null -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no'
NUMHOSTS=(1 2 3)
HOSTS=$(for i in "${NUMHOSTS[@]}"; do echo "adh-$i.ru-central1.internal"; done)

echo Generate keystore.jks on each host
for HOST in $HOSTS; do
  echo "Generating keypair"
        ssh $SSH_OPTS $HOST "keytool -genkeypair -noprompt -keyalg RSA -alias $HOST -dname \"CN=$HOST, OU=AD, O=AD, L=MSK, S=MO,
C=RU\" -keystore /tmp/keystore.jks -storepass bigdata -keypass bigdata -validity 360 -keysize 2048";
done

echo
echo Export certificates
for HOST in $HOSTS;do
        ssh $SSH_OPTS $HOST "keytool -exportcert -file /tmp/$HOST.crt -keystore /tmp/keystore.jks -storepass bigdata -alias $HOST -
rfc";
done

echo
echo Collect all certificates
for HOST in $HOSTS; do
        scp $SSH_OPTS $HOST:/tmp/$HOST.crt /tmp/
done

echo
echo Transfer certificates on hosts
for HOST in $HOSTS; do
        scp $SSH_OPTS /tmp/*.crt $HOST:/tmp/
done

echo
echo Import certificates on each host
for HOST in $HOSTS; do
        ssh $SSH_OPTS $HOST "for CERT in $(echo ${HOSTS[*]}); do
            keytool -importcert -noprompt -alias \$CERT -file /tmp/\$CERT.crt -keystore /tmp/truststore.jks -storepass bigdata;
            sudo bash -c \"cat /tmp/\$CERT.crt >> /etc/pki/tls/certs/ca-bundle.crt\";
        done";
done

echo
echo Import truststore to Java CA store
for HOST in $HOSTS; do
        ssh $SSH_OPTS $HOST "sudo keytool -importkeystore -noprompt -srckeystore /tmp/truststore.jks -destkeystore
/etc/pki/java/cacerts -deststorepass changeit -srcstorepass bigdata"
done

echo
echo Create and import OpenSSL cert for Nginx
for HOST in $HOSTS; do
        ssh $SSH_OPTS $HOST "sudo openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 -sub
j \"/C=RU/ST=Denial/L=MSK/O=AD/CN=$HOST\" -keyout /etc/ssl/host_cert.key  -out /etc/ssl/certs/h
ost_cert.cert"
        ssh $SSH_OPTS $HOST "sudo bash -c \"cat /etc/ssl/certs/host_cert.cert >> /etc/pki/tls/c
erts/ca-bundle.crt\""
done
```

Matching between ports and HTTP/HTTPS services

Service	Component/Server	Port	Protocol	Description
Airflow	Server	8080/8080	HTTP/HTTPS	Airflow Server Web UI port
	Server/Worker	8793/8793	HTTP/HTTPS	Celery Worker API port
	Server/Flower	5555/5555	HTTP/HTTPS	Flower Web UI port
Flink	Job Manager	8081/8081	HTTP/HTTPS	Flink Job Manager Web UI port
HBase	HBase Region	16030/16030	HTTP/HTTPS	HBase Region Server Web UI port
	Phoenix Query Server	8765/8765	HTTP/HTTPS	API port
	HBase REST Server	60080/60080	HTTP/HTTPS	API port
	HBase REST Server	8085/8085	HTTP/HTTPS	Web UI Port
	HBase Master	16010/16010	HTTP/HTTPS	HBase Master Web UI port
HBaseThrift2Server	Trift2 Server	9095/9095	HTTP/HTTPS	Thrift2 Server Web UI port
HDFS	NameNode	9870/9871	HTTP/HTTPS	NameNode Web UI port
	DataNode	9864/9865	HTTP/HTTPS	DataNode Web UI port
	JournalNode	8480/8481	HTTP/HTTPS	JournalNode Web UI port
	HTTPfs server	14000/14000	HTTP/HTTPS	HTTPfs API port
Hive	Hive Server	10002/10002	HTTP/HTTPS	Hive Server Web UI port
	Tez	9999/9999	HTTP/HTTPS	Tez Web UI port
Knox	Knox Gateway	8443	HTTPS	Gateway port
Ranger	Ranger Admin	6080/6182	HTTP/HTTPS	Ranger Admin web UI and API port
	Ranger KMS	9292/9393	HTTP/HTTPS	Port for Ranger KMS
Solr	Solr Server	8983/8985	HTTP/HTTPS	Solr Server Web UI and API port
Spark	History Server	18082/18082	HTTP/HTTPS	HS Web UO port
	Thrift Server	4040/4040	HTTP/HTTPS	Thrift Server Web UI
	Livy Server	8998/8998	HTTP/HTTPS	Livy Server Web UI port
YARN	Resource Manager	8088/8090	HTTP/HTTPS	RM Web UI port
	Node Manager	8042/8044	HTTP/HTTPS	NM Web UI port
	MapReduce History Server	19888/19890	HTTP/HTTPS	HS Web UI port
	Timeline Server	8188/8190	HTTP/HTTPS	TS Web UI port
Zeppelin	Server	8180/8180	HTTP/HTTPS	Zeppelin Web UI port

Set TLS version for an ADH cluster

Konstantin Alpashkin

Contents

To Table of Contents

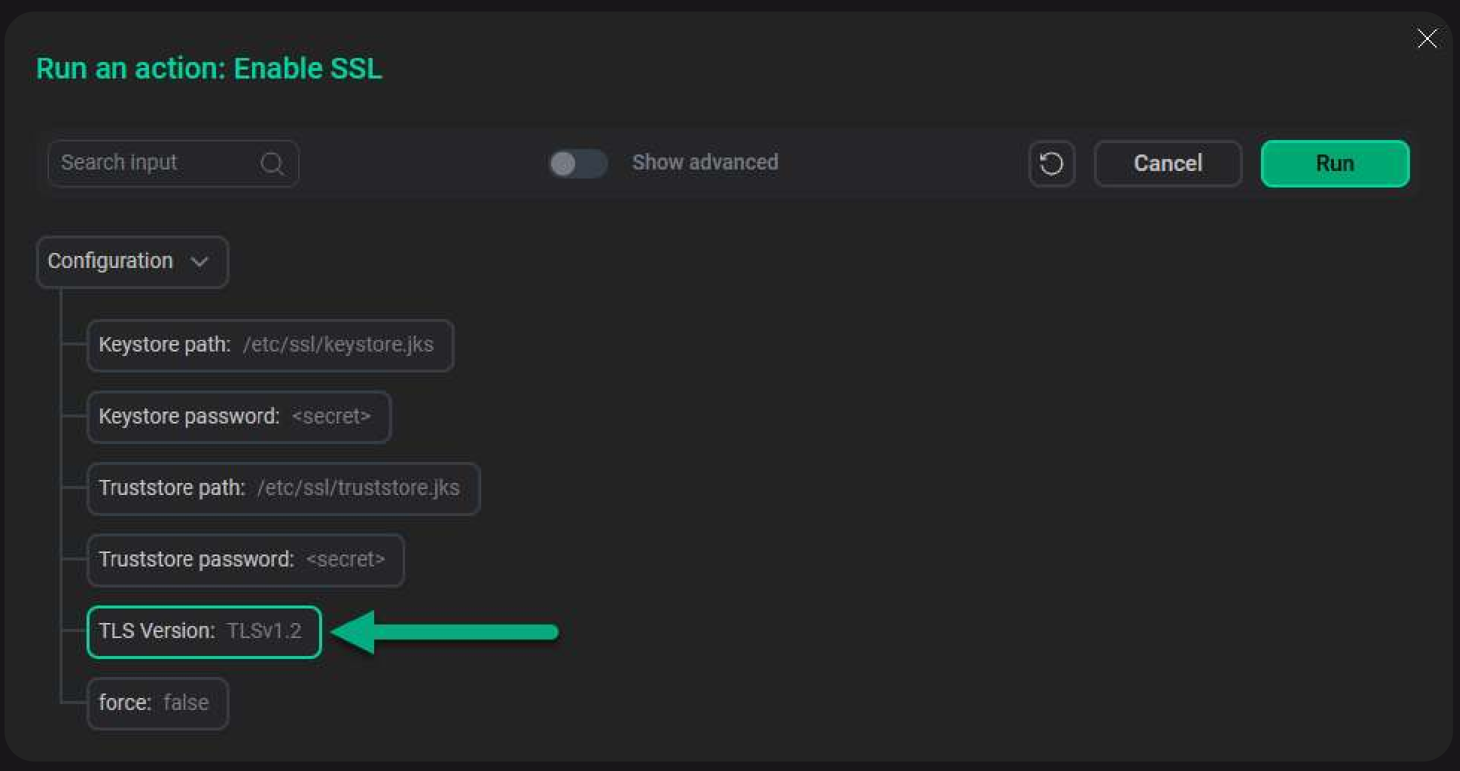
- Select TLS version
- Per-service TLS settings
- Test TLS version
  - Use curl
  - Use OpenSSL

This section shows how to configure an ADH cluster to use a specific TLS version for SSL communication and how to verify the current TLS version that is in use.

As of ADH 2.1.10.b1, the minimum supported TLS version is TLSv1.2 as restricted by Java 8. TLSv1.3 is supported on JDK11+ and JDK8 build 8u261 or later (JRE 1.8.0_261-b12+).

Select TLS version

You can specify the desired TLS version for entire ADH cluster using the ***TLS Version*** parameter when running the Manage SSL action. As of 2.1.10.b1, the only supported version is `TLSv1.2`.


Run an action: Enable SSL

Keystore path: /etc/ssl/keystore.jks
Keystore password: <secret>
Truststore path: /etc/ssl/truststore.jks
Truststore password: <secret>
TLS Version: TLSv1.2
force: false

Select TLS version

Per-service TLS settings

You can also set a specific TLS version for individual ADH components. The list of configuration files and parameters responsible for the TLS protocol selection is presented below.

TLS configuration for ADH services

Service	Configuration file	Parameter	Default value	Description
Hive	/etc/hive/conf/hive-site.xml	hive.ssl.protocol.blacklist	SSLv2Hello,SSLv2,SSLv3,TLSv1,TLSv1.1	Disables specified TLS protocols for all Hive Servers
TezUI	/etc/nginx/nginx.conf	server.ssl_protocols	TLSv1.2	A comma-separated list of TLS protocols supported for the SSL transport
Metastore	/etc/hive/conf/hive-site.xml	metastore.ssl.protocol.blacklist	SSLv2Hello,SSLv2,SSLv3,TLSv1,TLSv1.1	Disables specified TLS protocols for Hive Metastore
HDFS, YARN	/etc/hadoop/conf/core-site.xml	hadoop.ssl.enabled.protocols	TLSv1.2	A comma-separated list of TLS protocols used for SSL communication between HDFS DataNodes
Spark3	/etc/spark3/conf/spark-defaults.conf	spark.ssl.protocol	TLSv1.2	Specifies the TLS protocol to use
Livy	/etc/livy-spark3/conf/livy.conf	livy.server.thrift.ssl.protocol.blacklist	SSLv2,SSLv3,TLSv1,TLSv1.1	Disables specified TLS protocols
Flink	/etc/flink/conf/flink-conf.yaml	security.ssl.protocol	TLSv1.2	Specifies a single TLS version to be used for the SSL transport. Does not support comma-separated values
ZooKeeper 3.5.5+	/etc/zookeeper/conf/zoo.cfg	ssl.protocol ssl.quorum.protocol	TLSv1.2	`ssl.protocol` defines a protocol for the client SSL negotiation. `ssl.quorum.protocol` defines a protocol to use for quorum SSL negotiation

Test TLS version

To get the TLS version currently used by ADH, you can use curl or OpenSSL as shown below.

Use curl

Below is a sample HTTPS request to the Hive server forcing curl to use the fairly deprecated TLSv1.1.

```
$ curl https://ka-adh-1.ru-central1.internal:10002/  --insecure --tlsv1.1 --tls-max 1.1
```

The response indicates that TLSv1.1 is unsupported.

```
curl: (35) Peer reports incompatible or unsupported protocol version
```

If you omit `--tlsv1.1 --tls-max 1.1`, the default TLSv1.2 is used and curl returns a web page.

Use OpenSSL

The command below opens an SSL connection to a Tez server and OpenSSL attempts to get SSL certificates using the deprecated TLSv1.1.

```
$ openssl s_client -connect ka-adh-2.ru-central1.internal:9999 -tls1_1
```

OpenSSL fails to establish the connection due to the unsupported TLS version.

```
no peer certificate available. No client certificate CA names send
```

If you omit `-tls1_1`, the default TLSv1.2 is used and OpenSSL retrieves a certificate.

Impersonation settings in ADH

Sergei Tikhomirov

Contents

To Table of Contents

- Overview
- Livy impersonation
  - ADCM preparation
  - Host preparation
  - Testing
- SSM impersonation
- Zeppelin impersonation

Overview

In the Hadoop ecosystem, impersonation is a mechanism that allows a service to perform an action on behalf on a different user. Impersonation is implemented in order to conform with security policies (e.g. Kerberos or Ranger policies), isolate data and resources among users, and to audit actions while writing the name of an actual user who launched an action.

ADH supports impersonation for HBase (Phoenix, Thrift), Spark (Livy), Hive, Zeppelin, Flink, Solr, Sqoop, Airflow. By default, impersonation is enabled for all services, but some of them require additional tweaking which is described below. To make a service impersonate certain users, you need to change the ***hadoop.proxyuser.<service>.groups*** and ***hadoop.proxyuser.<service>.hosts*** parameters in the ***core-site.xml*** parameter group of the ***Core configuration*** service.

Livy impersonation

The example below shows how to enable impersonation for Livy.

ADCM preparation

1. In ADCM, select your ADH cluster and head to the Spark service. Expand the ***livy.conf*** parameter group and make sure that the ***livy.impersonation.enabled*** parameter is set to `true`.

2. Head to the ***Core configuration*** service and edit the ***hadoop.proxyuser.livy.groups*** and ***hadoop.proxyuser.livy.hosts*** parameters. By default, they are filled with `*`, but you need to put actual group names/host addresses there. The service will be allowed to impersonate users from the listed groups if a request came from a whitelisted host.

3. Save the configuration and run the ***Update Core configuration*** action.

Host preparation

1. Create user directories in HDFS (here and below, `<username>` is a user that Livy will be allowed to impersonate, while `<group>` is a group, which `<username>` is a part of):

```
$ sudo -u hdfs hdfs dfs -mkdir /user/<username>
$ sudo -u hdfs hdfs dfs -chown <username>:<group> /user/<username>
```

2. Grant the required permissions to a user and copy a JAR with Spark examples to HDFS:

```
$ hdfs dfs -chmod -R a+rwx /user/<username>
$ hdfs dfs -chmod -R a+rwx /var/log/spark/apps
$ hdfs dfs -mkdir /user/<username>/spark_examples
$ hdfs dfs -put /usr/lib/spark/examples/jars/* /user/<username>/spark_examples
```

Testing

1. To test, create a similar JSON:

```
{
"file": "/user/<username>/spark_examples/spark-examples_2.11-2.3.2.3.1.0.15-1.jar",
"className": "org.apache.spark.examples.SparkPi",
"name": "test-livy-1",
"args": [10],
"proxyUser": "<username>"
}
```

2. Send a request with the JSON to Livy:

```
$ curl --negotiate -v -u: -d @app_spark.json -H "X-Requested-By: <username>" -H 'Content-Type: application/json' -X POST
'http://kru-adh-22.ru-central1.internal:8998/batches'
```

SSM impersonation

To configure impersonation in SSM, complete the following steps:

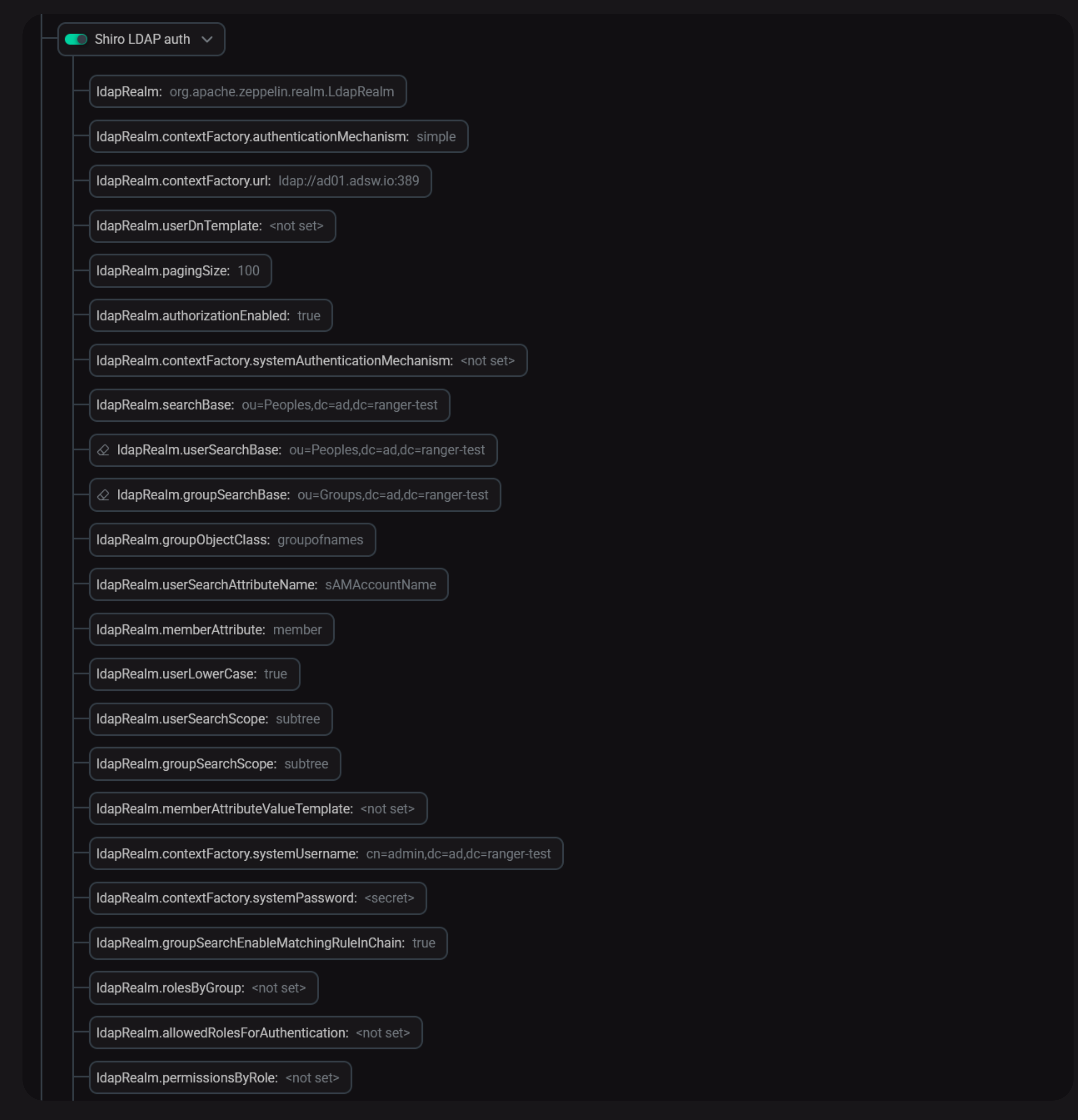
1. Go to the ADCM UI and select your cluster on the ***Clusters*** page.
2. Open the ***Services*** tab and select the ***SSM*** service.
3. Switch on the ***Show advanced*** toggle and expand the ***Custom smart-site.xml*** parameter group.
4. Click ***Add property*** and specify the following values:
   - Field name — `smart.proxy.user`;
   - Field value — name of the user for the impersonation.
5. Click ***Apply***.
6. Expand the ***smart-site.xml*** parameter group and specify the following parameters:
   - ***smart.proxy.user.strategy*** — defines the impersonation strategy to use. Select one of the following:
     - `NODE_SCOPE` — enable impersonation on node level. All actions will be performed by the previously specified user.
     - `CMDLET_SCOPE` — enable impersonation on cmdlet level. All actions will be performed by the cmdlet owner (the cmdlet creator).
   - ***smart.proxy.users.cache.ttl*** — time interval since the last access to the proxy user cache entry before that entry is discarded.
   - ***smart.proxy.users.cache.size*** — size of the proxy users cache.
7. Click ***Save***, then ***Create***.
8. Click ***Actions*** and select ***Restart***.

Zeppelin impersonation

Zeppelin impersonation requires the Active Directory Kerberos to be enabled both in ADH and integrated ADPS clusters.Ranger plugins should be enabled for HDFS, YARN, and Hive.

Follow the steps below to see Zeppelin impersonation in effect:

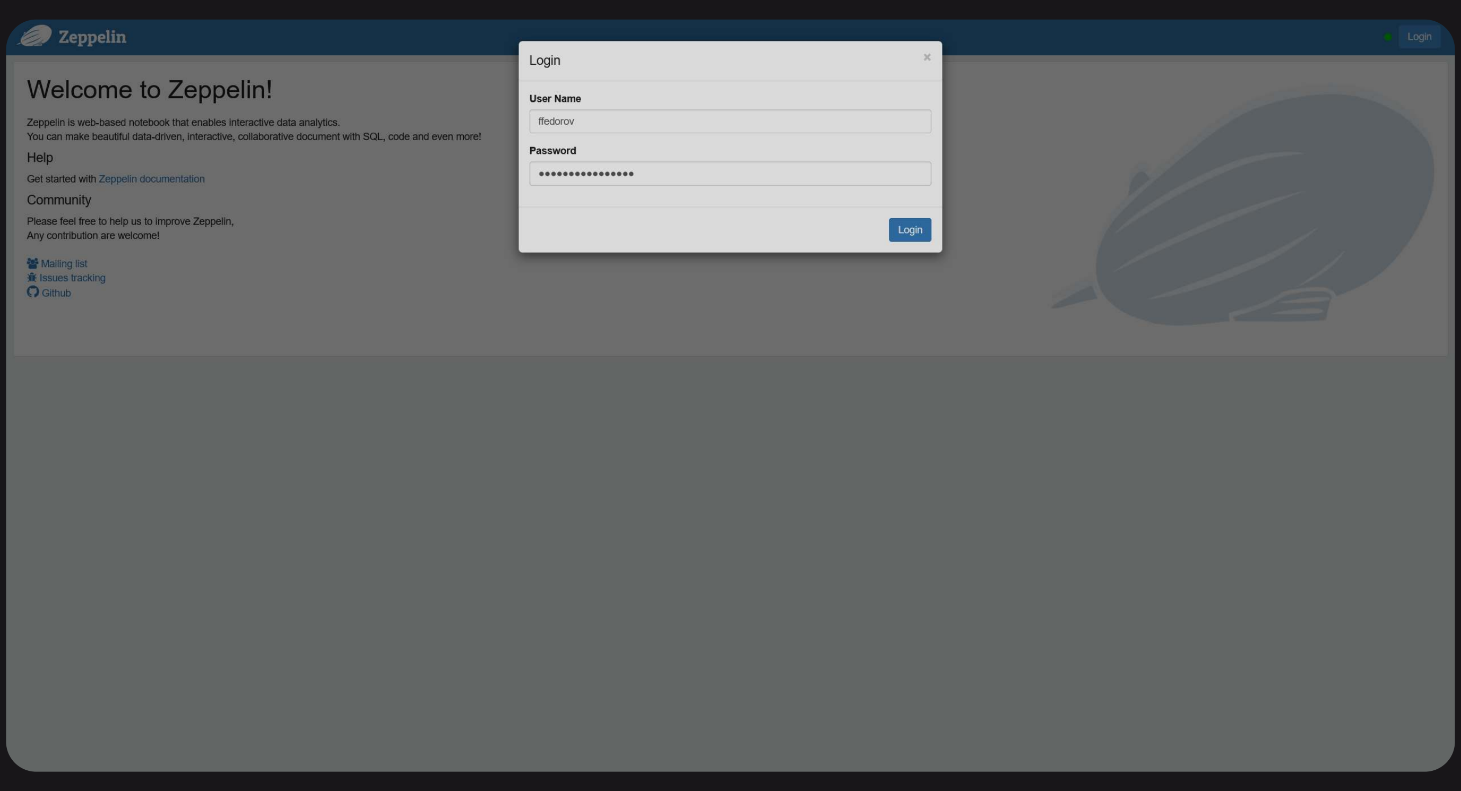
1. In ADCM, select your ADH cluster and head to the Zeppelin service. Enable the ***Shiro LDAP auth*** parameter and fill in the details. After that, restart Zeppelin.



ldapRealm: org.apache.zeppelin.realm.LdapRealm
ldapRealm.contextFactory.authenticationMechanism: simple
ldapRealm.contextFactory.url: ldap://ad01.adsw.io:389
ldapRealm.userDnTemplate: <not set>
ldapRealm.pagingSize: 100
ldapRealm.authorizationEnabled: true
ldapRealm.contextFactory.systemAuthenticationMechanism: <not set>
ldapRealm.searchBase: ou=Peoples,dc=ad,dc=ranger-test
ldapRealm.userSearchBase: ou=Peoples,dc=ad,dc=ranger-test
ldapRealm.groupSearchBase: ou=Groups,dc=ad,dc=ranger-test
ldapRealm.groupObjectClass: groupofnames
ldapRealm.userSearchAttributeName: sAMAccountName
ldapRealm.memberAttribute: member
ldapRealm.userLowerCase: true
ldapRealm.userSearchScope: subtree
ldapRealm.groupSearchScope: subtree
ldapRealm.memberAttributeValueTemplate: <not set>
ldapRealm.contextFactory.systemUsername: cn=admin,dc=ad,dc=ranger-test
ldapRealm.contextFactory.systemPassword: <secret>
ldapRealm.groupSearchEnableMatchingRuleInChain: true
ldapRealm.rolesByGroup: <not set>
ldapRealm.allowedRolesForAuthentication: <not set>
ldapRealm.permissionsByRole: <not set>

Zeppelin LDAP auth parameters

2. In the Zeppelin web interface, log in as an Active Directory user.



Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Help

Get started with Zeppelin documentation

Community

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

User Name: ffedorov
Password: ••••••••••••••••

Zeppelin web interface login

3. Choose an interpreter. Create a note with a test command and run it on the chosen interpreter.

4. In Ranger Admin, head to the ***Audit*** → ***Access*** page, and you will see a log record with an AD user.

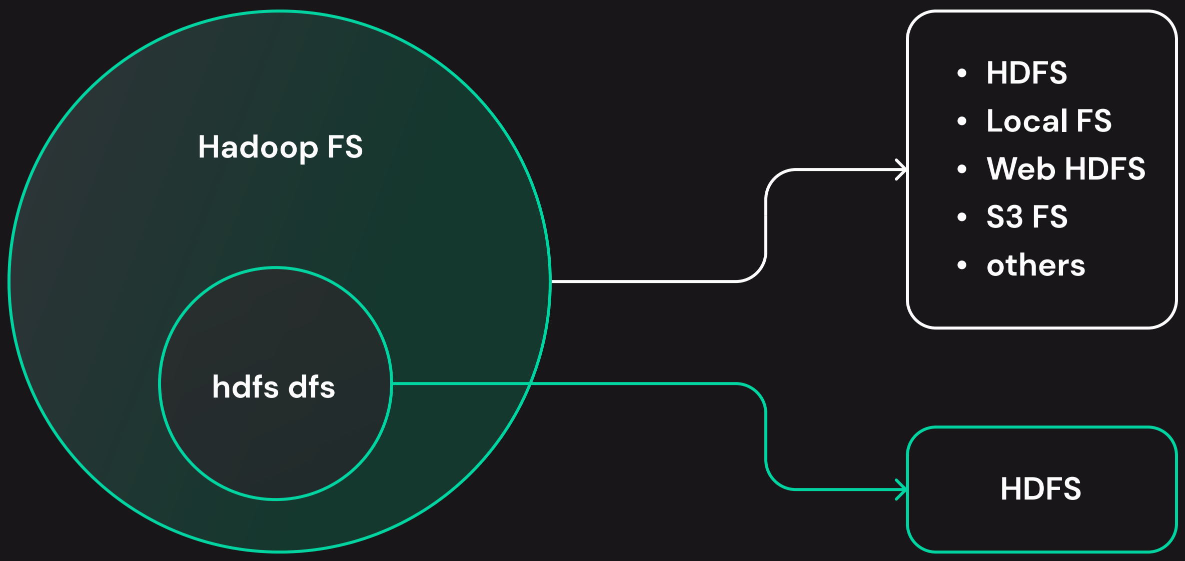
Hadoop command-line

Sergey Ostapov

To Table of Contents

This article explains the difference between `hadoop fs {args}`, `hadoop dfs {args}`, and `hdfs dfs {args}` commands.

The `fs` parameter refers to a shared file system that can point to any file system, such as local, HDFS, etc.
So, `fs` can be used while working with various file systems, such as Local FS, (S)FTP, S3, and others.



Hadoop FS layers

When you use the `dfs` parameter, you directly specify the interaction with HDFS.
So, if you want to work only with HDFS, use the `hdfs dfs` command.
But if you need to access/transfer data between different file systems, use the `hadoop fs` command.

To interact with the HDFS shell, a user must have sufficient read/write permissions to an HDFS directory(-ies). These directories can be created by a user that belongs to either `hadoop` or `hdfs` group. For demonstration purposes, we use the `hdfs` user in the example below.

First, you can grant `root` privileges to the current user:

```
$ sudo -s
```

And then you need to switch to `hdfs` user:

```
$ su - hdfs
```

Now you can execute commands in the HDFS shell.

Hadoop CLI

Sergey Ostapov

Contents

To Table of Contents

- Overview
- Hadoop shell basic structure
- User commands
- Administration commands

Overview

This article includes the reference documentation for the Hadoop shell command-line tool.

NOTE

Currently, the `hadoop dfs` command is deprecated, use `hdfs dfs` instead

All the Hadoop commands and subprojects follow the same basic structure:

The usage is as follows:

```
$ shellcommand [SHELL_OPTIONS] [COMMAND] [GENERIC_OPTIONS] [COMMAND_OPTIONS]
```

Hadoop shell basic structure

shellcommand	The command of the project being invoked. For example, Hadoop commonly uses `hadoop`, HDFS uses `hdfs`, and YARN uses `yarn`
SHELL_OPTIONS	Options that the shell processes before executing Java
COMMAND	Action to perform
GENERIC_OPTIONS	The common set of options supported by multiple commands
COMMAND_OPTIONS	Various command options for the Hadoop common subprojects

All the shell commands accept a common set of options. For some commands, these options are ignored. For example, passing `--hostnames` on a command that only executes on a single host will be ignored.

Shell options

--buildpaths	Enables developer versions of JARs
--config confdir	Overwrites the default configuration directory. The default directory is *$HADOOP_HOME/etc/hadoop*
--daemon mode	If the command supports daemonization (e.g., `hdfs namenode`), executes in the appropriate mode. Supported modes are `start` to start the process in a daemon mode, `stop` to stop the process, and `status` to determine the active status of the process. The status will return an LSB-compliant result code. If no option is provided, commands that support daemonization will run in the foreground. For commands that don't support daemonization, this option is ignored
--debug	Enables shell-level configuration debugging information
--help	Displays shell script usage information
--hostnames	When `--workers` is used, overrides the *workers* file with a whitespace-delimited list of hostnames where to execute a multi-host subcommand. If `--workers` isn't used, this option is ignored
--hosts	When `--workers` is used, overrides the *workers* file with another file that contains a list of hostnames where to execute a multi-host subcommand. If `--workers` isn't used, this option is ignored
--loglevel loglevel	Overrides the log level. Valid log levels are `FATAL`, `ERROR`, `WARN`, `INFO`, `DEBUG`, and `TRACE`. Default is `INFO`
--workers	If possible, executes this command on all hosts in the *workers* file

Many subcommands share a common set of configuration options to alter their behavior.

Generic options

-archives <comma separated list of archives>	Specifies comma-separated archives to be extracted onto the compute machines. Applies only to a job
-conf <configuration file>	Specifies an application configuration file
-D <property>=<value>	Sets a value for a given property
-files <comma separated list of files>	Specifies comma-separated files to be copied to the MapReduce cluster. Applies only to a job
-fs <file:///> or <hdfs://namenode:port>	Specifies the default file system URL to use. Overrides `fs.defaultFS` property from configurations
-jt <local> or <resourcemanager:port>	Specifies a ResourceManager. Applies only to a job
-libjars <comma separated list of jars>	Specifies comma-separated JAR files to include in the classpath. Applies only to a job

User commands

These commands are helpful for Hyperwave cluster users.

Command	Description
archive	Creates a Hadoop archive
checknative	Checks the availability of the Hadoop native code
CLASSNAME	Runs an arbitrary Java class
classpath	Prints the classpath
credential	Manages credentials, passwords, and secrets
distch	Changes the ownership and permissions on files
distcp	Copies file or directories recursively
dtutil	Utility to fetch and manage Hadoop tokens
envvars	Displays computed Hadoop environment variables
fs	It's a synonym for `hdfs dfs` when HDFS is in use
gridmix	Benchmark tool for Hyperwave cluster
jar	Runs a JAR file
jnipath	Prints the computed java.library.path
kerbname	Converts the named principal via the auth_to_local rules to the Hadoop username
kdiag	Diagnoses Kerberos problems
key	Manages keys via the KeyProvider
kms	Runs the Key Management Server
trace	Views and modifies Hadoop tracing settings
version	Prints the version

Administration commands

Commands are useful for administrators of a Hyperwave cluster.

Command	Description
daemonlog	Gets/sets the log level

archive

Sergey Ostapov

To Table of Contents

Creates a Hadoop archive. More information on working with archives can be found at Hadoop Archives Guide.

The usage is as follows:

```
$ hadoop archive -archiveName name -p <parent> [-r <replication factor>] <src>* <dest>
```

Arguments

-archiveName	The name of an *.har* archive to create
-p	The relative path to which the files should be archived to. For example: `-p /foo/bar a/b/c e/f/g` where `/foo/bar` is the parent path and `a/b/c`, `e/f/g` are relative paths to parent.
-r	Sets the desired replication factor. If this optional argument is not specified, a replication factor of 3 is used.

Example:

```
$ hadoop archive -archiveName zoo.har -p /foo/bar -r 3 /outputdir
```

checknative

Sergey Ostapov

To Table of Contents

Checks the availability of the Hadoop native code. See Native Libraries Guide for more information. By default, this command only checks the availability of *libhadoop*.

The usage is as follows:

```
$ hadoop checknative [-a] [-h]
```

Arguments

-a	Checks all libraries are available
-h	Prints help

CLASSNAME

Sergey Ostapov

To Table of Contents

Runs a Java class named `ClassName`. The class must be a part of a package.

The usage is as follows:

```
$ hadoop classname <ClassName>
```

classpath

Sergey Ostapov

To Table of Contents

Prints the classpath needed to get the Hadoop JAR and the required libraries.
If called without arguments, prints the classpath set up by the command scripts, which is likely to contain wildcards in the classpath entries.

Additional options print the classpath after wildcard expansion or write the classpath into the manifest of a JAR file. The latter is useful in environments where wildcards cannot be used and the expanded classpath exceeds the maximum supported command line length.

The usage is as follows:

```
$ hadoop fs classpath [--glob |--jar <path> |-h |--help]
```

Arguments

--glob	Expands wildcards
--jar path	Writes classpath as manifest in JAR named `path`
-h, --help	Prints help

conftest

Sergey Ostapov

To Table of Contents

Validates configuration XML files. If the `-conffile` option isn’t specified, the *.xml* files under *${HADOOP_CONF_DIR}* will be verified. If specified, that path will be verified.

You can specify either a file or a directory, if a directory is specified, the files in that directory whose names end with *.xml* will be verified. You can specify `-conffile` option multiple times.

The validation is fairly minimal: the XML is parsed and duplicated, empty property names are checked. The command does not support XInclude. If you are using that to pull in configuration items, it will declare the XML file invalid.

The usage is as follows:

```
$ hadoop conftest [-conffile <path>]...
```

Arguments

-conffile	A path to a configuration file or directory to validate
-h, --help	Prints help

Example:

```
$ hadoop conftest
$ hadoop conftest -conffile /etc/hadoop/conf/core-site.xml
```

credential

Sergey Ostapov

To Table of Contents

Manages credentials, passwords, and secrets within credential providers.

The CredentialProvider API in Arenadata Hyperwave allows for the separation of applications and how they store their passwords/secrets.

In order to indicate particular provider type and location, use the `hadoop.security.credential.provider.path` parameter in *core-site.xml* or use the command line option `-provider` on each of the following commands.

This provider path is a comma-separated list of URLs that indicates the type and location of list of providers that should be consulted.

For example, the path *user:///jceks://file/tmp/test.jceks,jceks://hdfs@nn1.example.com/my/path/test.jceks* indicates that:

- the current user's credentials file should be addressed via the User Provider;
- the local file */tmp/test.jceks* is a Java Keystore Provider;
- the HDFS file *nn1.example.com/my/path/test.jceks* is also a store for Java Keystore Provider.

The `credential` command is often used for provisioning passwords or secrets to a particular credential store provider.

In order to explicitly indicate which provider store to use, the the `-provider` option.
Otherwise, for the given path of multiple providers, the first non-transient provider will be used. This may or may not be the one that you want.

Providers frequently require a password or other secrets.
If a provider requires a password and is unable to find one, it will use the default password and emit a warning message that the default password is being used.

If the `-strict` flag is supplied, the warning message becomes an error message and the command returns immediately with an error status.

The usage is as follows:

```
$ hadoop credential <subcommand> [options]
```

Arguments

print [-alias alias] filename [filename2 ...]	Prints the token fields contained in *filename* (*filename2*, and so on). If `alias` is specified, prints only the tokens matching alias. Otherwise, prints all tokens
get URL *[-service scheme]* *[-format (java|protobuf)]* *[-alias alias]* *[-renewer renewer]* *filename | filename [filename2 ...]*	Fetches tokens from a service using a URL and places it in the file. The URL is required and must be the first parameter. `URL` is the service URL, e.g. *hdfs://localhost:9000*. `alias` will overwrite the service field in the token. It's intended for hosts that have external and internal names, e.g. *firewall.com:14000* filename should come last and is the name of the token file. It will be created if it does not exist. Otherwise, token(s) are added to an existing file. The `-service` flag should only be used with a URL that starts with http or https. The following are equivalent: *hdfs://localhost:9000/* vs. *http://localhost:9000 -service hdfs*
append *[-format (java|protobuf)]* *filename filename2 [filename3 ...]*	Appends the contents of the first `N` files to the last file. When tokens with common service fields are present in multiple files, earlier file tokens are overwritten. Thus, tokens present in the last file are always preserved
remove -alias alias *[-format (java|protobuf)]* *filename [filename2 ...]*	Removes the tokens matching `alias` from each file specified and writes out each file using the specified format. The `alias` argument must be specified
cancel -alias alias *[-format (java|protobuf)]* *filename [filename2 ...]*	Acts similarly to `remove`, except the tokens are also cancelled using the service specified in the token object. The `alias` argument must be specified
renew -alias alias *[-format (java|protobuf)]* *filename [filename2 ...]*	For each file specified, renews the tokens matching `alias` and writes out each file using the specified format. The `alias` argument must be specified
import base64 *[-alias alias]* *filename*	Imports a Base64 token. The `alias` argument will overwrite the service field in the token

Example:

```
$ hadoop credential list -provider jceks://file/tmp/test.jceks
```

distch

Sergey Ostapov

To Table of Contents

Changes ownership and permissions of several files at once.

The usage is as follows:

```
$ hadoop distch [-f urilist_url] [-i] [-log logdir] path:owner:group:permissions
```

Arguments

-f <urilist_uri>	Uses the list at `<urilist_uri>` as a source list
-i	Ignores failures
-log	Sets a directory to log the output

distcp

Sergey Ostapov

To Table of Contents

Used for large inter/intra-cluster copying. It uses MapReduce to effect its distribution, error handling, recovery, and reporting. It expands a list of files/directories into input to map tasks, each of which will copy a partition of the files specified in the source list.

More information can be found at Hadoop DistCp Guide.

The usage is as follows:

```
$ hadoop distcp [OPTIONS] <src> <dst>
```

Arguments

-append	Reuses existing data in target files and appends new data to them if possible
-async	Defines whether the `distcp` execution should be blocking
-atomic	Commits all changes or none
-bandwidth <arg>	Specifies bandwidth per map in MB, accepts bandwidth as a fraction
-blocksperchunk <arg>	If set to a positive value, files with more blocks than this value will be split into chunks of <blocksperchunk> blocks to be transferred in parallel, and reassembled on the destination. By default, <blocksperchunk> is 0 and the files will be transmitted in their entirety without splitting. This switch is only applicable when the source file system implements the `getBlockLocations` method and the target file system implements the `concat` method.
-copybuffersize <arg>	The size of the copy buffer to use (in bytes). Defaults to 8192
-delete	Deletes those files on target that are missing in source. Delete is applicable only with `update` or `overwrite` options
-diff <arg>	Uses the snapshot diff report to identify the difference between source and target
-f <arg>	Specifies a list of files to copy
-filters <arg>	The path to a file containing a list of strings for paths to be excluded from the copy
-i	Ignores failures during copy
-log <arg>	Specifies a directory on DFS where `distcp` execution logs are saved
-m <arg>	The maximum number of concurrent maps to use for copy
-numListstatusThreads <arg>	The number of threads to use for building file listing (max 40)
-overwrite	Overwrites target files unconditionally, even if they exist
-p <arg>	Preserves status (rbugpcaxt)(replication, block-size, user, group, permission, checksum-type, ACL, XATTR, timestamps). If `-p` is specified with no <arg>, then preserves replication, block size, user, group, permission, checksum type, and timestamps.
-rdiff <arg>	Use target snapshot diff reports to identify changes made on target
-skipcrccheck	Whether to skip CRC checks between source and target paths.
-strategy <arg>	The copy strategy to use. The default is dividing work based on file sizes
-tmp <arg>	Intermediate work path to be used for atomic commits
-update	Updates the target, copying only missing files and overwriting the files that are different from source
-v	Logs additional info (path, size) to the SKIP/COPY log
-xtrack <arg>	Saves information about missing source files to the specified directory

Examples:

```
$ hadoop distcp hdfs://nn1:8020/foo/bar hdfs://nn2:8020/bar/foo
$ hadoop distcp hdfs://nn1:8020/foo/a hdfs://nn1:8020/foo/b hdfs://nn2:8020/bar/foo
```

dtutil

Sergey Ostapov

To Table of Contents

A utility to fetch and manage Hadoop delegation tokens inside credentials files.

It’s intended to replace the simpler command fetchdt. There are multiple subcommands, each with their own flags and options.

For every subcommand that writes out to a file, the `-format` option specifies the internal format to use . `java` is the legacy format that matches `fetchdt`. The default is `protobuf`.

For every subcommand that connects to service, convenience flags are provided to specify a Kerberos principal name and a keytab file to use for authentication.

The usage is as follows:

```
$ hadoop dtutil [-keytab keytab_file -principal principal_name ] subcommand [-format (java|protobuf)] [-alias alias ] [-renewer renewer ] filename…
```

Arguments

print [-alias alias] filename [filename2 …]	Prints out the fields in the tokens contained in *filename* (and *filename2* …). If `alias` is specified, prints only tokens matching alias. Otherwise, prints all tokens
get URL [-service scheme] [-format (java|protobuf)] [-alias alias] [-renewer renewer] filename | filename [filename2 …]	Fetches tokens from a service at URL and places it in the file. `URL` is required and must immediately follow `get`. `URL` is the service URL, e.g. *hdfs://localhost:9000*. The `alias` argument will overwrite the service field in the token. It’s intended for hosts that have external and internal names, for example *firewall.com:14000*. The `filename` argument should come last and is the name of the token file. It will be created if it does not exist. Otherwise, token(s) are added to existing file. The `-service` flag should only be used with URLs that start with "http" or "https". The following are equivalent: *hdfs://localhost:9000/* and *http://localhost:9000 -service hdfs*
append [-format (java|protobuf)] filename filename2 [filename3 …]	Appends the contents of the first `N` files to the last file. When tokens with common service fields are present in multiple files, earlier files' tokens are overwritten. Thus, tokens present in the last file are always preserved
remove -alias alias [-format (java|protobuf)] filename [filename2 …]	From each file specified, removes the tokens matching `alias` and writes out each file using the specified format. The `alias` argument must be specified
cancel -alias alias [-format (java|protobuf)] filename [filename2 …]	Identical to `remove`, except the tokens are also cancelled using the service specified in the token object. The `alias` argument must be specified
renew -alias alias [-format (java|protobuf)] filename [filename2 …]	For each file specified, renews the tokens matching alias and writes out each file using specified format. The `alias` argument must be specified
import base64 [-alias alias] filename	Imports Base64 token. The `alias` argument overwrites the service field in the token

envvars

Sergey Ostapov

To Table of Contents

Displays computed Hadoop environment variables.

The usage is as follows:

```
$ hadoop envvars
```

fs

Sergey Ostapov

To Table of Contents

This command allows direct interaction with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, WebHDFS, S3 FS, and others.

Usage:

```
$ hadoop fs <args>
```

The arguments accepted by the `fs` command are described in the FileSystem shell section.

gridmix

Sergey Ostapov

To Table of Contents

Gridmix is a benchmark tool for Hyperwave cluster. See more details about Gridmix in the Gridmix Guide.

The usage without configuration parameters is as follows:

```
$ hadoop gridmix [-generate <size>] [-users <users-list>] <iopath> <trace>
```

Arguments

-R	Recursively through the directory structure

The usage with the configuration parameters is as follows:

```
$ hadoop gridmix \
  -Dgridmix.client.submit.threads=10 -Dgridmix.output.directory=foo \
  [-generate <size>] [-users <users-list>] <iopath> <trace>
```

jar

Sergey Ostapov

To Table of Contents

Runs a JAR file. Use YARN jar to launch YARN applications instead.

The usage is as follows:

```
$ hadoop jar <jar> [mainClass] args...
```

jnipath

Sergey Ostapov

To Table of Contents

Prints the computed java.library.path.

The usage is as follows:

```
$ hadoop jnipath
```

kerbname

Sergey Ostapov

To Table of Contents

Converts a named principal via the auth_to_local rules to the Arenadata Hyperwave username.

The usage is as follows:

```
$ hadoop kerbname
```

Example:

```
$ hadoop kerbname user@EXAMPLE.COM
```

kdiag

Sergey Ostapov

To Table of Contents

Diagnoses Kerberos Problems.

The usage is as follows:

```
$ hadoop kdiag
```

key

Sergey Ostapov

To Table of Contents

Manages keys via the KeyProvider.
For more details on KeyProviders, see Transparent Encryption Guide.

Providers frequently require a password/other secret.
If a provider requires a password and is unable to find one, it uses a default password and emits a warning message that the default password is being used.

If the `strict` flag is provided, the warning message becomes an error and the command returns immediately with an error status.

> **NOTE**
>
> - Some KeyProviders (for example, `org.apache.hadoop.crypto.key.JavaKeyStoreProvider`) don't support uppercase key names.
> - Some KeyProviders don't directly execute a key deletion (for example, perform a soft-deleting instead, or delay the actual deletion, to prevent mistake). In such cases, one may encounter errors when creating/deleting a key with the same name after deleting it. Please check the underlying KeyProvider for details.

The usage is as follows:

```
$ hadoop key <args> [options]
```

Arguments

create <keyname> [-cipher cipher] [-size size] [-description description] [-attr attribute=value] [-provider provider] [-strict] [-help]	Creates a new key for the name specified by the `<keyname>` argument within the provider specified by `provider`. The `strict` flag causes the command to fail if the provider uses a default password. You can specify a cipher with the `cipher` argument. The default cipher is currently `AES/CTR/NoPadding`. The default keysize is `128`. You may specify the required key length using the `size` argument. Arbitrary attributes `attribute=value` may be specified using the `attr` argument. The `attr` argument may be specified multiple times, once per attribute
roll <keyname> [-provider provider] [-strict] [-help]	Creates a new version for the specified key within the provider indicated using the `provider` argument. The `strict` flag causes the command to fail if the provider uses a default password
delete <keyname> [-provider provider] [-strict] [-f] [-help]	Deletes all versions of the key specified by the `<keyname>` argument from within the provider specified by `provider`. The `strict` flag causes the command to fail if the provider uses a default password. The command asks for user confirmation unless `f` is specified
list [-provider provider] [-strict] [-metadata] [-help]	Displays the key names contained within a particular provider as configured in *core-site.xml* or specified with the `provider` argument. The `strict` flag causes the command to fail if the provider uses a default password The `metadata` argument displays the metadata
check <keyname> [-provider provider] [-strict] [-help]	Checks password of the `<keyname>` contained within a particular provider as configured in *core-site.xml* or specified with the `provider` argument. The `strict` flag causes the command to fail if the provider uses a default password
help	Prints the command usage details

kms

Sergey Ostapov

To Table of Contents

Allows to manage KMS (Key Management Server).

The usage is as follows:

```
$ hadoop kms [command]
```

Example:

```
$ hadoop --daemon start kms
```

trace

Sergey Ostapov

To Table of Contents

Allows to view and modify Hadoop tracing settings.
For more information on tracing, see Tracing Guide.

The usage is as follows:

```
$ hadoop trace
```

version

Sergey Ostapov

To Table of Contents

Prints the current Hadoop version.

The usage is as follows:

```
$ hadoop version
```

daemonlog

Sergey Ostapov

To Table of Contents

Gets/sets the log level for a log identified by a qualified class name in the daemon dynamically.

By default, this command sends an HTTP request, but this can be overridden by using `-protocol https` to send an HTTPS request.

Note that the setting isn’t permanent and will reset when the daemon is restarted. This command works by sending an HTTP/HTTPS request to the daemon’s internal Jetty servlet, so it supports the following daemons:

- Common
  - key management server
- HDFS
  - name node
  - secondary name node
  - data node
  - journal node
  - HttpFS server
- YARN
  - resource manager
  - node manager
  - Timeline server

The usage is as follows:

```
$ hadoop daemonlog -getlevel <host:port> <classname> [-protocol (http|https)]
$ hadoop daemonlog -setlevel <host:port> <classname> <level> [-protocol (http|https)]
```

Arguments

-getlevel host:port classname [-protocol (httphttps)]	Prints the log level of the log identified by a qualified classname, in the daemon running at `host:port`. The `-protocol` flag specifies the protocol for connection
-setlevel host:port classname level [-protocol (httphttps)]	Sets the log level of the log identified by a qualified classname, in the daemon running at `host:port`. The `-protocol` flag specifies the protocol for connection

Examples:

```
$ bin/hadoop daemonlog -setlevel 127.0.0.1:9870 org.apache.hadoop.hdfs.server.namenode.NameNode DEBUG
$ bin/hadoop daemonlog -getlevel 127.0.0.1:9871 org.apache.hadoop.hdfs.server.namenode.NameNode DEBUG -protocol https
```

HDFS CLI

Sergey Ostapov

Contents

To Table of Contents

- Overview
- User commands
- Administration commands
- Debug commands

Overview

This section includes the reference documentation for the HDFS shell CLI. Using `hdfs` commands without any arguments outputs a description for all these commands. The usage is as follows:

```
$ hdfs [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [COMMAND_OPTIONS]
```

Hadoop has an option parsing framework that employs parsing generic options as well as running classes.

Hadoop shell basic structure

SHELL_OPTIONS	The common set of shell options
GENERIC_OPTIONS	The common set of options supported by multiple commands
COMMAND_OPTIONS	Various commands with their options are described in the following sections

User commands

These commands are helpful for users of a Hyperwave cluster.

Command	Description
classpath	Prints the class path needed to get the Hadoop JAR and the required libraries
dfs	Runs a filesystem command on the file system supported in Hadoop
envvars	Displays computed Hadoop environment variables
fetchdt	Gets Delegation Token from a NameNode
fsck	Runs the HDFS filesystem checking utility
getconf	Gets configuration information from the configuration directory, post-processing
groups	Returns the group information given one or more usernames
httpfs	Runs HttpFS server, the HDFS HTTP Gateway
lsSnapshottableDir	Gets the list of snapshot table directories
jmxget	Dumps JMX information from a service
oev	Hadoop offline edits viewer
oiv	Hadoop Offline Image Viewer for image files
oiv_legacy	Hadoop Offline Image Viewer for older versions of Hadoop
snapshotDiff	Determines the difference between HDFS snapshots
version	Prints the version

Administration commands

These commands are useful for administrators of a Hyperwave cluster.

Command	Description
balancer	Runs a cluster balancing utility
cacheadmin	Interacts with cache pools and directives via the hdfs cacheadmin subcommand
crypto	Creates a new encryption zone
datanode	Runs an HDFS DataNode
dfsadmin	Runs an HDFS dfsadmin client
dfsrouter	Runs the DFS router
dfsrouteradmin	Manages Router-based federation
diskbalancer	Runs the disk balancer CLI
ec	Runs the ErasureCoding CLI
haadmin	Gets access to some additional commands to administer the HA HDFS cluster
journalnode	Starts a JournalNode for use with HDFS HA with QJM
mover	Runs the data migration utility
namenode	Runs the NameNode
nfs3	Starts the NFS3 gateway for use with the HDFS NFS3 Service
portmap	Starts the RPC portmap for use with the HDFS NFS3 Service
secondarynamenode	Runs the HDFS secondary NameNode
storagepolicies	Lists out all/gets/sets/unsets storage policies
zkfc	Starts a ZooKeeper Failover Controller process for use with HDFS HA with QJM

Debug commands

These commands are used to help administrators debug HDFS issues.

Command	Description
computeMeta	Computes HDFS metadata from block files
recoverLease	Recovers the lease on the specified path
verifyMeta	Verifies HDFS metadata and block files

classpath

Sergey Ostapov

To Table of Contents

Prints the class path needed to get the Hadoop JAR and the required libraries. If called without arguments, then prints the classpath set up by the command scripts, which is likely to contain wildcards in the classpath entries.

Additional options print the classpath after wildcard expansion or write the classpath into the manifest of a JAR-file. The latter is useful in environments where wildcards cannot be used and the expanded classpath exceeds the maximum supported command line length.

The usage is as follows:

```
$ hdfs classpath [--glob |--jar <path> |-h |--help]
```

Arguments

--glob	Expands wildcards
--jar <path>	Writes the classpath as manifest in a JAR named `<path>`
-h, --help	Prints detailed information about the command

dfs

Sergey Ostapov

To Table of Contents

Runs a filesystem command on the file system supported in Arenadata Hyperwave HDFS.

The usage is as follows:

```
$ hdfs dfs [COMMAND [COMMAND_OPTIONS]]
```

Example:

```
$ hdfs dfs -cat file:///file3 /user/hadoop/file4
$ hdfs dfs -count -q hdfs://nn1.example.com/file1
$ hdfs dfs -cp /user/hadoop/file1 /user/hadoop/file2
```

envvars

Sergey Ostapov

To Table of Contents

Displays computed Hadoop environment variables.

The usage is as follows:

```
$ hdfs envvars
```

Example:

```
$ hdfs envvars
JAVA_HOME='/usr/lib/jvm/jre-openjdk'
HADOOP_HDFS_HOME='/usr/lib/hadoop-hdfs'
HDFS_DIR='./'
HDFS_LIB_JARS_DIR='lib'
HADOOP_CONF_DIR='/etc/hadoop/conf'
HADOOP_TOOLS_HOME='/usr/lib/hadoop'
HADOOP_TOOLS_DIR='share/hadoop/tools'
HADOOP_TOOLS_LIB_JARS_DIR='share/hadoop/tools/lib'
```

fetchdt

Sergey Ostapov

To Table of Contents

Gets Delegation Token from a NameNode.

The usage is as follows:

```
$ hdfs fetchdt <opts> <token_file_path>
```

Arguments

--webservice NN_Url	Url to contact NN on (starts with http or https)
--renewer name	Name of the delegation token renewer
--cancel	Cancels the delegation token
--renew	Renews the delegation token. Delegation token must have been fetched using the `-renewer` name option
--print	Prints the delegation token
token_file_path	File path to store the token into

fsck

Sergey Ostapov

To Table of Contents

Runs the HDFS checking utility. This command allows to detect problems with various files, for example, missing file blocks or under-replicated blocks.

Unlike a traditional `fsck` utility for native file systems, this command does not fix the errors it detects. Normally NameNode automatically corrects most of the recoverable failures. By default, `fsck` ignores open files but provides an option to select all files during the analysis.

The usage is as follows:

```
$ hdfs fsck <path>
  [-list-corruptfileblocks |
  [-move | -delete | -openforwrite]
  [-files [-blocks [-locations | -racks | -replicaDetails | -upgradedomains]]]
  [-includeSnapshots][-storagepolicies] [-maintenance]
  [-blockId <blk_Id>] [-replicate]
```

Arguments

path	Starts checking from this path
-delete	Deletes corrupted files
-files	Prints out files being checked
-files -blocks	Prints out the block report
-files -blocks -locations	Prints out locations for every block
-files -blocks -racks	Prints out network topology for DataNode locations
-files -blocks -replicaDetails	Prints out each replica details
-files -blocks -upgradedomains	Prints out upgrade domains for every block
-move	Moves corrupted files to */lost+found*
-openforwrite	Prints out files opened for write
-storagepolicies	Prints out storage policy summary for the blocks
-storagepolicies	Prints out storage policy summary for the blocks
-maintenance	Prints out maintenance state node details
-blockId	Prints out information about the block
-replicate	Initiates replication work to make mis-replicated blocks satisfy block placement policy

Example:

```
$ hdfs fsck myDir/file_test.csv -files -blocks -locations
```

getconf

Sergey Ostapov

To Table of Contents

Gets configuration information from the configuration directory, post-processing.

The usage is as follows:

```
$ hdfs getconf -namenodes
$ hdfs getconf -secondaryNameNodes
$ hdfs getconf -backupNodes
$ hdfs getconf -journalNodes
$ hdfs getconf -includeFile
$ hdfs getconf -excludeFile
$ hdfs getconf -nnRpcAddresses
$ hdfs getconf -confKey [key]
```

Arguments

-namenodes	Gets the list of NameNodes in the cluster
-secondaryNameNodes	Gets the list of secondary NameNodes in the cluster
-backupNodes	Gets the list of backup nodes in the cluster
-journalNodes	Gets the list of journal nodes in the cluster
-includeFile	Gets the include file path that defines the DataNodes that can join the cluster
-excludeFile	Gets the exclude file path that defines the DataNodes that need to decommissioned
-nnRpcAddresses	Gets the NameNode RPC addresses
-confKey [key]	Gets a specific key from the configuration

groups

Sergey Ostapov

To Table of Contents

Returns the group information given one or more usernames.

The usage is as follows:

```
$ hdfs groups [username ...]
```

httpfs

Sergey Ostapov

To Table of Contents

Runs HttpFS server, the HDFS HTTP Gateway.

The usage is as follows:

```
$ hdfs httpfs
```

lsSnapshottableDir

Sergey Ostapov

To Table of Contents

Gets the list of snapshottable directories. If run as a superuser, returns all snapshottable directories. Otherwise, returns those directories that are owned by the current user.

The usage is as follows:

```
$ hdfs lsSnapshottableDir [-help]
```

Arguments

-help	Prints detailed information about the command

jmxget

Sergey Ostapov

To Table of Contents

Dumps JMX information from a service.

The usage is as follows:

```
$ hdfs jmxget [-localVM ConnectorURL | -port port | -server mbeanserver | -service service]
```

Arguments

-help	Prints detailed information about the command
-localVM ConnectorURL	Connects to the VM on the same machine
-port mbean server port	Specifies mbean server port, if missing it will try to connect to MBean Server in the same VM
-server	Specifies mbean server (`localhost` by default)
-service NameNode|DataNode	Specifies jmx service (`NameNode` by default)

oev

Sergey Ostapov

To Table of Contents

Hadoop offline edits viewer.
Details on using the utility can be found at Offline Edits Viewer Guide.

The usage is as follows:

```
$ hdfs oev [OPTIONS] -i INPUT_FILE -o OUTPUT_FILE
```

Required arguments

-i,--inputFile arg	Edits file to process, `xml` (case insensitive) extension means XML format, any other filename means the binary format
-o,--outputFile arg	A name for the output file. If the specified file exists, it will be overwritten, the format of the file is determined by `-p` option

Optional arguments

-f,--fix-txids	Renumbers the transaction IDs in the input, so that there are no gaps or invalid transaction IDs
-h,--help	Displays usage information
-r,--recover	When reading binary edit logs, uses the recovery mode. This allows to skip corrupt parts of the edit log
--p,--processor arg	Selects the type of processor to apply for an image file, currently supported processors are: ▪ `binary` (native binary format that Hadoop uses); ▪ `xml` (default, XML format); ▪ `stats` (prints statistics about edits file).
-v,--verbose	Produces a more verbose output, prints the input and output filenames, for processors that write to a file, output their names. For large image files this will dramatically increase the processing time (default is `false`)

Example:

```
$ hdfs oev -p xml -i edits -o edits.xml
```

oiv

Sergey Ostapov

To Table of Contents

Hadoop Offline Image Viewer for image files.

The usage is as follows:

```
$ hdfs oiv [OPTIONS] -i INPUT_FILE
```

Required arguments

-i,--inputFile INPUT_FILE	Specifies the input fsimage file (or XML file, if ReverseXML processor is used) to process

Optional arguments

-o,--outputFile output file	Specifies the output filename, if the specified output processor generates one. If the specified file already exists, it is silently overwritten. If the input file is an XML file, it also creates an *<outputFile>.md5*
-p,--processor processor	Specifies the image processor to apply against the image file. Currently valid options are `Web` (default), `XML`, `Delimited`, `FileDistribution` and `ReverseXML`
-addr address	Specifies the address(`host:port`) to listen. This option is used with Web processor
-maxSize size	Specifies the range [0, maxSize] of file sizes to be analyzed in bytes (`128GB` by default). This option is used with FileDistribution processor
-step size	Specifies the granularity of the distribution in bytes (`2MB` by default). This option is used with FileDistribution processor
-format	Formats the output result to human-readable fashion rather than a number of bytes. This option is used with FileDistribution processor
-delimiter arg	Delimiting string to use with Delimited processor
-t,--temp temporary dir	Uses temporary dir to cache intermediate result to generate Delimited outputs. If not set, Delimited processor constructs the namespace in memory before outputting text
-h,--help	Displays the tool usage, help information and exits

oiv_legacy

Sergey Ostapov

To Table of Contents

Hadoop Offline Image Viewer for older versions of Hadoop. See oiv_legacy Command for more info.

The usage is as follows:

```
$ hdfs oiv_legacy [OPTIONS] -i INPUT_FILE -o OUTPUT_FILE
```

Arguments

-i,--inputFile input file	Specifies the input fsimage file to process. Required
-o,--outputFile output file	Specifies the output filename, if the specified output processor generates one. If the specified file already exists, it is silently overwritten. Required
-p|--processor processor	Specifies the image processor to apply against the image file. Valid options are Ls (default), XML, Delimited, Indented, FileDistribution and NameDistribution
-maxSize size	Specifies the range [0, maxSize] of file sizes to be analyzed in bytes (`128GB` by default). This option is used with FileDistribution processor
-step size	Specifies the granularity of the distribution in bytes (`2MB` by default). This option is used with FileDistribution processor
-format	Formats the output result in a human-readable fashion rather than a number of bytes. (false by default). This option is used with FileDistribution processor
-skipBlocks	Does not enumerate individual blocks within files. This may save processing time and outfile file space on namespaces with very large files. The Ls processor reads the blocks to correctly determine file sizes and ignores this option
-printToScreen	Pipes output of processor to console as well as specified file. On extremely large namespaces, this may increase processing time by an order of magnitude
-delimiter arg	When used in conjunction with the Delimited processor, replaces the default tab delimiter with the string specified by arg
-h|--help	Displays the tool usage and help information and exit

snapshotDiff

Sergey Ostapov

To Table of Contents

Determines the difference between HDFS snapshots.

The usage is as follows:

```
$ hdfs snapshotDiff <path> <fromSnapshot> <toSnapshot>
```

version

Sergey Ostapov

Prints the current HDFS version.

The usage is as follows:

```
$ hdfs version
```

balancer

Sergey Ostapov

To Table of Contents

Runs a cluster balancing utility. An administrator can simply press `Ctrl+C` to stop the rebalancing process.

NOTE

The blockpool policy is stricter than the DataNode policy.

The usage is as follows:

```
$ hdfs balancer
    [-policy <policy>]
    [-threshold <threshold>]
    [-exclude [-f <hosts-file> | <comma-separated list of hosts>]]
    [-include [-f <hosts-file> | <comma-separated list of hosts>]]
    [-source [-f <hosts-file> | <comma-separated list of hosts>]]
    [-blockpools <comma-separated list of blockpool ids>]
    [-idleiterations <idleiterations>]
    [-runDuringUpgrade]
    [-asService]
```

Arguments

-policy <policy>	Possible values: ▪ `DataNode` (default) — cluster is balanced if each DataNode is balanced; ▪ `Blockpool` — cluster is balanced if each block pool in each DataNode is balanced.
threshold <threshold> | <comma-separated list of hosts>	A percentage of disk capacity. This overwrites the default threshold
-exclude -f <hosts-file> | <comma-separated list of hosts>	Excludes the specified DataNodes from being balanced by the balancer
-include -f <hosts-file> | <comma-separated list of hosts>	Includes only the specified DataNodes to be balanced by the balancer
-source -f <hosts-file> | <comma-separated list of hosts>	Picks only the specified DataNodes as source nodes
-blockpools <comma-separated list of blockpool ids>	Runs the balancer only on blockpools included in this list
-idleiterations <iterations>	The maximum number of idle iterations before exit. This overwrites the default `idleiterations`
-runDuringUpgrade	Specifies whether to run the balancer during an ongoing HDFS upgrade. This is not usually desired since it will not affect used space on over-utilized machines
-asService	Runs the balancer as a long-running service
-h, --help	Displays the tool usage and help information

cacheadmin

Sergey Ostapov

Contents

To Table of Contents

- Cache directive commands
  - addDirective
  - removeDirective
  - listDirectives
- Cache pool commands
  - addPool
  - modifyPool
  - removePool
  - listPools
  - help

On the command-line, you can interact with cache pools and directives via the `hdfs cacheadmin` subcommand.

Cache directives are identified by a unique, non-repeating 64-bit integer ID. IDs will not be reused even if a cache directive is later removed. Cache pools are identified by a unique string name.

Cache directive commands

addDirective

Adds a new cache directive.

```
$ hdfs cacheadmin -addDirective -path <path> -pool <pool-name> [-force] [-replication <replication>] [-ttl <time-to-live>]
```

Arguments

<path>	A path to the cache. A path can be a directory or a file
<pool-name>	A pool to which the directive will be added. The user should have the write permission on the cache pool in order to add new directives
-force	Skips checking of cache pool resource limits
<replication>	The cache replication factor to use
<time-to-live>	Specifies the time period for the directive to be valid. Can be specified in minutes, hours, and days, e.g. `30m`, `4h`, `2d`. If unspecified, the directive never expires

removeDirective

Removes every cache directive with the specified path.

```
$ hdfs cacheadmin -removeDirective <id>
```

Arguments

<path>	The path of the cache directives to remove. You must have the write permission on the pool of the directive in order to remove it

listDirectives

Lists cache directives.

```
$ hdfs cacheadmin -listDirectives [-stats] [-path <path>] [-pool <pool>]
```

Arguments

<path>	Lists only cache directives with this path. If there is a cache directive for a path in a cache pool that the user doesn’t have read access for, it will not be listed
<pool>	Lists only path cache directives in that pool
-stats	Lists path-based cache directive statistics

Cache pool commands

addPool

Adds a new cache pool.

```
$ hdfs cacheadmin -addPool <name> [-owner <owner>] [-group <group>] [-mode <mode>] [-limit <limit>] [-maxTtl <maxTtl>]
```

Arguments

<name>	A name for the new pool.
<owner>	The pool owner’s username. Defaults to the current user
<group>	The group of the pool. Defaults to the primary group name of the current user
<mode>	UNIX-style permissions for the pool. Permissions are specified in octal, e.g. `0755`
<limit>	The maximum number of bytes that can be cached by directives in this pool, in aggregate. By default, no limit is set
<maxTtl>	The maximum allowed time-to-live for directives being added to the pool. Can be specified in seconds, minutes, hours, and days, e.g. `120s`, `30m`, `4h`, `2d`. By default, no maximum is set. The value of `never` specifies that there is no limit

modifyPool

Modifies the metadata of an existing cache pool.

```
$ hdfs cacheadmin -modifyPool <name> [-owner <owner>] [-group <group>] [-mode <mode>] [-limit <limit>] [-maxTtl <maxTtl>]
```

Arguments

<name>	The name of the pool to modify
<owner>	The pool owner’s username
<group>	The name of the group of the pool
<mode>	Unix-style permissions of the pool in octal
<limit>	The maximum number of bytes that can be cached by this pool
<maxTtl>	The maximum allowed time-to-live for directives being added to the pool

removePool

Removes a cache pool. This also uncaches paths associated with the pool.

```
$ hdfs cacheadmin -removePool <name>
```

Arguments

<name>	The name of the cache pool to remove

listPools

Displays information about one or more cache pools, e.g. name, owner, group, permissions, etc.

```
hdfs cacheadmin -listPools [-stats] [<name>]
```

Arguments

-stats	Displays additional cache pool statistics
<name>	If specified, lists only the named cache pool

help

Displays detailed information about the command.

```
$ hdfs cacheadmin -help <command-name>
```

Arguments

<command-name>	Command to display the information about
help	If no command is specified, prints detailed information about all commands

crypto

Sergey Ostapov

Contents

To Table of Contents

createZone
listZones
provisionTrash
getFileEncryptionInfo
reencryptZone
listReencryptionStatus

For more information on the crypto command-line interface, see HDFS Transparent Encryption Documentation.

The usage is as follows:

```
$ hdfs crypto -createZone -keyName <keyName> -path <path>
$ hdfs crypto -listZones
$ hdfs crypto -provisionTrash -path <path>
$ hdfs crypto -help <command-name>
```

createZone

Creates a new encryption zone.

```
[-createZone -keyName <keyName> -path <path>]
```

Arguments

path	The path of the encryption zone to create. It must be an empty directory. A trash directory is provisioned under this path
keyName	Name of the key to use for the encryption zone. Uppercase key names are unsupported

listZones

Lists all encryption zones. Requires superuser permissions.

```
[-listZones]
```

provisionTrash

Provision a trash directory for an encryption zone.

```
[-provisionTrash -path <path>]
```

Arguments

path	The path to the root of the encryption zone

getFileEncryptionInfo

Gets encryption information from a file. This can be used to find whether a file is being encrypted, and the key name/key version used to encrypt it.

```
[-getFileEncryptionInfo -path <path>]
```

Arguments

path	The path of the file to get encryption information

reencryptZone

Re-encrypts an encryption zone by iterating through the encryption zone and calls the KeyProvider's `reencryptEncryptedKeys` interface to batch-re-encrypt all files with the latest version encryption zone key in the key provider. Requires superuser permissions.

```
[-reencryptZone <action> -path <zone>]
```

NOTE

The re-encryption does not apply to snapshots, due to snapshots immutable nature.

Arguments

action	The re-encrypt action to perform. Must be either `-start` or `-cancel`
path	The path to the root of the encryption zone

Re-encryption is a NameNode-only operation in HDFS, so could potentially put intensive load to the NameNode. The following configurations can be changed to control the stress on the NameNode, depending on the acceptable throughput impact to the cluster.

dfs.namenode.reencrypt.batch.size	The number of EDEKs in a batch to be sent to the KMS for re-encryption. Each batch is processed when holding the name system read/write lock, with throttling happening between batches. See configs below
dfs.namenode.reencrypt.throttle.limit.handler.ratio	The ratio of read locks to be held during re-encryption. `1.0` means no throttling. `0.5` means re-encryption can hold the readlock at most 50% of its total processing time. Negative value or `0` are invalid
dfs.namenode.reencrypt.throttle.limit.updater.ratio	The ratio of write locks to be held during re-encryption. `1.0` means no throttling. `0.5` means re-encryption can hold the writelock at most 50% of its total processing time. Negative value or `0` are invalid

listReencryptionStatus

List re-encryption information for all encryption zones. Requires superuser permissions.

```
[-listReencryptionStatus]
```

datanode

Sergey Ostapov

To Table of Contents

Runs a HDFS DataNode.

The usage is as follows:

```
$ hdfs datanode [-regular | -rollback | -rollingupgrade rollback]
```

Arguments

-regular	Normal DataNode startup (default)
-rollback	Rollbacks the DataNode to the previous version
-rollingupgrade rollback	Rollbacks a rolling upgrade operation

dfsadmin

Sergey Ostapov

To Table of Contents

Runs a HDFS dfsadmin.

The usage is as follows:

```
$ hdfs dfsadmin [-report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]]
$ hdfs dfsadmin [-safemode enter | leave | get | wait | forceExit]
$ hdfs dfsadmin [-saveNamespace [-beforeShutdown]]
$ hdfs dfsadmin [-rollEdits]
$ hdfs dfsadmin [-restoreFailedStorage true |false |check]
$ hdfs dfsadmin [-refreshNodes]
$ hdfs dfsadmin [-setQuota <quota> <dirname>...<dirname>]
$ hdfs dfsadmin [-clrQuota <dirname>...<dirname>]
$ hdfs dfsadmin [-setSpaceQuota <quota> [-storageType <storagetype>] <dirname>...<dirname>]
$ hdfs dfsadmin [-clrSpaceQuota [-storageType <storagetype>] <dirname>...<dirname>]
$ hdfs dfsadmin [-finalizeUpgrade]
$ hdfs dfsadmin [-rollingUpgrade [<query> |<prepare> |<finalize>]]
$ hdfs dfsadmin [-upgrade [query | finalize]]
$ hdfs dfsadmin [-refreshServiceAcl]
$ hdfs dfsadmin [-refreshUserToGroupsMappings]
$ hdfs dfsadmin [-refreshSuperUserGroupsConfiguration]
$ hdfs dfsadmin [-refreshCallQueue]
$ hdfs dfsadmin [-refresh <host:ipc_port> <key> [arg1..argn]]
$ hdfs dfsadmin [-reconfig <namenode|datanode> <host:ipc_port> <start |status |properties>]
$ hdfs dfsadmin [-printTopology]
$ hdfs dfsadmin [-refreshNamenodes datanodehost:port]
$ hdfs dfsadmin [-getVolumeReport datanodehost:port]
$ hdfs dfsadmin [-deleteBlockPool datanode-host:port blockpoolId [force]]
$ hdfs dfsadmin [-setBalancerBandwidth <bandwidth in bytes per second>]
$ hdfs dfsadmin [-getBalancerBandwidth <datanode_host:ipc_port>]
$ hdfs dfsadmin [-fetchImage <local directory>]
$ hdfs dfsadmin [-allowSnapshot <snapshotDir>]
$ hdfs dfsadmin [-disallowSnapshot <snapshotDir>]
$ hdfs dfsadmin [-shutdownDatanode <datanode_host:ipc_port> [upgrade]]
$ hdfs dfsadmin [-evictWriters <datanode_host:ipc_port>]
$ hdfs dfsadmin [-getDatanodeInfo <datanode_host:ipc_port>]
$ hdfs dfsadmin [-metasave filename]
$ hdfs dfsadmin [-triggerBlockReport [-incremental] <datanode_host:ipc_port> [-namenode <namenode_host:ipc_port>]]
$ hdfs dfsadmin [-listOpenFiles [-blockingDecommission] [-path <path>]]
$ hdfs dfsadmin [-help [cmd]]
```

Arguments

--report [-live] [-dead] [-decommissioning] [-enteringmaintenance] [-inmaintenance]	Reports basic filesystem information and statistics. The `dfs` usage can be different from `du` usage, because it measures raw space used by replication, checksums, snapshots etc., on all the DNs. Optional flags may be used to filter the list of displayed DataNodes
safemode enter|leave|get|wait|forceExit	Safe mode maintenance command. Safe mode is a NameNode state in which it: - does not accept changes to a namespace (read-only); - does not replicate or delete blocks. Safe mode is entered automatically at NameNode startup, and the node leaves the safe mode automatically when the configured minimum percentage of blocks satisfies the minimum replication condition. If NameNode detects any anomaly then it will linger in safe mode till that issue is resolved. If that anomaly is the consequence of a deliberate action, then administrator can use `-safemode forceExit` to exit safe mode. The cases where `forceExit` may be required are: - NameNode metadata is not consistent. If a NameNode detects that metadata has been modified out of band and can cause data loss, then the NameNode enters the `forceExit` state. At that point a user can either restart the NameNode with correct metadata files or `forceExit` (if data loss is acceptable); - Rollback causes metadata to be replaced, and rarely it can trigger the safe mode `forceExit` state in NameNode. In that case you may proceed by issuing `-safemode forceExit`. Safe mode can also be entered manually, but then it can only be turned off manually as well.
-saveNamespace [-beforeShutdown]	Saves the current namespace into storage directories and resets the edits log. If the `beforeShutdown` option is given, the NameNode creates a checkpoint if and only if no checkpoint has been done during a time frame (a configurable number of checkpoint periods). This is usually used before shutting down the NameNode to prevent potential fsimage/editlog corruption
-rollEdits	Rolls the edit log on the active NameNode
-restoreFailedStorage true|false|check	Turns on/off automatic attempt to restore failed storage replicas. If a failed storage becomes available again the system attempts to restore edits and/or fsimage during the checkpoint
-refreshNodes	Re-reads the hosts and exclude files to update the set of DataNodes that are allowed to connect to the NameNode and those that should be decommissioned or recommissioned
-setQuota <quota> <dirname>…<dirname>	Sets the quota for each directory
-clrQuota <dirname>…<dirname>	Removes any name quota for each directory. Best effort for each directory, with faults reported if the directory does not exist, or it is a file. A directory may have no quota at all
-setSpaceQuota <quota> [-storageType <storagetype>] <dirname>…<dirname>	Sets the storage type quota to bytes of storage type specified for each directory
-clrSpaceQuota [-storageType <storagetype>] <dirname>…<dirname>	Removes storage type quota specified for each directory. Best effort for each directory, with faults reported if the directory does not exist, or it is a file. It is not a fault if the directory has no storage type quota on for storage type specified. The storage type specific quota is cleared when `-storageType` option is specified
-finalizeUpgrade	Finalizes upgrade of HDFS. DataNodes delete their previous version working directories, followed by NameNode doing the same. This completes the upgrade process
-rollingUpgrade [<query>|<prepare>|<finalize>]	Executes a rolling upgrade action: - `query` — queries the current rolling upgrade status; - `prepare` — prepares a new rolling upgrade; - `finalize` — finalizes the current rolling upgrade.
-upgrade query|finalize	Queries the current upgrade status Finalize upgrade of HDFS (equivalent to `-finalizeUpgrade`)
-refreshServiceAcl	Reloads the service-level authorization policy file
-refreshUserToGroupsMappings	Refreshes user-to-groups mappings
-refreshSuperUserGroupsConfiguration	Refreshes superuser proxy groups mappings
-refreshCallQueue	Reloads the call queue from config
-refresh <host:ipc_port> <key> [arg1..argn]	Triggers a runtime-refresh of the resource specified by `key` on `<host:ipc_port>`. All other args after are sent to the host
-reconfig <datanode |namenode> <host:ipc_port> <start|status|properties>	Starts reconfiguration or gets the status of an ongoing reconfiguration, or gets a list of reconfigurable properties. The second parameter specifies the node type
-printTopology	Prints a tree of the racks and their nodes as reported by the NameNode
-refreshNameNodes datanodehost:port	For the given DataNode, reloads the configuration files, stops serving the removed block-pools and starts serving new block-pools
-getVolumeReport datanodehost:port	For the given DataNode, get the volume report
-deleteBlockPool datanode-host:port blockpoolId [force]	If `force` is passed, block pool directory for the given blockpool ID on the given DataNode is deleted along with its contents, otherwise the directory is deleted only if it is empty. The command will fail if DataNode is still serving the block pool. Refer to refresh NameNodes to shutdown a block pool service on a DataNode
-setBalancerBandwidth <bandwidth in bytes per second>	Changes the network bandwidth used by each DataNode during HDFS block balancing. `<bandwidth>` is the maximum number of bytes per second that will be used by each DataNode. This value overrides the `fs.datanode.balance.bandwidthPerSec` parameter. The new value is not persistent on the DataNode
-getBalancerBandwidth <datanode_host:ipc_port>	Gets the network bandwidth for the given DataNode (in bytes per second). This is the maximum network bandwidth used by the DataNode during HDFS block balancing
-fetchImage <local directory>	Downloads the most recent fsimage from the NameNode and saves it in the specified local directory
-allowSnapshot <snapshotDir>	Allows snapshots of a directory to be created. If the operation completes successfully, the directory becomes snapshot-table
-disallowSnapshot <snapshotDir>	Disallows snapshots of a directory to be created. All snapshots of the directory must be deleted before disallowing snapshots
-shutdownDatanode <datanode_host:ipc_port> [upgrade]	Submits a shutdown request for the given DataNode
-evictWriters <datanode_host:ipc_port>	Makes the DataNode evict all clients that are writing a block. This is useful if decommissioning is hung due to slow writers
-getDatanodeInfo <datanode_host:ipc_port>	Gets the information about the given DataNode
-metasave filename	Saves NameNode’s primary data structures to filename in the directory specified by the `hadoop.log.dir` property. The filename will be overwritten if it exists. The filename will contain one line for each of the following: - DataNodes heart beating with NameNode; - Blocks waiting to be replicated; - Blocks currently being replicated; - Blocks waiting to be deleted.
-triggerBlockReport [-incremental] <datanode_host:ipc_port> [-namenode <namenode_host:ipc_port>]	Triggers a block report for the given DataNode. If `-incremental` is specified, it will be otherwise, it will be a full block report. If `-namenode <namenode_host:ipc_port>` is given, it only sends block report to a specified NameNode
-listOpenFiles [-blockingDecommission] [-path <path>]	Lists all open files currently managed by the NameNode along with client name and client machine accessing them. Open files list will be filtered by given type and path. Add `-blockingDecommission` option if you only want to list open files that are blocking the DataNode decommissioning
-help [cmd]	Displays help for the given command or all commands if none is specified

dfsrouter

Sergey Ostapov

To Table of Contents

Runs the DFS router. More information is available at HDFS Router-based Federation.

The usage is as follows:

```
$ hdfs dfsrouter
```

dfsrouteradmin

Sergey Ostapov

To Table of Contents

Commands for managing Router-based federation.

The usage is as follows:

```
$ hdfs dfsrouteradmin
    [-add <source> <nameservice1, nameservice2, ...> <destination> [-readonly] [-faulttolerant] [-order HASH|LOCAL|RANDOM|HASH_ALL]
-owner <owner> -group <group> -mode <mode>]
    [-update <source> [<nameservice1, nameservice2, ...> <destination>] [-readonly true|false] [-faulttolerant true|false] [-order
HASH|LOCAL|RANDOM|HASH_ALL] -owner <owner> -group <group> -mode <mode>]
    [-rm <source>]
    [-ls [-d] <path>]
    [-getDestination <path>]
    [-setQuota <path> -nsQuota <nsQuota> -ssQuota <quota in bytes or quota size string>]
    [-setStorageTypeQuota <path> -storageType <storage type> <quota in bytes or quota size string>]
    [-clrQuota <path>]
    [-clrStorageTypeQuota <path>]
    [-safemode enter | leave | get]
    [-nameservice disable | enable <nameservice>]
    [-getDisabledNameservices]
    [-refresh]
    [-refreshRouterArgs <host:ipc_port> <key> [arg1..argn]]
    [-refreshSuperUserGroupsConfiguration]
```

Arguments

-add source nameservices destination	Adds a mount table entry or update if it exists
-update source nameservices destination	Updates a mount table entry attributes
-rm source	Removes mount point of specified path
--ls [-d] path	Lists mount points under specified path. Specify `-d` parameter to get detailed listing
-getDestination path	Gets the subcluster where a file is or should be created
-setQuota path -nsQuota nsQuota -ssQuota ssQuota	Sets quota for specified path
-setStorageTypeQuota path -storageType storageType stQuota	Clears quota of given mount point
-clrQuota path	Sets storage type quota for specified path
-clrStorageTypeQuota path	Clears storage type quota of given mount point
-safemode enter leave get	Manually sets the router entering or leaving the safe mode. The option get will be used for verifying if the Router is in the safe mode state
-nameservice disable enable nameservice	Disables/enables a name service from the federation. If disabled, requests will not go to that name service
-getDisabledNameservices	Gets the name services that are disabled in the federation
-refresh	Updates mount table cache of the connected router
-refreshRouterArgs <host:ipc_port> <key> [arg1..argn]	Triggers a runtime-refresh of the resource specified by <key> on `host:ipc_port`. For example, to enable white list checking, you have to send a refresh command other than restart the router server
-refreshSuperUserGroupsConfiguration	Refreshes superuser proxy groups mappings on the router

diskbalancer

Sergey Ostapov

To Table of Contents

Runs the diskbalancer tool.

The usage is as follows:

```
$ hdfs diskbalancer
     [-plan <datanode> -fs <namenodeURI>]
     [-execute <planfile>]
     [-query <datanode>]
     [-cancel <planfile>]
     [-cancel <planID> -node <datanode>]
     [-report -node <file://> | [<DataNodeID|IP|Hostname>,...]]
     [-report -node -top <topnum>]
```

Arguments

-plan	Creates a disbalancer plan
-execute	Executes a given plan on a DataNode
-query	Gets the current diskbalancer status from a DataNode
-cancel	Cancels a running plan
-report	Reports the volume information from DataNode(s)

ec

Sergey Ostapov

To Table of Contents

Runs the ErasureCoding CLI.

The usage is as follows:

```
$ hdfs ec [generic options]
     [-setPolicy -policy <policyName> -path <path>]
     [-getPolicy -path <path>]
     [-unsetPolicy -path <path>]
     [-listPolicies]
     [-addPolicies -policyFile <file>]
     [-listCodecs]
     [-enablePolicy -policy <policyName>]
     [-disablePolicy -policy <policyName>]
     [-removePolicy -policy <policyName>]
     [-verifyClusterSetup -policy <policyName>...<policyName>]
     [-help [cmd ...]]
```

Arguments

-setPolicy	Sets a specified ErasureCoding policy to a directory
-getPolicy	Gets ErasureCoding policy information about a specified path
-unsetPolicy	Unsets an ErasureCoding policy set by a previous call to `setPolicy` on a directory
-listPolicies	Lists all supported ErasureCoding policies
-addPolicies	Adds a list of erasure coding policies
-listCodecs	Gets the list of supported erasure coding codecs and coders in system
-enablePolicy	Enables an ErasureCoding policy in system
-disablePolicy	Disables an ErasureCoding policy in system
-removePolicy	Removes an ErasureCoding policy from system
-verifyClusterSetup	Verifies if the cluster setup can support a list of erasure coding policies

haadmin

Sergey Ostapov

To Table of Contents

Allows to manage the HA mode for an HDFS cluster.

The usage is as follows:

```
$ hdfs haadmin -transitionToActive <serviceId> [--forceactive]
$ hdfs haadmin -transitionToStandby <serviceId>
$ hdfs haadmin -transitionToObserver <serviceId>
$ hdfs haadmin -failover [--forcefence] [--forceactive] <serviceId> <serviceId>
$ hdfs haadmin -getServiceState <serviceId>
$ hdfs haadmin -getAllServiceState
$ hdfs haadmin -checkHealth <serviceId>
$ hdfs haadmin -help <command>
```

Arguments

-checkHealth	Checks the health of the given NameNode
-failover	Initiates a failover between two NameNodes
-getServiceState	Determines whether the given NameNode is Active or Standby
-getAllServiceState	Returns the state of all the NameNodes
-transitionToActive	Transition the state of the given NameNode to Active (Warning: No fencing is done)
-transitionToStandby	Transition the state of the given NameNode to Standby (Warning: No fencing is done)
-transitionToObserver	Transition the state of the given NameNode to Observer (Warning: No fencing is done)
-help [cmd]	Displays help for the given command or all commands if none is specified

journalnode

Sergey Ostapov

To Table of Contents

Starts a JournalNode for use with HDFS HA with QJM (Quorum Journal Manager).

The usage is as follows:

```
$ hdfs journalnode
```

mover

Sergey Ostapov

To Table of Contents

Runs the data migration utility, and it’s similar to Balancer. This utility periodically scans the files in HDFS to check if the block placement satisfies the storage policy. For the blocks violating the storage policy, it moves the replicas to a different storage type in order to fulfill the storage policy requirement.

This utility always tries to move block replicas within the same node whenever possible. If that is not possible (e.g. when a node doesn’t have the target storage type), then it will copy the block replicas to another node over the network.

NOTE

When both `-p` and `-f` options are omitted, the default path is the root directory.

The usage is as follows:

```
$ hdfs mover [-p <files/dirs> | -f <local file name>]
```

Arguments

-f <local file>	Specifies a local file containing a list of HDFS files/dirs to migrate
-p <files/dirs>	Specifies a space-separated list of HDFS files/dirs to migrate

namenode

Sergey Ostapov

To Table of Contents

Runs the NameNode.

The usage is as follows:

```
$ hdfs namenode [-backup] |
          [-checkpoint] |
          [-format [-clusterid cid ] [-force] [-nonInteractive] ] |
          [-upgrade [-clusterid cid] [-renameReserved<k-v pairs>] ] |
          [-upgradeOnly [-clusterid cid] [-renameReserved<k-v pairs>] ] |
          [-rollback] |
          [-rollingUpgrade <rollback |started> ] |
          [-importCheckpoint] |
          [-initializeSharedEdits] |
          [-bootstrapStandby [-force] [-nonInteractive] [-skipSharedEditsCheck] ] |
          [-recover [-force] ] |
          [-metadataVersion ]
```

Arguments

-backup	Starts the Backup node
-checkpoint	Starts a Checkpoint node
-format [-clusterid cid]	Formats the specified NameNode. It starts the NameNode, formats it and then shuts it down. Will throw `NameNodeFormatException` if the directory already exists and if reformatting is disabled for the cluster
-upgrade [-clusterid cid] [-renameReserved <k-v pairs>]	NameNode should be started with the `upgrade` option after the distribution of a new Arenadata Hyperwave version
-upgradeOnly [-clusterid cid] [-renameReserved <k-v pairs>]	Upgrades the specified NameNode and then shuts it down
-rollback	Rolls back a NameNode to the previous version. This should be used after stopping the cluster and distributing the old Arenadata Hyperwave version
-rollingUpgrade <rollback|started>	When a rolling upgrade is in progress, the `-rollingUpgrade` NameNode startup option is used to specify various rolling upgrade options: ▪ `rollback`: restores the NameNode back to the pre-upgrade release and also reverts the user data back to the pre-upgrade state; ▪ `started`: specifies a rolling upgrade already started so that the NameNode should allow image directories with different layout versions during startup.
-importCheckpoint	Loads an image from a checkpoint directory and saves it into the current one. Checkpoint dir is read from property `dfs.namenode.checkpoint.dir`
-initializeSharedEdits	Formats a new shared edits dir and copies to available edit log segments so that the Standby NameNode can start up
-bootstrapStandby [-force] [-nonInteractive] [-skipSharedEditsCheck]	Allows a Standby NameNode's storage directories to be bootstrapped by copying the latest namespace snapshot from the active NameNode. This is used when first configuring an HA cluster. The option `-force` or `-nonInteractive` has the same meaning as described in the namenode -format command
-recover [-force]	Recovers lost metadata on a corrupt filesystem
-metadataVersion	Verifies that configured directories exist, then prints the metadata versions of the software and the image

nfs3

Sergey Ostapov

To Table of Contents

Starts the NFS3 gateway for use with the HDFS NFS3 Service.

The usage is as follows:

```
$ hdfs nfs3
```

portmap

Sergey Ostapov

To Table of Contents

Starts the RPC portmap for use with the HDFS NFS3 Service.

The usage is as follows:

```
$ hdfs portmap
```

secondarynamenode

Sergey Ostapov

To Table of Contents

Runs an HDFS secondary NameNode.

The usage is as follows:

```
$ hdfs secondarynamenode [-checkpoint [force]] | [-format] | [-geteditsize]
```

Arguments

-checkpoint [force]	Checkpoints the secondary NameNode if EditLog size >= `fs.checkpoint.size`. If `-force` is used, the checkpoint is irrespective of the EditLog size
-format	Formats the local storage during the startup
-geteditsize	Prints the number of uncheckpointed transactions on the NameNode

storagepolicies

Sergey Ostapov

To Table of Contents

Lists/sets/unsets storage policies.

The usage is as follows:

```
$ hdfs storagepolicies
      [-listPolicies]
      [-setStoragePolicy -path <path> -policy <policy>]
      [-getStoragePolicy -path <path>]
      [-unsetStoragePolicy -path <path>]
      [-satisfyStoragePolicy -path <path>]
      [-isSatisfierRunning]
      [-help <command-name>]
```

Arguments

-listPolicies	Lists out all the storage policies
-setStoragePolicy -path <path> -policy <policy>	Sets a storage policy to a file or a directory: ▪ `<path>` — the path referring to either a directory or a file; ▪ `<policy>` — the name of the storage policy.
-getStoragePolicy -path <path>	Gets the storage policy of a file or a directory. `<path>` refers to either a directory or a file
-unsetStoragePolicy -path <path>	Unsets a storage policy to a file or a directory. `<path>` refers to either a directory or a file. After the unset command the storage policy of the nearest ancestor will apply, and if there is no policy on any ancestor, then the default storage policy will apply
-satisfyStoragePolicy -path <path>	Schedules blocks to move based on file's/directory's current storage policy. `<path>` refers to either a directory or a file

zkfc

Sergey Ostapov

To Table of Contents

Starts the ZooKeeper Failover Controller process for use with HDFS HA with QJM (Quorum Journal Manager).

The usage is as follows:

```
$ hdfs zkfc [-formatZK [-force] [-nonInteractive]]
```

Arguments

-formatZK	Formats the ZooKeeper instance: ▪ `-force` — formats the Znode if the Znode exists; ▪ `-nonInteractive` — formats the Znode.
-h	Prints detailed information about the command

computeMeta

Sergey Ostapov

To Table of Contents

Computes HDFS metadata from block files. If a block file is specified, it will compute the checksums from the block file, and save it to the specified output metadata file.

CAUTION

Use this command with caution. If the block file is corrupt and you overwrite its meta file, it will show up as "good" in HDFS, but you can’t read the data. Only use as a last measure, and when you are 100% certain the block file is not corrupted.

The usage is as follows:

```
$ hdfs debug computeMeta -block <block-file> -out <output-metadata-file>
```

Arguments

-block block-file	An absolute path for the block file on the local file system of the data node
-out output-metadata-file	An absolute path for the output metadata file to store the checksum computation result from the block file

recoverLease

Sergey Ostapov

To Table of Contents

Recovers the lease on the specified path. The path must point to the HDFS file system. The default number of retries is `1`.

The usage is as follows:

```
$ hdfs debug recoverLease -path <path> [-retries <num-retries>]
```

Arguments

-path <path>	An HDFS path to recover the lease
[-retries <num-retries>]	Number of times the client will retry calling `recoverLease`. The default number of retries is `1`

verifyMeta

Sergey Ostapov

To Table of Contents

Verifies the HDFS metadata and block files. If a block file is specified, verifies that the checksums in the metadata file match the block file.

The usage is as follows:

```
$ hdfs debug verifyMeta -meta <metadata-file> [-block <block-file>]
```

Arguments

-block block-file	An optional parameter to specify the absolute path for the block file on the local file system of the DataNode
-meta metadata-file	An absolute path for the metadata file on the local file system of the DataNode

FileSystem shell

Sergey Ostapov

Contents

To Table of Contents

Overview
Commands

Overview

The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, WebHDFS, S3 FS, and others. The FS shell is invoked by:

```
$ bin/hadoop fs <args>
```

All FS shell commands take path URIs as arguments. The URI format is *scheme://authority/path*. For HDFS the scheme is `hdfs`, and for the local FS the scheme is `file`. The scheme and authority are optional. If not specified, the default scheme specified in the configuration is used. An HDFS file or directory such as */parent/child* can be specified as *hdfs://namenodehost/parent/child* or simply as */parent/child* (given that your configuration is set to point to *hdfs://namenodehost*).

Most of the commands in FS shell behave like corresponding Unix commands. The difference is described for each command. Error information is sent to stderr and the output is sent to stdout.

If HDFS is being used, `hdfs dfs` is a synonym.

Relative paths can be used. For HDFS, the current working directory is the HDFS home directory */user/<username>* that often has to be created manually. The HDFS home directory can also be implicitly accessed, e.g., when using the HDFS trash folder, the *.Trash* directory in the home directory.

Commands

Command	Description
appendToFile	Appends sources from the local file system to the destination file system
cat	Prints the contents of a file to stdout
checksum	Returns the checksum information of a file
chgrp	Changes group association of files
chmod	Changes the permissions of files
chown	Changes the owner of files
copyFromLocal	Identical to the put command
copyToLocal	Identical to the get command
count	Counts the number of directories and files
cp	Copies files from source to destination
createSnapshot	Creates a snapshot of a snapshot table directory
deleteSnapshot	Deletes a snapshot from a snapshot table directory
df	Displays free space
du	Displays sizes of files and directories
dus	Displays a summary of file lengths
expunge	Permanently deletes files in checkpoints older than the retention threshold from the trash directory, and creates a new checkpoint
find	Finds all files that match the specified expression and applies selected actions to them
get	Copies files to the local file system
getfacl	Displays the ACLs of files and directories
getfattr	Displays the extended attribute names and values (if any) for a file or directory
getmerge	Takes a source directory and a destination file as inputted and concatenates files into the destination local file
head	Displays the first kilobyte of a file to stdout
help	Returns usage output
ls	Returns the file stats or list of its direct children for directory
lsr	Recursive version of ls
mkdir	Takes path URI's as argument and creates directories
moveFromLocal	Identical to the put command
moveToLocal	Moves a file/directory from HDFS to the local file system
mv	Moves files from source to destination
put	Copies sources from the local file system to the destination file system
renameSnapshot	Renames a snapshot
rm	Deletes files specified as args
rmdir	Deletes a directory
rmr	Recursive version of rm
setfacl	Sets ACLs of files and directories
setfattr	Sets an extended attribute name and value for a file or directory
setrep	Changes the replication factor of a file
stat	Prints statistics about the file/directory in the specified format
tail	Displays last kilobyte of the file to stdout
test	Performs a test
text	Takes a source file and outputs the file in a text format
touch	Updates the access and modification times of the file specified by the URI to the current time
touchz	Creates a file of zero-length
truncate	Truncates all files that match the specified file pattern to the specified length
usage	Prints help for an individual command

appendToFile

Sergey Ostapov

To Table of Contents

Appends a single source, or multiple sources from a local file system to the destination file system.
Also reads input from stdin and appends to the destination file system.

The destination file gets created if it doesn't already exist.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -appendToFile <localsrc> ... <dst>
```

Example:

```
$ hadoop fs -appendToFile localfile /user/hadoop/hadoopfile
$ hadoop fs -appendToFile localfile1 localfile2 /user/hadoop/hadoopfile
$ hadoop fs -appendToFile localfile hdfs://nn.example.com/hadoop/hadoopfile
$ hadoop fs -appendToFile - hdfs://nn.example.com/hadoop/hadoopfile
```

cat

Sergey Ostapov

To Table of Contents

Prints an HDFS file contents to stdout.

The usage is as follows:

```
$ hadoop fs -cat <path/to/file>
```

Arguments

-ignoreCrc	Disables checksum verification

Examples:

```
$ hadoop fs -cat /user/admin/testload/file_test.csv
$ hadoop fs -checksum file:///etc/hosts
```

checksum

Sergey Ostapov

To Table of Contents

Returns the checksum information of a file.

The usage is as follows:

```
$ hadoop fs -checksum [-v] URI
```

Arguments

-v	Displays blocks size for the file

Example:

```
$ hadoop fs -checksum hdfs://nn1.example.com/file1
$ hadoop fs -checksum file:///etc/hosts
```

chgrp

Sergey Ostapov

To Table of Contents

Changes group association of files.
The user must be the owner of files, or a superuser.

The usage is as follows:

```
$ hadoop fs -chgrp [-R] GROUP URI [URI ...]
```

Arguments

-R	Recursively through the directory structure

chmod

Sergey Ostapov

To Table of Contents

Changes the permissions of files.
The user must be the owner of the file, or a superuser.

The usage is as follows:

```
$ hadoop fs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI ...]
```

Arguments

-R	Recursively through the directory structure

chown

Sergey Ostapov

To Table of Contents

Changes the owner of files.
The user must be a superuser.

The usage is as follows:

```
$ hadoop fs -chown [-R] [OWNER][:[GROUP]] URI [URI ]
```

Arguments

-R	Recursively through the directory structure

copyFromLocal

Sergey Ostapov

To Table of Contents

Works similarly to the put command, except that the source is restricted to a local file reference.

Arguments

-d	Skips creation of a temporary file with the suffix *.COPYING*.
-f	Overwrites the destination file if it already exists
-l	Allows DataNode to lazily persist files to disk, forces the replication factor of 1. This flag results in reduced durability. Use this flag with care
-p	Preserves access and modification times, ownership and the permissions (assuming the permissions can be propagated across filesystems)

copyToLocal

Sergey Ostapov

To Table of Contents

Works similarly to the get command, except that the destination is restricted to a local file reference.

The usage is as follows:

```
$ hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
```

count

Sergey Ostapov

To Table of Contents

Counts the number of directories, files and bytes under the paths that match the specified file pattern. Gets the quota and the usage.

The output columns with `-count` are: DIR_COUNT, FILE_COUNT, CONTENT_SIZE, PATHNAME.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -count [-q] [-h] [-v] [-x] [-t [<storage type>]] [-u] [-e] <paths>
```

Arguments

-u	Controls what columns the output contains: QUOTA, REMAINING_QUOTA, SPACE_QUOTA, REMAINING_SPACE_QUOTA, PATHNAME
-q	Controls what columns the output contains: QUOTA, REMAINING_QUOTA, SPACE_QUOTA, REMAINING_SPACE_QUOTA, DIR_COUNT, FILE_COUNT, CONTENT_SIZE, PATHNAME
-t	The list of possible parameters that can be used: `all`, `ram_disk`, `ssd`, `disk`, `archive`
-h	Shows sizes in human-readable format
-v	Displays a header line
-x	Excludes snapshots from the result calculation. Without the `-x` option (default), the result is always calculated from all INodes, including all snapshots under the given path. The `-x` option is ignored if `-u` or `-q` option is given
-e	Shows the erasure coding policy for each file
-e	Controls what columns the output contains: DIR_COUNT, FILE_COUNT, CONTENT_SIZE, ERASURECODING_POLICY, PATHNAME
-ERASURECODING_POLICY	Name of the policy for the file. If an erasure coding policy is set on that file, it will return name of the policy. If no erasure coding policy is set, it will return `Replicated` which means it uses replication storage strategy

Examples:

```
$ hadoop fs -count hdfs://nn1.example.com/file1 hdfs://nn2.example.com/file2
$ hadoop fs -count -q hdfs://nn1.example.com/file1
$ hadoop fs -count -q -h hdfs://nn1.example.com/file1
$ hadoop fs -count -q -h -v hdfs://nn1.example.com/file1
$ hadoop fs -count -u hdfs://nn1.example.com/file1
$ hadoop fs -count -u -h hdfs://nn1.example.com/file1
$ hadoop fs -count -u -h -v hdfs://nn1.example.com/file1
$ hadoop fs -count -e hdfs://nn1.example.com/file1
```

cp

Sergey Ostapov

To Table of Contents

Copies files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.

The raw.* namespace extended attributes are preserved if the source and destination filesystems support them (HDFS only), and all source and destination pathnames are in the */.reserved/raw* hierarchy.

Determination of whether the raw.* namespace `xattrs` are preserved is independent of the `-p` (preserve) flag.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>
```

Arguments

-f	Overwrites the destination if it already exists
-p	Preserves file attributes [topx] (timestamps, ownership, permission, ACL, XAttr). If `-p` is specified with no arg, then preserves timestamps, ownership, permission. If `-pa` is specified, then preserves permission also because ACL is a super-set of permission. Determination of whether raw namespace extended attributes are preserved is independent of the `-p` flag

Examples:

```
$ hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2
$ hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir
```

createSnapshot

Sergey Ostapov

To Table of Contents

Creates a snapshot of a snapshottable directory.
This operation requires owner privilege of the snapshottable directory. For more information on snapshots, see the Snapshots in HDFS article.

The usage is as follows:

```
$ hdfs dfs -createSnapshot <path> [<snapshotName>]
```

Arguments

<path>	The path of the snapshottable directory
[<snapshotName>]	The snapshot name, which is an optional argument. When it is omitted, a default name is generated using a timestamp with the format `'s’yyyyMMdd-HHmmss.SSS`, e.g. `s20130412-151029.033`

deleteSnapshot

Sergey Ostapov

To Table of Contents

Deletes snapshot from snapshottable directory. This operation requires owner privilege of the snapshottable directory. For more information on snapshots, see the Snapshots in HDFS article.

The usage is as follows:

```
$ hdfs dfs -deleteSnapshot <path> <snapshotName>
```

Arguments

<path>	Path to the snapshottable directory
<snapshotName>	Snapshot name

df

Sergey Ostapov

Displays free space.

The usage is as follows:

```
$ hadoop fs -df [-h] URI [URI ...]
```

Arguments

-h	Formats file sizes in human-readable format (e.g `64.0m` instead of `67108864`)

Example:

```
$ hadoop fs -df /user/hadoop/dir1
```

du

Sergey Ostapov

To Table of Contents

Displays the size of files and directories contained in the given directory or the length of a file in case it’s just a file.

The `du` returns three columns with the following format:

```
size disk_space_consumed_with_all_replicas full_path_name
```

Returns `0` on success and `-1` on error

The usage is as follows:

```
$ hadoop fs -du [-s] [-h] [-v] [-x] URI [URI ...]
```

Arguments

-s	Shows the result in the aggregate summary of file lengths rather than the individual files. Without the `-s` option, the calculation is done by going 1 level deep from the given path
-h	Formats files size in the human-readable format (e.g `64.0m` instead of `67108864`)
-v	Displays the names of columns as the header line
-x	Excludes snapshots from the result calculation. Without the `-x` option (default), the result is always calculated from all INodes, including all snapshots under the given path

Example:

```
$ hadoop fs -du /user/hadoop/dir1 /user/hadoop/file1 hdfs://nn.example.com/user/hadoop/dir1
```

dus

Sergey Ostapov

To Table of Contents

Displays a summary of file lengths.

The usage is as follows:

```
$ hadoop fs -dus <args>
```

NOTE

This command is deprecated. Instead, use `hadoop fs -du -s`.

expunge

Sergey Ostapov

To Table of Contents

Permanently deletes files in checkpoints older than the retention threshold from trash directory, and creates new checkpoint.

When checkpoint is created, recently deleted files in trash are moved under the checkpoint. Files in checkpoints older than `fs.trash.interval` will be permanently deleted on the next invocation of `-expunge` command.

If the file system supports this feature, users can configure creation and deletion of checkpoints periodically by the parameter stored as `fs.trash.checkpoint.interval` (in *core-site.xml*). This value should be smaller or equal to `fs.trash.interval`.

If the `-immediate` option is passed, all files in the trash for the current user are immediately deleted, ignoring the `fs.trash.interval setting`.

If the `-fs` option is passed, the supplied filesystem will be expunged, rather than the default filesystem and checkpoint is created.

The usage is as follows:

```
$ hadoop fs -expunge [-immediate] [-fs <path>]
```

Example:

```
$ hadoop fs -expunge --immediate -fs s3a://landsat-pds/
```

find

Sergey Ostapov

To Table of Contents

Finds all files that match the specified expression and applies selected actions to them. If no path is specified, then defaults to the current working directory. If no expression is specified, then defaults to `-print`.

The following primary expressions are recognized:

- -name pattern;
- -iname pattern.

Evaluates as `true` if the base name of the file matches the pattern using standard file system globbing. If `-iname` is used then the match is case-insensitive.

`-print` and `-print0` always return `true` and are used to print the current path to stdout. If the `-print0` expression is used, then an ASCII NULL character is appended.

The following operators are recognized:

- expression -a expression;
- expression -and expression;
- expression expression.

The logical operator `AND` is used for joining two expressions.

Returns `true` if both child expressions return `true`. Implied by the juxtaposition of two expressions and so doesn't need to be explicitly specified. The second expression won't be applied if the first fails. Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -find <path> ... <expression> ...
```

Example:

```
$ hadoop fs -find / -name test -print
```

get

Sergey Ostapov

To Table of Contents

Copies files to the local file system. Files that fail the CRC check may be copied with the `-ignorecrc` option. Files and CRCs may be copied using the `-crc` option.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -get [-ignorecrc] [-crc] [-p] [-f] <src> <localdst>
```

Arguments

-p	Preserves access and modification times, ownership and the sessions, assuming the permissions can be propagated across filesystems
-f	Overwrites the destination if it already exists
-ignorecrc	Skips CRC checks on the file(s) downloaded
-crc	Writes CRC checksums for the downloaded files

Examples:

```
$ hadoop fs -get /user/hadoop/file localfile
$ hadoop fs -get hdfs://nn.example.com/user/hadoop/file localfile
```

getfacl

Sergey Ostapov

To Table of Contents

Displays the Access Control Lists (ACLs) of files and directories.

If a directory has a default ACL, then `getfacl` also displays the default ACL.

Returns `0` on success and non-zero on error.

The usage is as follows:

```
$ hadoop fs -getfacl [-R] <path>
```

Arguments

-R	Lists the ACLs of all files and directories recursively
<path>	File or directory to list

Example:

```
$ hadoop fs -getfacl /file
$ hadoop fs -getfacl -R /dir
```

getfattr

Sergey Ostapov

To Table of Contents

Displays the extended attribute names and values (if any) for a file or directory. Returns `0` on success and non-zero on error.

The usage is as follows:

```
$ hadoop fs -getfattr [-R] -n name | -d [-e en] <path>
```

Examples:

```
$ hadoop fs -getfattr -d /file
$ hadoop fs -getfattr -R -n user.myAttr /dir
```

Arguments

-R	Recursively lists the attributes for all files and directories
-n	Dumps the named extended attribute value
-d	Dumps all extended attribute values associated with pathname
-e encoding	Encodes values after retrieving them. Valid encodings are `text`, `hex`, and `base64`. Values encoded as text strings are enclosed in double quotes (`"`), and values encoded as hexadecimal and base64 are prefixed with `0x` and `0s`, respectively
path	A path to file or directory

getmerge

Sergey Ostapov

To Table of Contents

Takes a source directory and a destination file as inputted and concatenates files into the destination local file.
Optionally `-nl` can be set to enable adding a newline character (LF) at the end of each file; `-skip-empty-file` can be used to avoid unwanted newline characters in case of empty files.

Returns `0` on success and non-zero on error.

The usage is as follows:

```
$ hadoop fs -getmerge [-nl] <src> <localdst>
```

Example:

```
$ hadoop fs -getmerge -nl /src /opt/output.txt
$ hadoop fs -getmerge -nl /src/file1.txt /src/file2.txt /output.txt
```

head

Sergey Ostapov

To Table of Contents

Displays the first kilobyte of the file to stdout.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -head URI
```

Example:

```
$ hadoop fs -head pathname
```

help

Sergey Ostapov

To Table of Contents

Returns the usage output.

The usage is as follows:

```
$ hadoop fs -help
```

Arguments

-help	Prints help

ls

Sergey Ostapov

To Table of Contents

For a file, `ls` returns stat on the file with the following format:

```
permissions number_of_replicas userid groupid filesize modification_date modification_time filename
```

For a directory, `ls` returns the list of its direct children as in Unix.
A directory is listed as:

```
permissions userid groupid modification_date modification_time dirname
```

Files within a directory are ordered by name by default.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] <args>
```

Arguments

-C	Displays the paths to files and directories only
-d	Directories are listed as plain files
-h	Formats file sizes to the human-readable format (e.g. `64.0m` instead of `67108864`)
-q	Prints `?` instead of non-printable characters
-R	Recursively lists subdirectories encountered
-t	Sorts the output by modification time (most recent first)
-S	Sorts output by file size
-r	Reverses the sort order
-u	Uses access time rather than modification time for display and sorting
-e	Displays the erasure coding policy of files and directories only

Example:

```
$ hadoop fs -ls /user/hadoop/file1
$ hadoop fs -ls -e /ecdir
```

lsr

Sergey Ostapov

To Table of Contents

The recursive version of the ls command.

The usage is as follows:

```
$ hadoop fs -lsr <args>
```

NOTE

This command is deprecated. Instead, use `hadoop fs -ls -R`.

mkdir

Sergey Ostapov

To Table of Contents

Creates directories at given path URIs.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -put [-f] [-p] [-l] [-d] [-t <thread count>] [ - | <localsrc1> .. ]. <dst>
```

Arguments

-p	Similar to the Unix `mkdir -p`, creating parent directories along the path

Example:

```
$ hadoop fs -mkdir /user/hadoop/dir1 /user/hadoop/dir2
$ hadoop fs -mkdir hdfs://nn1.example.com/user/hadoop/dir hdfs://nn2.example.com/user/hadoop/dir
```

moveFromLocal

Sergey Ostapov

To Table of Contents

Similar to the put command, except that the source `localsrc` is deleted after it’s copied.

The usage is as follows:

```
$ hadoop fs -moveFromLocal <localsrc> <dst>
```

moveToLocal

Sergey Ostapov

To Table of Contents

NOTE

This command is not implemented yet.

Moves a file or directory from the Hadoop file system to a destination in the local file system.

mv

Sergey Ostapov

To Table of Contents

Moves files from source to destination on the HDFS file system. This command allows multiple sources and in this case, the destination must be a directory. Moving files between different file systems is prohibited.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -mv URI [URI ...] <dest>
```

Example:

```
$ hadoop fs -mv /user/hadoop/file1 /user/hadoop/file2
$ hadoop fs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2 hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1
```

put

Sergey Ostapov

To Table of Contents

Copies a single file or multiple files from the local file system to the destination file system. It also reads input from stdin and writes to the destination file system if the source is set to `-`. Copying fails if the file already exists, unless the `-f` flag is given.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -put [-f] [-p] [-l] [-d] [-t <thread count>] [ - | <localsrc1> .. ]. <dst>
```

Arguments

-p	Preserves access and modification time, ownership, and the permissions (assuming the permissions can be propagated across filesystems)
-f	Overwrites the destination if it already exists
-l	Allows DataNode to lazily persist a file to disk; forces the replication factor of `1`. This flag results in reduced durability. Be cautious when using this argument
-d	Skips the creation of the temporary file with the suffix `.COPYING`

Example:

```
$ hadoop fs -put localfile /user/hadoop/hadoopfile
$ hadoop fs -put -f localfile1 localfile2 /user/hadoop/hadoopdir
$ hadoop fs -put -d localfile hdfs://nn.example.com/hadoop/hadoopfile
$ hadoop fs -put - hdfs://nn.example.com/hadoop/hadoopfile
```

renameSnapshot

Sergey Ostapov

To Table of Contents

Renames a snapshot.

This operation requires owner privilege of the snapshottable directory. For more information on snapshots, see the HDFS Snapshots article.

The usage is as follows:

```
$ hdfs dfs -renameSnapshot <path> <oldName> <newName>
```

Arguments

<path>	The path to the snapshottable directory
<oldName>	The old snapshot name
<newName>	The new snapshot name

rm

Sergey Ostapov

To Table of Contents

Deletes files specified as args.

If trash is enabled, moves the deleted file to the *.Trash* directory (given by getTrashRoot).

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -rm [-f] [-r |-R] [-skipTrash] [-safely] URI [URI ...]
```

Arguments

-f	Doesn’t display a diagnostic message or modify the exit status to reflect an error if the file doesn’t exist
-R	Deletes the directory and any content under it recursively
-r	Equivalent to the `-R` argument
-skipTrash	Bypasses the trash function, if enabled, and deletes the specified file(s) immediately. This can be useful when it is necessary to delete files from an over-quota directory
-safely	Requires safety confirmation before deleting directory with total number of files greater than `hadoop.shell.delete.limit.num.files` (in *core-site.xml*, default: `100`). It can be used with `-skipTrash` to prevent accidental deletion of large directories. A delay is expected when walking over large directory recursively to count the number of files to be deleted before the confirmation

Example:

```
$ hadoop fs -rm hdfs://nn.example.com/file /user/hadoop/emptydir
```

rmdir

Sergey Ostapov

To Table of Contents

Deletes a directory.

The usage is as follows:

```
$ hadoop fs -rmdir [--ignore-fail-on-non-empty] URI [URI ...]
```

Arguments

--ignore-fail-on-non-empty	When using wildcards, doesn’t fail if a directory still contains files

Example:

```
$ hadoop fs -rmdir /user/hadoop/emptydir
```

rmr

Sergey Ostapov

To Table of Contents

The recursive version of the rm command.

The usage is as follows:

```
$ hadoop fs -rmr [-skipTrash] URI [URI ...]
```

NOTE

This command is deprecated. Instead, use `hadoop fs -rm -r`.

setfacl

Sergey Ostapov

To Table of Contents

Sets Access Control Lists (ACLs) of files and directories. Returns `0` on success and non-zero on error.

The usage is as follows:

```
$ hadoop fs -setfacl [-R] [-b
|-k -m |-x <acl_spec> <path>] |[--set <acl_spec> <path>]
```

Arguments

-b	Removes everything, but the base ACL entries. The entries for user, group, and others are retained for compatibility with permission bits
-k	Removes the default ACL
-R	Applies operations to all files and directories recursively
-m	Modifies ACL. New entries are added to the ACL, and existing entries are retained
-x	Removes specified ACL entries. Other ACL entries are retained
--set	Fully replaces the ACL, discarding all existing entries. The `acl_spec` must include entries for user, group, and others for compatibility with permission bits. If the `acl_spec` contains only access entries, then the existing default entries are retained. If the `acl_spec` contains only default entries, then the existing access entries are retained. If the `acl_spec` contains both access and default entries, then both are replaced
acl_spec	A comma-separated list of ACL entries
path	A path to the file or directory to modify

Example:

```
$ hadoop fs -setfacl -m user:hadoop:rw- /file
$ hadoop fs -setfacl -x user:hadoop /file
$ hadoop fs -setfacl -b /file
$ hadoop fs -setfacl -k /dir
$ hadoop fs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /file
$ hadoop fs -setfacl -R -m user:hadoop:r-x /dir
$ hadoop fs -setfacl -m default:user:hadoop:r-x /di
```

setfattr

Sergey Ostapov

To Table of Contents

Sets an extended attribute name and value for a file or directory.

Returns `0` on success and non-zero on error.

The usage is as follows:

```
$ hadoop fs -setfattr -n name [-v value] | -x name <path>
```

Arguments

-n	The extended attribute name
-v	The extended attribute value. There are three different encoding methods for the value. If the argument is enclosed in double quotes, then the value is the string inside the quotes. If the argument is prefixed with `0x` or `0X`, then it’s taken as a hexadecimal number. If the argument begins with `0s` or `0S`, then it’s taken as a base64 encoding
-x	Removes the extended attribute
<path>	A path to a file or a directory

Example:

```
$ hadoop fs -setfattr -n user.myAttr -v myValue /file
$ hadoop fs -setfattr -n user.noValue /file
$ hadoop fs -setfattr -x user.myAttr /file
```

setrep

Sergey Ostapov

To Table of Contents

Changes the replication factor of a file.

If the specified path points to a directory, the command recursively changes the replication factor of all files under the specified directory. The EC files will be ignored when executing this command.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -setrep [-R] [-w] <numReplicas> <path>
```

Arguments

-w	Requests the command to wait for the replication to complete. This can potentially take a very long time
-R	Accepted for the backward compatibility. It has no effect

Example:

```
$ hadoop fs -setrep -w 3 /user/hadoop/dir1
```

stat

Sergey Ostapov

To Table of Contents

Prints statistics about the file/directory at `<path>` in the specified format.

Format accepts permissions in octal (`%a`) and symbolic (`%A`), filesize in bytes (`%b`), type (`%F`), group name of owner (`%g`), name (`%n`), block size (`%o`), replication (`%r`), username of owner(`%u`), access date(`%x` , `%X`), and modification date (`%y` , `%Y`).

The `%x` and `%y` show UTC dates as `yyyy-MM-dd HH:mm:ss` , and `%X` and `%Y` show milliseconds since January 1, 1970 UTC.

If the format isn’t specified, `%y` is used by default.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -stat [format] <path> ...
```

Example:

```
$ hadoop fs -stat "type:%F perm:%a %u:%g size:%b mtime:%y atime:%x name:%n" /file
```

tail

Sergey Ostapov

To Table of Contents

Displays the last kilobyte of the file to stdout.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -tail [-f] URI
```

Arguments

-f	Outputs appended data as the file grows, as in Unix

Example:

```
$ hadoop fs -tail pathname
```

test

Sergey Ostapov

To Table of Contents

Performs a file test.

The usage is as follows:

```
$ hadoop fs -test -[defswrz] URI
```

Arguments

-d	If the path is a directory, returns `0`
-e	If the path exists, returns `0`
-f	If the path is a file, returns `0`
-s	If the path is not empty, returns `0`
-s	If the path exists and write permission is granted, returns `0`
-r	If the path exists and read permission is granted, returns `0`
-z	If the file is zero length, returns `0`

Example:

```
$ hadoop fs -test -e filename
```

text

Sergey Ostapov

To Table of Contents

Takes a source file and outputs the file in a text format. The allowed formats are ZIP and TextRecordInputStream.

The usage is as follows:

```
$ hadoop fs -text <src>
```

touch

Sergey Ostapov

To Table of Contents

Updates the access and modification times of the file specified by the URI to the current time.

If the file doesn't exist, then a zero-length file is created at URI with current time as the timestamp of that URI.

The timestamp format is as follows: `yyyyMMddHHmmss`. Where:

- `yyyy` — four digit year (e.g. `2018`);
- `MM` — two digit month (e.g. `08` for August);
- `dd` — two digit day (e.g. `25`);
- `HH` — two digit hour (e.g. `12`);
- `mm` — two digit minute (e.g. `30`);
- `ss` — two digit second (e.g. `00`).

Thus, the value `20180825:123000` represents August 25th 2018, 12:30:00 pm.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -touch [-a] [-m] [-t TIMESTAMP] [-c] URI [URI ...]
```

Arguments

-a	Changes only the access time
-m	Changes only the modification time
-t	Specifies a timestamp (in format `yyyyMMdd:HHmmss`) instead of the current time
-c	Doesn't create file if it doesn't exist

Example:

```
$ hadoop fs -touch -m -t 20180809:230000 pathname
$ hadoop fs -touch -t 20180809:230000 pathname
$ hadoop fs -touch -a pathname
```

touchz

Sergey Ostapov

Creates a file of zero length.
Returns an error if the file exists with non-zero length.

Returns `0` on success and `-1` on error.

The usage is as follows:

```
$ hadoop fs -touchz URI [URI ...]
```

Example:

```
$ hadoop fs -touchz pathname
```

truncate

Sergey Ostapov

To Table of Contents

Truncates all files that match the specified file pattern to the specified length.

The usage is as follows:

```
$ hadoop fs -truncate [-w] <length> <paths>
```

Arguments

-w	Waits for the block recovery to complete. Without the `-w` flag the file may remain unclosed for some time, while the recovery is in progress. During this time, the file can’t be reopened for append

Example:

```
$ hadoop fs -truncate 55 /user/hadoop/file1 /user/hadoop/file2
$ hadoop fs -truncate -w 127 hdfs://nn1.example.com/user/hadoop/file1
```

usage

Sergey Ostapov

To Table of Contents

Prints help for an individual command.

The usage is as follows:

```
$ hadoop fs -usage command
```

HDFS command cheatsheet

Sergey Ostapov

Contents

To Table of Contents

List commands
Read and write commands
Download and upload commands
File management commands
Ownership and validation commands
Filesystem commands
Administration commands

List commands

hdfs dfs -ls /	Lists all the files/directories in HDFS destination path
hdfs dfs -ls -d /tmp	Lists directories as plain files. In this case, this command lists the details of the *tmp* directory
hdfs dfs -ls -h /hdfsDirectory	Formats file sizes in a human-readable fashion in an HDFS directory
hdfs dfs -ls -R /hdfsDirectory	Recursively lists all files and all subdirectories in an HDFS directory
*hdfs dfs -ls /hdfsDirectory/hdfsFile**	Lists all the files matching the pattern. In this case, lists all the files inside the *hdfsDirectory* HDFS directory that start with *hdfsFile*

Read and write commands

hdfs dfs -text /hdfsDirectory/hdfsFile	Takes a source file and outputs the file in text format to stdout. The allowed formats are ZIP and TextRecordInputStream
hdfs dfs -cat /hdfsDirectory/hdfsFile	Displays the content of an HDFS file to stdout
hdfs dfs -appendToFile localfsFile /tmp/hdfsFile	Appends the content from a local file to an HDFS file

Download and upload commands

hdfs dfs -put localfsFile /hdfsDirectory	Copies a file from the local file system to an HDFS directory
hdfs dfs -put -f localfsFile /hdfsDirectory	Copies a file from the local file system to an HDFS directory, and in case the HDFS file already exists, overwrites the file using the `-f` option
hdfs dfs -put -l localfsFile /hdfsDirectory	Copies a file from the local file system to an HDFS directory. Allows a DataNode to save the file to disk lazily
hdfs dfs -put -p localfsFile /hdfsDirectory	Copies a file from the local file system to an HDFS directory. Passing `-p` preserves access and modification times, ownership, and the mode
hdfs dfs -get hdfsFile /localfsDirectory	Copies a file from an HDFS directory to the local file system directory
hdfs dfs -get -p hdfsFile /localfsDirectory	Copies a file from an HDFS directory to the local file system directory. Passing `-p` preserves access and modification times, ownership, and the mode
hdfs dfs -get /localfsDirectory/.*/hdfsDirectory*	Copies all files matching the pattern from the local file system directory to an HDFS directory
hdfs dfs -copyFromLocal localfsFile /hdfsDirectory	Works similarly to the put command, except that the source is restricted to a local file reference
hdfs dfs -copyToLocal hdfsFile /localfsDirectory	Works similarly to the put command, except that the destination is restricted to a local file reference
hdfs dfs -moveFromLocal localfsFile /hdfsDirectory	Works similarly to the put command, except that the source is deleted after it's copied

File management commands

hdfs dfs -cp /hdfsDirectory1/hdfsFile1 /hdfsDirectory2	Copies a file from an HDFS directory to another HDFS directory
hdfs dfs -cp -p /hdfsDirectory1/hdfsFile1 /hdfsDirectory2	Copies a file from an HDFS directory to another HDFS directory. Passing `-p` preserves access and modification times, ownership, and the mode
hdfs dfs -cp -f /hdfsDirectory1/hdfsFile1 /hdfsDirectory2	Copies a file from an HDFS directory to another HDFS directory. Passing `-f` overwrites the destination if it already exists
hdfs dfs -mv /hdfsDirectory1/hdfsFile1 /hdfsDirectory2	Moves a file from an HDFS directory to another HDFS directory
hdfs dfs -rm /hdfsDirectory/hdfsFile *hdfs dfs -rm -r /hdfsDirectory* *hdfs dfs -rm -R /hdfsDirectory* *hdfs dfs -rmr /hdfsDirectory*	Deletes all subdirectories and files recursively from an HDFS directory
hdfs dfs -rm -skipTrash /hdfsDirectory	The `-skipTrash` option bypasses the trash, if enabled, and deletes the specified file(s) in an HDFS directory immediately
hdfs dfs -rm -f hdfsFile	If the file does not exist, a warning is not displayed when executing the command
hdfs dfs -rmdir /hdfsDirectory	Deletes an HDFS directory
hdfs dfs -mkdir /hdfsDirectory	Creates a directory in the specified HDFS location
hdfs dfs -mkdir -f /hdfsDirectory1/hdfsDirectory2	Creates a directory in the specified HDFS location. This command does not fail even if the directory already exists
hdfs dfs -touchz /hdfsDirectory/hdfsFile	Creates a file of zero length at the specified path with the current time as the timestamp of that path

Ownership and validation commands

hdfs dfs -checksum /hdfsDirectory/hdfsFile	Dumps the checksum information for a file to stdout
hdfs dfs -chmod 755 /hdfsDirectory/hdfsFile	Changes the permissions of the file
hdfs dfs -chmod -R 755 /hdfsDirectory	Changes the permissions of the files recursively
hdfs dfs -chown owner:group /hdfsDirectory	Changes the owner of the file
hdfs dfs -chown -R owner:group /hdfsDirectory	Changes the owner of the files recursively
hdfs dfs -chgrp group /hdfsDirectory	Changes the group association of the file
hdfs dfs -chgrp -R group /hdfsDirectory	Changes the group association of the files recursively

Filesystem commands

hdfs dfs -df /	Shows the capacity, free, and used space
hdfs dfs -df -h /	Shows the capacity, free, and used space of the filesystem. The key `-h` formats the sizes of files in a human-readable format
hdfs dfs -du /hdfsDirectory/hdfsFile	Shows the occupied space (in bytes) used by the files that match the specified file pattern
hdfs dfs -du -s /hdfsDirectory/hdfsFile	Rather than showing the size of each file that matches the pattern, shows the total (summary) size
hdfs dfs -du -h /hdfsDirectory/hdfsFile	Shows the occupied space (in bytes) used by the files that match the specified file pattern. Formats the sizes of files in a human-readable format

Administration commands

hdfs balancer -threshold 30	Launches the cluster balancing utility. The default threshold value is being overwritten
hadoop version	Prints the current Hadoop version
hdfs fsck /	Checks the health of HDFS
hdfs dfsadmin -safemode leave	Turns off the safe mode for a NameNode
hdfs dfsadmin -refreshNodes	Re-reads the *hosts* and *exclude* files to update the set of DataNodes that are allowed to connect to the NameNode and those that should be decommissioned or recommissioned
hdfs namenode -format	Formats the NameNode

HBase shell commands

Daria Barysheva

Contents

To Table of Contents

General commands
Data definition commands (DDL)
Namespace commands
Data manipulation commands (DML)
Tools commands
Replication commands
Snapshots commands
Configuration commands
Quotas commands
Security commands
Procedures commands
Visibility labels commands
Rsgroup commands

This topic describes the main commands available while working with HBase shell.

General commands

These commands are useful for general purposes, such as displaying the current cluster and user information, getting help, etc.

Command	Description
status	Shows a status of the HBase cluster
table_help	Displays help on such data operations, as scan, put, get, disable, drop, etc.
whoami	Displays the current user details
version	Displays the HBase version
processlist	Displays tasks of Region Servers

Data definition commands (DDL)

These commands are used to create table schemas and work with them.

Command	Description
create	Creates a new table
describe	Displays a table schema
list	Displays tables matching the given regex
exists	Checks the table existence
locate_region	Locates a Region by the given table name and the row key
list_regions	Returns all Regions for the particular table
alter	Allows users to add/modify/delete column families and to change the table configuration
alter_async	Performs the same as alter command, but does not wait for all Regions to receive the schema changes
alter_status	Gets the status of the alter command
disable	Disables a table
disable_all	Disables tables matching the given regex
is_disabled	Checks whether the table is disabled or not
enable	Enables the table which might be currently disabled
enable_all	Enables tables matching the given regex
is_enabled	Checks whether the table is enabled or not
get_table	Returns the given table as an object to be manipulated by the user
drop	Deletes a table permanently
drop_all	Deletes all tables matching the given regex
show_filters	Shows all filters in HBase

Namespace commands

These commands allow users to work with namespaces — logical groups of tables.

Command	Description
create_namespace	Creates a new namespace
describe_namespace	Returns the namespace description
list_namespace	Returns a list of namespaces — all or matching the given regex
list_namespace_tables	Returns tables included into the namespace
alter_namespace	Changes a namespace
drop_namespace	Drops a namespace

Data manipulation commands (DML)

These commands are useful for working with data stored in HBase tables: reading, writing, deleting, etc.

Command	Description
put	Inserts a cell value at the specified table/row/column and optionally timestamp coordinates
append	Appends a cell value at the specified table/row/column coordinates
incr	Increments a cell value at the specified table/row/column coordinates
get_counter	Returns a counter cell value at the specified table/row/column coordinates
get	Fetches the row/cell contents
scan	Scans the table and returns all the rows matching the given specifications
count	Returns a number of rows in the table
delete	Deletes a cell value at the specified table/row/column and optionally timestamp coordinates
deleteall	Deletes all cells in the specified row
truncate	Disables, drops, and recreates the specified table
truncate_preserve	Disables, drops, and recreates the specified table, maintaining the previous Region boundaries
get_splits	Returns split points for the given table

Tools commands

These commands are intended for experts only. They help to perform operations with Regions, such as: split, merge, compaction, etc. Use them carefully.

Command	Description
assign	Assigns a Region
unassign	Unassigns a Region
close_region	Deprecated. Use unassign instead
compact	Runs compaction for the defined table/Region/column family
compact_rs	Compacts all Regions within the defined Region Server
major_compact	Runs Major compaction for the defined table/Region/column family
compaction_state	Gets the table compaction status
splitormerge_enabled	Checks whether split or merge operations are enabled
splitormerge_switch	Enables/disables split or merge operations
merge_region	Merges two Regions
split	Splits the entire table or the specified Region
move	Moves a Region
flush	Flushes all Regions for the table/Region Server or all Regions with the specified name
wal_roll	Rolls the WAL writer
list_deadservers	Lists all dead Region Servers
clear_deadservers	Clears all/specified dead Region Servers
clear_block_cache	Clears blocks of the specified table from BlockCache
clear_compaction_queues	Clears compaction queues at the Region Server
balance_switch	Enables/disables the cluster Balancer
balancer_enabled	Checks the Balancer state
balancer	Triggers the cluster Balancer
catalogjanitor_switch	Enables/disables the CatalogJanitor
catalogjanitor_enabled	Checks the CatalogJanitor state
catalogjanitor_run	Runs the CatalogJanitor
cleaner_chore_switch	Enables/disables the Cleaner chore
cleaner_chore_enabled	Checks the Cleaner chore state
cleaner_chore_run	Runs the Cleaner chore
normalizer_switch	Enables/disables the Region Normalizer
normalizer_enabled	Checks the Region Normalizer state
normalize	Runs the Region Normalizer
is_in_maintenance_mode	Checks whether the Master is in maintenance mode
zk_dump	Shows the dump status of the HBase cluster, as seen by ZooKeeper
trace	Starts/stops/checks tracing via HTrace

Replication commands

These commands are used for configuring replication.

Command	Description
add_peer	Adds a replication relationship (peer)
append_peer_namespaces	Appends replicable namespaces to the replication peer
append_peer_tableCFs	Appends replicable tables/column families to the replication peer
set_peer_namespaces	Sets all replicable namespaces for the replication peer
set_peer_tableCFs	Sets all replicable tables/column families for the replication peer
set_peer_bandwidth	Sets the source bandwidth for the replication peer (per Region Server)
set_peer_replicate_all	Sets the `REPLICATE_ALL` flag for the replication peer
set_peer_exclude_namespaces	Sets not replicable namespaces for the replication peer
set_peer_exclude_tableCFs	Sets not replicable tables/column families for the replication peer
disable_table_replication	Disables replication for all column families of the table
enable_table_replication	Enables replication for all column families of the table
disable_peer	Disables a replication peer
enable_peer	Enables the previously disabled replication peer
get_peer_config	Lists configuration parameters of the replication peer
update_peer_config	Updates configuration parameters of the replication peer
list_peer_configs	Lists configuration parameters of all replication peers
list_peers	Lists all replication peers
list_replicated_tables	Lists all tables and column families replicated from the cluster
show_peer_tableCFs	Shows replicable tables/column families for the replication peer
remove_peer_namespaces	Removes namespaces from the replication peer configuration
remove_peer_tableCFs	Removes tables/column families from the replication peer configuration
remove_peer	Disables and removes a replication peer

Snapshots commands

These commands allow users to work with table snapshots.

Command	Description
snapshot	Takes a snapshot of the specified table
list_snapshots	Returns a list of snapshots — all or matching the given regex
list_table_snapshots	Returns snapshots of the specified table
restore_snapshot	Restores the specified snapshot
clone_snapshot	Creates a new table by cloning the specified snapshot
delete_snapshot	Deletes a snapshot with the specified name
delete_all_snapshot	Deletes all snapshots with the name matching the given regex
delete_table_snapshots	Deletes snapshots of the specified table

Configuration commands

These commands are used to reload configuration parameters without restarting a cluster.

The set of configuration parameters that can be currently changed by this command is limited. See more details in the HBase documentation.

Command	Description
update_all_config	Reloads a subset of configuration parameters on all cluster servers
update_config	Reloads a subset of configuration parameters on the specified cluster servers

Quotas commands

These commands are used to work with quotas — resource and other restrictions applied to database objects.
Before using the commands from this group, add the following property to *hbase-site.xml* and restart your cluster.

Configuration

```
<property>
   <name>hbase.quota.enabled</name>
   <value>true</value>
</property>
```

Command	Description
set_quota	Sets a new quota for the defined user/table/namespace
list_quotas	Returns quotas added to HBase — all or matching the given regex
list_quota_table_sizes	Returns the computed size of each table in the cluster
list_quota_snapshots	Returns information about tables with defined space quotas
list_snapshot_sizes	Returns the size of snapshots made for tables with defined space quotas

Security commands

These commands are useful for security configuring needs.
Before using the commands from this group, add the following properties to *hbase-site.xml* and restart your cluster. If properties `hbase.coprocessor.master.classes` or `hbase.coprocessor.region.classes` already have another values, use comma-separated lists for keeping several values for them.

Configuration

```
<property>
   <name>hbase.coprocessor.master.classes</name>
   <value>org.apache.hadoop.hbase.security.access.AccessController</value>
</property>
<property>
   <name>hbase.coprocessor.region.classes</name>
   <value>org.apache.hadoop.hbase.security.access.AccessController</value>
</property>
<property>
   <name>hbase.rpc.engine</name>
   <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>
</property>
<property>
   <name>hbase.security.authorization</name>
   <value>true</value>
</property>
```

Command	Description
grant	Grants a user the specific rights
list_security_capabilities	Returns supported security capabilities
revoke	Revokes a user access rights
user_permission	Returns permissions for the specified user

Procedures commands

These commands are used to work with HBase procedures.

Command	Description
list_procedures	Returns all the procedures in HBase
abort_procedure	Aborts running of the procedure with the specified ID
list_locks	Returns all the locks in HBase

Visibility labels commands

These commands are used to control visibility labels, which permit users, associated with the given label, to read or access cells with that label.
Before using the commands from this group, add the following properties to *hbase-site.xml* and restart your cluster. If properties `hbase.coprocessor.master.classes` or `hbase.coprocessor.region.classes` already have another values, use comma-separated lists for keeping several values for them.

Configuration

```
<property>
   <name>hbase.coprocessor.master.classes</name>
   <value>org.apache.hadoop.hbase.security.visibility.VisibilityController</value>
</property>
<property>
   <name>hbase.coprocessor.region.classes</name>
   <value>org.apache.hadoop.hbase.security.visibility.VisibilityController</value>
</property>
<property>
   <name>hbase.security.authorization</name>
   <value>true</value>
</property>
<property>
   <name>hfile.format.version</name>
   <value>3</value>
</property>
```

Command	Description
add_labels	Adds a set of visibility labels
list_labels	Returns a list of available visibility labels
set_auths	Assigns a set of visibility labels to the defined user or group
get_auths	Gets visibility labels assigned to the defined user or group
clear_auths	Clears visibility labels assigned to the defined user or group
set_visibility	Sets the visibility expression on one or more existing table cells

Rsgroup commands

These commands allow users to manage Region Server groups — logical unions of Region Servers. Each group can mount different tables, if necessary. Thus, each group is equivalent to a logical subcluster. It helps to achieve resource isolation and to reduce management costs. The default Region Server group is called `default`.
Before using the commands from this group, add the following properties to *hbase-site.xml* and restart your cluster. If properties `hbase.coprocessor.master.classes` or `hbase.coprocessor.region.classes` already have another values, use comma-separated lists for keeping several values for them.

Configuration

```
<property>
   <name>hbase.coprocessor.master.classes</name>
   <value>org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint</value>
</property>
<property>
   <name>hbase.master.loadbalancer.class</name>
   <value>org.apache.hadoop.hbase.rsgroup.RSGroupBasedLoadBalancer</value>
</property>
```

Command	Description
add_rsgroup	Creates a new Region Server group
list_rsgroups	Returns all Region Server groups
move_servers_rsgroup	Assigns the given servers to the specified Region Server group
move_tables_rsgroup	Assigns the given tables to the specified Region Server group
move_servers_tables_rsgroup	Assigns the given servers and tables to the specified Region Server group
move_namespaces_rsgroup	Assigns the given namespaces to the specified Region Server group
move_servers_namespaces_rsgroup	Assigns the given servers and namespaces to the specified Region Server group
balance_rsgroup	Balances a Region Server group
get_rsgroup	Gets the Region Server group information
get_server_rsgroup	Gets a name of the group, which the given Region Server is a member of
get_table_rsgroup	Gets a name of the group, which the given table is a member of
remove_servers_rsgroup	Removes decommissioned servers from the Region Server group
remove_rsgroup	Removes the Region Server group

processlist

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Shows active tasks of Region Servers. Can be displayed in different view modes, depending on the input parameter.

Usage

```
processlist ['<view_mode>']
```

Arguments

Parameter	Description
view_mode	Possible values: - `all` — returns all types of tasks (described below); - `general` — returns non-RPC tasks, such as `compact`, `flush`, etc.; - `handler` — returns RPC Handler tasks; - `rpc` — returns RPC Handler tasks, which state is `RUNNING`; - `operation` — returns RPC Handler tasks, which state is `RUNNING` and from client. The default view mode is `general`

Examples

```
hbase(main):001:0> processlist 'general'
0 tasks as of: 2021-11-17 14:26:33
No general tasks currently running.
Took 0.5531 seconds
hbase(main):002:0> processlist 'rpc'
0 tasks as of: 2021-11-17 14:26:42
No rpc tasks currently running.
Took 0.0331 seconds
```

status

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - summary
  - simple
  - detailed
  - replication

Definition

Shows the current cluster status. Can be displayed in different view modes, depending on the input parameter.

Usage

```
status ['<view_mode>']
```

Arguments

Parameter	Description
view_mode	Possible values: ▪ `summary` — returns information about active and backup master servers, Region Servers, and average cluster load as a single line; ▪ `simple` — returns additional information about Region Servers, such as: number of Regions, number and sizes of store files, etc.; ▪ `detailed` — adds to the output of the previous view mode system log records and information about Regions transitions; ▪ `replication` — shows replication status. The default view mode is `summary`

Examples

summary

```
hbase(main):001:0> status
1 active master, 0 backup masters, 3 servers, 0 dead, 13.0000 average load
Took 0.3476 seconds
```

simple

```
hbase(main):003:0> status 'simple'
active master:  bds-adh-2.ru-central1.internal:16000 1637127966912
0 backup masters
3 live servers
    bds-adh-1.ru-central1.internal:16020 1637127969828
        requestsPerSecond=0.0, numberOfOnlineRegions=13, usedHeapMB=99, maxHeapMB=3905, numberOfStores=13, numberOfStorefiles=13,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeKB=0, readRequestsCount=0,
filteredReadRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0,
totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[GroupedAggregateRegionObserver, Indexer,
ScanRegionObserver, SequenceRegionObserver, ServerCachingEndpointImpl, UngroupedAggregateRegionObserver]
    bds-adh-2.ru-central1.internal:16020 1637127970177
        requestsPerSecond=0.0, numberOfOnlineRegions=14, usedHeapMB=54, maxHeapMB=3905, numberOfStores=15, numberOfStorefiles=15,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeKB=2, readRequestsCount=3891,
filteredReadRequestsCount=85, writeRequestsCount=76, rootIndexSizeKB=2, totalStaticIndexSizeKB=1, totalStaticBloomSizeKB=0,
totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[GroupedAggregateRegionObserver, Indexer,
MetaDataEndpointImpl, MultiRowMutationEndpoint, ScanRegionObserver, ServerCachingEndpointImpl, UngroupedAggregateRegionObserver]
    bds-adh-3.ru-central1.internal:16020 1637127969753
        requestsPerSecond=0.0, numberOfOnlineRegions=12, usedHeapMB=89, maxHeapMB=3905, numberOfStores=12, numberOfStorefiles=12,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeKB=0, readRequestsCount=1,
filteredReadRequestsCount=59080, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0,
totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[GroupedAggregateRegionObserver, Indexer,
MetaDataEndpointImpl, MetaDataRegionObserver, MultiRowMutationEndpoint, ScanRegionObserver, ServerCachingEndpointImpl,
UngroupedAggregateRegionObserver]
0 dead servers
Aggregate load: 0, regions: 39
Took 0.0264 seconds
```

detailed

```
hbase(main):004:0> status 'detailed'
version 2.0.2
0 regionsInTransition
active master:  bds-adh-2.ru-central1.internal:16000 1637127966912
0 backup masters
master coprocessors: []
3 live servers
    bds-adh-1.ru-central1.internal:16020 1637127969828
        requestsPerSecond=0.0, numberOfOnlineRegions=13, usedHeapMB=101, maxHeapMB=3905, numberOfStores=13,
numberOfStorefiles=13, storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeKB=0,
readRequestsCount=0, filteredReadRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=
[GroupedAggregateRegionObserver, Indexer, ScanRegionObserver, SequenceRegionObserver, ServerCachingEndpointImpl,
UngroupedAggregateRegionObserver]
        "SYSTEM.LOG,\\x01\\x00\\x00\\x00,1630426768774.96683724537040b683d0a24eb1f3bef6."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x02\\x00\\x00\\x00,1630426768774.346859a56a3dd85348d1d283ef830fc5."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x05\\x00\\x00\\x00,1630426768774.d11d2b44a4501c82e9cf2acdd50aaaf7."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x0A\\x00\\x00\\x00,1630426768774.287246b94f0727e6995dec5e7a5f6693."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x0C\\x00\\x00\\x00,1630426768774.7682bbcec9a40fc7bca560169eeff228."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x0E\\x00\\x00\\x00,1630426768774.759b0d6c372b17a9efdbbcbd2319e3a8."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x16\\x00\\x00\\x00,1630426768774.c98eb78752cf0073077f0e32277800dd."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x18\\x00\\x00\\x00,1630426768774.6d8cf50500196ce5b88b78c4b89da8a5."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x19\\x00\\x00\\x00,1630426768774.4ac5876e153fc8bc2e3eed202d935f33."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x1A\\x00\\x00\\x00,1630426768774.3f972f73d5d6f996c656083ae391075c."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x1B\\x00\\x00\\x00,1630426768774.93488f87483790344655d02ad3e085b3."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.MUTEX,,1630426773323.c28426b0346e789ba03bf338d649ec3a."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=1637054058546,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=1.0
        "SYSTEM.SEQUENCE,,1630426764347.7addb995515e262ebae6c4a21be37082."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
    bds-adh-2.ru-central1.internal:16020 1637127970177
        requestsPerSecond=0.0, numberOfOnlineRegions=14, usedHeapMB=56, maxHeapMB=3905, numberOfStores=15, numberOfStorefiles=15,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeKB=2, readRequestsCount=3891,
filteredReadRequestsCount=85, writeRequestsCount=76, rootIndexSizeKB=2, totalStaticIndexSizeKB=1, totalStaticBloomSizeKB=0,
totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[GroupedAggregateRegionObserver, Indexer,
MetaDataEndpointImpl, MultiRowMutationEndpoint, ScanRegionObserver, ServerCachingEndpointImpl, UngroupedAggregateRegionObserver]
        "SYSTEM.FUNCTION,,1630426767460.49cd07b4f3c549a9f1dbba531e8135ad."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x03\\x00\\x00\\x00,1630426768774.67b1c68668d4c37f2322d471665503fc."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x04\\x00\\x00\\x00,1630426768774.2b75793a1d9cf7d9ce3255539c4beca7."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x0B\\x00\\x00\\x00,1630426768774.811964f87fb0a84a7234d7a8d972eece."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x0D\\x00\\x00\\x00,1630426768774.14d9edf8f4ce99c1597595e133893c37."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x11\\x00\\x00\\x00,1630426768774.0a16d0cde0d9b68e9cfe82042c585770."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x14\\x00\\x00\\x00,1630426768774.78a990188fd99794c9553208429591e5."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x15\\x00\\x00\\x00,1630426768774.620fb8312d0ada2eaf1c84a1823ae001."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x17\\x00\\x00\\x00,1630426768774.c4c90bde3c271f7c1066999dd7e7d0c1."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x1C\\x00\\x00\\x00,1630426768774.f9fb782c652503128e76bb53f95cdf35."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,,1630426768774.664c1b894adf58a3c8b0961936da6e2b."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.STATS,,1630426765916.50e9d7b7b2f75034edffedc1ab3d4be7."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "hbase:meta,,1"
            numberOfStores=2, numberOfStorefiles=2, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=1637047655761,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=3887, writeRequestsCount=76, rootIndexSizeKB=2, totalStaticIndexSizeKB=1,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=1426,
dataLocality=1.0
        "hbase:namespace,,1630426726207.d4a22db7f8447b04e8be8ab0d6a1e8b3."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=4, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=1.0
    bds-adh-3.ru-central1.internal:16020 1637127969753
        requestsPerSecond=0.0, numberOfOnlineRegions=12, usedHeapMB=91, maxHeapMB=3905, numberOfStores=12, numberOfStorefiles=12,
storefileUncompressedSizeMB=0, storefileSizeMB=0, memstoreSizeMB=0, storefileIndexSizeKB=0, readRequestsCount=1,
filteredReadRequestsCount=59220, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0, totalStaticBloomSizeKB=0,
totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, coprocessors=[GroupedAggregateRegionObserver, Indexer,
MetaDataEndpointImpl, MetaDataRegionObserver, MultiRowMutationEndpoint, ScanRegionObserver, ServerCachingEndpointImpl,
UngroupedAggregateRegionObserver]
        "SYSTEM.CATALOG,,1630426759159.de2be860631b95097d3c710319e15053."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=1635260340542,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=1, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=1.0
        "SYSTEM.LOG,\\x06\\x00\\x00\\x00,1630426768774.99209a63b8aad46633c5bc197653874e."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x07\\x00\\x00\\x00,1630426768774.103e18787444a6d5d8b8760f457986f7."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x08\\x00\\x00\\x00,1630426768774.870b23dd7bf02f5581110acfe063f4f7."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x09\\x00\\x00\\x00,1630426768774.c4384d2f8fe4c35546e4ae770fddb476."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x0F\\x00\\x00\\x00,1630426768774.36f4b9e518d3f6d8b4aec209170ff6d3."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x10\\x00\\x00\\x00,1630426768774.68e7e7c2e5e430b4c779d7ae335f0cee."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x12\\x00\\x00\\x00,1630426768774.be32639f50a14ed50b7f9b4b779482f7."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x13\\x00\\x00\\x00,1630426768774.eb5cebd439b59fda2d6217b33d1942b9."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x1D\\x00\\x00\\x00,1630426768774.f86eb1f1018a991ebf95a8857f6c1032."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x1E\\x00\\x00\\x00,1630426768774.5754c2c290bc5dbc93b653cc891a2f4c."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
        "SYSTEM.LOG,\\x1F\\x00\\x00\\x00,1630426768774.a3574b91b1ca17bf18dd17e90e62d198."
            numberOfStores=1, numberOfStorefiles=1, storefileUncompressedSizeMB=0, lastMajorCompactionTimestamp=0,
storefileSizeMB=0, memstoreSizeMB=0, readRequestsCount=0, writeRequestsCount=0, rootIndexSizeKB=0, totalStaticIndexSizeKB=0,
totalStaticBloomSizeKB=0, totalCompactingKVs=0, currentCompactedKVs=0, compactionProgressPct=NaN, completeSequenceId=-1,
dataLocality=0.0
0 dead servers
Took 0.0255 seconds
=> #<Java::JavaUtil::Collections::UnmodifiableRandomAccessList:0x3de88f64>
```

replication

```
hbase(main):005:0> status 'replication'
version 2.0.2
3 live servers
    bds-adh-1.ru-central1.internal:
       SOURCE:
       SINK  : AgeOfLastAppliedOp=0, TimeStampsOfLastAppliedOp=Wed Nov 17 05:46:12 UTC 2021
    bds-adh-2.ru-central1.internal:
       SOURCE:
       SINK  : AgeOfLastAppliedOp=0, TimeStampsOfLastAppliedOp=Wed Nov 17 05:46:12 UTC 2021
    bds-adh-3.ru-central1.internal:
       SOURCE:
       SINK  : AgeOfLastAppliedOp=0, TimeStampsOfLastAppliedOp=Wed Nov 17 05:46:12 UTC 2021
Took 0.0139 seconds
=> #<Java::JavaUtil::Collections::UnmodifiableSet:0x290c266c>
```

table_help

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Shows a small hint for table-reference commands: create, put, get, scan, etc.

Usage

```
table_help
```

Examples

```
hbase(main):014:0> table_help
Help for table-reference commands.

You can either create a table via 'create' and then manipulate the table via commands like 'put', 'get', etc.
See the standard help information for how to use each of these commands.

However, as of 0.96, you can also get a reference to a table, on which you can invoke commands.
For instance, you can get create a table and keep around a reference to it via:

   hbase> t = create 't', 'cf'

Or, if you have already created the table, you can get a reference to it:

   hbase> t = get_table 't'

You can do things like call 'put' on the table:

  hbase> t.put 'r', 'cf:q', 'v'

which puts a row 'r' with column family 'cf', qualifier 'q' and value 'v' into table t.

To read the data out, you can scan the table:

  hbase> t.scan

which will read all the rows in table 't'.

Essentially, any command that takes a table name can also be done via table reference.
Other commands include things like: get, delete, deleteall,
get_all_columns, get_counter, count, incr. These functions, along with
the standard JRuby object methods are also available via tab completion.

For more information on how to use each of these commands, you can also just type:

   hbase> t.help 'scan'

which will output more information on how to use that command.

You can also do general admin actions directly on a table; things like enable, disable,
flush and drop just by typing:

   hbase> t.enable
   hbase> t.flush
   hbase> t.disable
   hbase> t.drop

Note that after dropping a table, your reference to it becomes useless and further usage
is undefined (and not recommended).
Took 0.0002 seconds
```

version

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Shows the current HBase instance version.

Usage

```
version
```

Examples

```
hbase(main):019:0> version
2.0.2, rUnknown, Thu Apr 15 20:20:26 UTC 2021
Took 0.0002 seconds
```

whoami

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Shows the current HBase user.

Usage

```
whoami
```

Examples

```
hbase(main):021:0> whoami
dasha (auth:SIMPLE)
    groups: dasha
Took 0.0090 seconds
```

alter

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Modifying a column family
  - Adding a new column family
  - Deleting a column family
  - Setting a table-scope parameter
  - Resetting a table-scope parameter
  - Setting a configuration option
  - Resetting a configuration option

Definition

Allows to add, modify or delete column families and change table configuration options. Column families are defined in the same way, as in the case of create command. Each column family specification can be either a simple string or a dictionary with mandatory `NAME` attribute.

Usage

- Adding or modifying a column family:

```
alter '[<namespace_name>:]<table_name>',
      '<column_family>' | {NAME => '<column_family>'[, <cf_attribute_name> => <cf_attribute_value>,...]}[,...]
```

- Deleting a column family:
  - Method 1:

```
alter '[<namespace_name>:]<table_name>',
      NAME => '<column_family>',
      METHOD => 'delete'
```

  - Method 2:

```
alter '[<namespace_name>:]<table_name>',
      'delete' => '<column_family>'
```

- Setting table-scope parameters:

```
alter '[<namespace_name>:]<table_name>',
      <table_attribute_name> => <table_attribute_value>[,...]
```

- Resetting table-scope parameters:

```
alter '[<namespace_name>:]<table_name>',
      METHOD => 'table_att_unset',
      NAME => '<table_attribute_name>'
```

- Setting configuration options:

```
alter '[<namespace_name>:]<table_name>',
      CONFIGURATION => {'<conf_attribute_name>' => <conf_attribute_value>[,...]}
```

- Resetting configuration options:

```
alter '[<namespace_name>:]<table_name>',
      METHOD => 'table_conf_unset',
      NAME => '<conf_attribute_name>'
```

TIP

There could be more than one alteration in one command.

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
column_family	A column family name
cf_attribute_name	A column family attribute name
cf_attribute_value	A value of the specified column family attribute
table_attribute_name	A table-scope attribute name
table_attribute_value	A value of the specified table-scope attribute
conf_attribute_name	A configuration attribute name
conf_attribute_value	A value of the specified configuration attribute

Examples

Modifying a column family

```
hbase(main):001:0> describe 't1'
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
2 row(s)
Took 0.5144 seconds
hbase(main):002:0> alter 't1', {NAME => 'cf1', VERSIONS => 10}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.5811 seconds
hbase(main):003:0> describe 't1'
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '10', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
2 row(s)
Took 0.0206 seconds
```

Adding a new column family

```
hbase(main):004:0> alter 't1', 'cf3'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.1848 seconds
hbase(main):005:0> describe 't1'
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '10', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'}
{NAME => 'cf3', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
3 row(s)
Took 0.0183 seconds
```

Deleting a column family

```
hbase(main):007:0> alter 't1', NAME => 'cf3', METHOD => 'delete'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.5314 seconds
hbase(main):008:0> describe 't1'
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '10', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE
=> '0', BLOOMFILTER => 'ROW', CACHE_INDEX_ON_WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false',
PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE => 'true', BLOCKSIZE => '65536'}
2 row(s)
Took 0.0175 seconds
```

Setting a table-scope parameter

```
hbase(main):001:0> alter 't1', MAX_FILESIZE => '134217728'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.9316 seconds
```

Resetting a table-scope parameter

```
hbase(main):002:0> alter 't1', METHOD => 'table_att_unset', NAME => 'MAX_FILESIZE'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.3027 seconds
```

Setting a configuration option

```
hbase(main):003:0> alter 't1', CONFIGURATION => {'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.3146 seconds
```

Resetting a configuration option

```
hbase(main):004:0> alter 't1', METHOD => 'table_conf_unset', NAME => 'hbase.hregion.scan.loadColumnFamiliesOnDemand'
Updating all regions with the new schema...
1/1 regions updated.
Done.
Took 2.4628 seconds
```

alter_async

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Works in the same way as alter command, but does not wait for all Regions to receive the schema changes. So, unlike the alter command, the output does not contain the message "Updating all regions with the new schema…".

Usage

- Adding or modifying a column family:

```
alter_async '[<namespace_name>:]<table_name>',
            '<column_family>' | {NAME => '<column_family>'[, <cf_attribute_name> => <cf_attribute_value>,...]}[,...]
```

- Deleting a column family:
  - Method 1:

```
alter_async '[<namespace_name>:]<table_name>',
            NAME => '<column_family>',
            METHOD => 'delete'
```

  - Method 2:

```
alter_async '[<namespace_name>:]<table_name>',
            'delete' => '<column_family>'
```

- Setting table-scope parameters:

```
alter_async '[<namespace_name>:]<table_name>',
            <table_attribute_name> => <table_attribute_value>[,...]
```

- Resetting table-scope parameters:

```
alter_async '[<namespace_name>:]<table_name>',
            METHOD => 'table_att_unset',
            NAME => '<table_attribute_name>'
```

- Setting configuration options:

```
alter_async '[<namespace_name>:]<table_name>',
            CONFIGURATION => {'<conf_attribute_name>' => <conf_attribute_value>[,...]}
```

- Resetting configuration options:

```
alter_async '[<namespace_name>:]<table_name>',
            METHOD => 'table_conf_unset',
            NAME => '<conf_attribute_name>'
```

> TIP
>
> There could be more than one alteration in one command.

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
column_family	A column family name
cf_attribute_name	A column family attribute name
cf_attribute_value	A value of the specified column family attribute
table_attribute_name	A table-scope attribute name
table_attribute_value	A value of the specified table-scope attribute
conf_attribute_name	A configuration attribute name
conf_attribute_value	A value of the specified configuration attribute

Examples

```
hbase(main):006:0> alter_async 't1', 'cf3'
Took 1.2380 seconds
hbase(main):007:0> describe 't1'
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '10', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'f
alse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
 IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536
'}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
{NAME => 'cf3', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
3 row(s)
Took 0.0181 seconds
```

alter_status

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns the status of the alter command indicating the number of table Regions that have received the updated schema.

Usage

```
alter_status '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):013:0> alter_status 't1'
1/1 regions updated.
Done.
Took 1.0178 seconds
```

create

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Creating a table in the default namespace without dictionaries
  - Using a new namespace and dictionaries for column families defining
  - Defining table-scope parameters after column families specifications
  - Defining configuration options after column families specifications

Definition

Creates a new table. Requires a table name and at least one column family specification. Optionally, you can define table-scope and configuration parameters. Each column specification can be a simple string containing a column family name, or a dictionary using a key/value syntax with curly-braces. In the second case specifying the `NAME` attribute is mandatory.

Usage

```
create '[<namespace_name>:]<table_name>',
       '<column_family>' | {NAME => '<column_family>'[,
                           <cf_attribute_name> => <cf_attribute_value>,...]
                           }[,...][,
       {<table_attribute_name> => <table_attribute_value>,[...]}][,
       CONFIGURATION => {'<conf_attribute_name>' => <conf_attribute_value>[,...]}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
column_family	A column family name
cf_attribute_name	A column family attribute name
cf_attribute_value	A value of the specified column family attribute
table_attribute_name	A table-scope attribute name
table_attribute_value	A value of the specified table-scope attribute
conf_attribute_name	A configuration attribute name
conf_attribute_value	A value of the specified configuration attribute

Examples

Creating a table in the default namespace without dictionaries

```
hbase(main):005:0> create 't1', 'cf1', 'cf2'
Created table t1
Took 1.2482 seconds
=> Hbase::Table - t1
```

Using a new namespace and dictionaries for column families defining

```
hbase(main):001:0> create 'ns1:t2', {NAME => 'cf1', TTL => 2592000}, {NAME => 'cf2', VERSIONS => 5}
Created table ns1:t2
Took 1.6133 seconds
=> Hbase::Table - ns1:t2
```

Defining table-scope parameters after column families specifications

```
hbase(main):019:0> create 't3', 'cf1', {NUMREGIONS => 15, SPLITALGO => 'HexStringSplit'}
Created table t3
Took 1.2313 seconds
=> Hbase::Table - t3
```

Defining configuration options after column families specifications

```
hbase(main):016:0> create 't4', 'cf1', CONFIGURATION => {'hbase.hregion.scan.loadColumnFamiliesOnDemand' => 'true'}
Created table t4
Took 1.3492 seconds
=> Hbase::Table - t4
```

describe

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Describing a table from the default namespace
  - Describing a table from the different namespace

Definition

Displays the defined table schema. Alternatively, you can use the abbreviated `desc`.

Usage

```
describe '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

Describing a table from the default namespace

```
hbase(main):001:0> describe 't1'
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
2 row(s)
Took 0.4877 seconds
```

Describing a table from the different namespace

```
hbase(main):002:0> describe 'ns1:t2'
Table ns1:t2 is ENABLED
ns1:t2
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => '2592000 SECONDS (30 DAYS)', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0',
BLOOMFILTER => 'ROW', CACHE_INDEX_ON_
WRITE => 'false', IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION =>
'NONE', BLOCKCACHE => 'true', BL
OCKSIZE => '65536'}
{NAME => 'cf2', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
2 row(s)
Took 0.0163 seconds
```

disable

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Disables the given table. You cannot apply any data operations to disabled tables.

IMPORTANT

Execution of the `disable` command is mandatory before applying the drop, drop_all, restore_snapshot commands.

Usage

```
disable '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):002:0> disable 't1'
Took 1.1979 seconds
hbase(main):003:0> disable 'ns1:t2'
Took 0.7729 seconds
```

disable_all

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Disables all the tables matching the given regular expression. At first it locates the tables matching the regex, requests confirmation of the operation, and then disables them.

Usage

```
disable_all '<table_regex>'
```

Arguments

Parameter	Description
table_regex	A regular expression for searching tables

Examples

```
hbase(main):005:0> disable_all 't.'
t1
t3

Disable the above 2 tables (y/n)?
y
2 tables successfully disabled
Took 7.2381 seconds
```

drop

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Deletes the given table permanently. Can be applied only to disabled tables.

Usage

```
drop '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):010:0> drop "ns1:t2"

ERROR: Table ns1:t2 is enabled. Disable it first.

Drop the named table. Table must first be disabled:
  hbase> drop 't1'
  hbase> drop 'ns1:t1'

Took 0.0351 seconds
hbase(main):011:0> disable "ns1:t2"
Took 0.8114 seconds
hbase(main):012:0> drop "ns1:t2"
Took 0.4729 seconds
```

drop_all

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Deletes the tables matching the given regular expression. Can be applied only to disabled tables. At first it locates the tables matching the regex, requests confirmation of the operation, and then deletes them permanently.

Usage

```
drop_all '<table_regex>'
```

Arguments

Parameter	Description
table_regex	A regular expression for searching tables

Examples

```
hbase(main):013:0> drop_all 't.'
t1
t3

Drop the above 2 tables (y/n)?
y
0 tables successfully dropped
2 tables not dropped due to an exception: t1,t3
Took 4.2576 seconds
hbase(main):014:0> disable_all 't.'
t1
t3

Disable the above 2 tables (y/n)?
y
2 tables successfully disabled
Took 4.4517 seconds
hbase(main):015:0> drop_all 't.'
t1
t3

Drop the above 2 tables (y/n)?
y
2 tables successfully dropped
Took 3.0036 seconds
```

enable

Daria Barysheva

Contents

To Table of Contents

Usage

Examples

Enables the given table. You can apply any data operations to only enabled tables.

IMPORTANT

Do not forget to apply `enable` after executing the drop, drop_all, restore_snapshot commands.

Usage

```
enable '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):010:0> enable 'ns1:t2'
Took 0.7863 seconds
```

enable_all

Daria Barysheva

Contents

To Table of Contents

Usage

Examples

Enables all the tables matching the given regular expression. At first it locates the tables matching the regex, requests confirmation of the operation, and then enables them.

Usage

```
enable_all '<table_regex>'
```

Arguments

Parameter	Description
table_regex	A regular expression for searching tables

Examples

```
hbase(main):012:0> enable_all 't.'
t1
t3

Enable the above 2 tables (y/n)?
y
2 tables successfully enabled
Took 10.9382 seconds
```

exists

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
  Checking an existing table
  Checking a non-existent table

Definition

Checks whether the table exists.

Usage

```
exists '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

Checking an existing table

```
hbase(main):017:0> exists 't1'
Table t1 does exist
Took 0.0077 seconds
=> true
```

Checking a non-existent table

```
hbase(main):018:0> exists 'ns1:t1'
Table ns1:t1 does not exist
Took 0.0072 seconds
=> false
```

get_table

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Locates a table by the given name and returns it as an object, available for user manipulations. All subsequent operations can be applied to this object, without specifying the table name.

Usage

```
<object_name> = get_table '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
object_name	A name of the variable, which will refer to the table and allow you to call it without specifying its name
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):025:0> t1 = get_table 't1'
Took 0.0003 seconds
=> Hbase::Table - t1
hbase(main):027:0> t1.describe
Table t1 is ENABLED
t1
COLUMN FAMILIES DESCRIPTION
{NAME => 'cf1', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
{NAME => 'cf2', VERSIONS => '1', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'fa
lse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536'
}
2 row(s)
Took 0.0223 seconds
hbase(main):030:0> t1.put 'r1', 'cf1:c1', 'v2'
Took 0.0051 seconds
```

is_enabled

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns `true` if the given table is currently enabled, otherwise returns `false`.

Usage

```
is_enabled '[namespace_name:]table_name'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):013:0> is_enabled 't3'
true
Took 0.0146 seconds
=> true
```

is_disabled

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns `true` if the given table is currently disabled, otherwise returns `false`.

Usage

```
is_disabled '[namespace_name:]table_name'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):007:0> is_disabled 't3'
true
Took 0.0102 seconds
=> 1
```

list

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Listing of all tables
  - Listing of tables matching the regular expression

Definition

Returns all tables in HBase. An optional regular expression parameter can be used to filter the output.

Usage

```
list ['[<namespace_name>:]<table_name>' | 'table_regex']
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
table_regex	A regular expression for searching tables

Examples

Listing of all tables

```
hbase(main):010:0> list
TABLE
SYSTEM.CATALOG
SYSTEM.FUNCTION
SYSTEM.LOG
SYSTEM.MUTEX
SYSTEM.SEQUENCE
SYSTEM.STATS
ns1:t2
t1
t3
9 row(s)
Took 0.0044 seconds
=> ["SYSTEM.CATALOG", "SYSTEM.FUNCTION", "SYSTEM.LOG", "SYSTEM.MUTEX", "SYSTEM.SEQUENCE", "SYSTEM.STATS", "ns1:t2", "t1", "t3"]
```

Listing of tables matching the regular expression

```
hbase(main):015:0> list 't.'
TABLE
t1
t3
2 row(s)
Took 0.0034 seconds
=> ["t1", "t3"]
```

list_regions

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Listing Regions without using filters
  - Listing Regions for the specified Region Server
  - Restricting number of output fields

Definition

Displays all Regions for the specified table.

Usage

```
list_regions '[namespace_name:]<table_name>'[,
             {<attr_name> => '<attr_value>'[,...]}[,
             <output_fields[]>]]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
attr_name	An optional parameter used for filtering results. Possible values: `'SERVER_NAME'` (a Region Server name), `'LOCALITY_THRESHOLD'` (a Region locality)
attr_value	A desired value of the `attr_name` parameter
output_fields	An optional array containing one or several fields that should be displayed in the output information. Possible values: `'SERVER_NAME'`, `'REGION_NAME'`, `'START_KEY'`, `'END_KEY'`, `'SIZE'`, `'REQ'`, `'LOCALITY'`. You can use one of them or several. Values are not case sensitive. By default, all the fields are displayed (see their description below)

Output parameters

Parameter	Description
SERVER_NAME	A Region Server name
REGION_NAME	A full Region name
START_KEY	A start row key for the Region
END_KEY	An end row key for the Region
SIZE	A Region size in MB
REQ	A number of requests to the Region
LOCALITY	A Region locality (`1` in the case of flushing information into HFiles)

Examples

Listing Regions without using filters

```
hbase(main):007:0> list_regions 't1'
                                        SERVER_NAME |                                         REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | --------------------------------------------------- | ---------- | --------
-- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1637214343541 | t1,,1637234844003.562aabcac6ff0688b31260c44c9863c8. |            |
|     0 |     0 |        1.0 |
 1 rows
Took 0.5142 seconds
```

Listing Regions for the specified Region Server

```
hbase(main):010:0> list_regions 't1', {SERVER_NAME => 'bds-adh-2.ru-central1.internal,16020,1637214343541'}
                                        SERVER_NAME |                                         REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | --------------------------------------------------- | ---------- | --------
-- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1637214343541 | t1,,1637234844003.562aabcac6ff0688b31260c44c9863c8. |            |
|     0 |     0 |        1.0 |
 1 rows
Took 0.0180 seconds
```

Restricting number of output fields

```
hbase(main):011:0> list_regions 't1', {}, ['SERVER_NAME','REGION_NAME']
                                        SERVER_NAME |                                         REGION_NAME |
 -------------------------------------------------- | --------------------------------------------------- |
 bds-adh-2.ru-central1.internal,16020,1637214343541 | t1,,1637234844003.562aabcac6ff0688b31260c44c9863c8. |
 1 rows
Took 0.0181 seconds
```

locate_region

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Locates a Region by the given table name and the row key.

Usage

```
locate_region '[<namespace_name>:]<table_name>', '<row_key>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key

Examples

```
hbase(main):021:0> put 't1', 'r1', 'cf1:c1', 'v1'
Took 0.1000 seconds
hbase(main):022:0> locate_region 't1', 'r1'
HOST                                     REGION
 bds-adh-2.ru-central1.internal:16020    {ENCODED => 562aabcac6ff0688b31260c44c9863c8, NAME => 't1,,1637234844003.
562aabcac6ff0688b31260c44c9863c8.', STARTKEY => '', ENDKEY => ''}
1 row(s)
Took 0.0070 seconds
=> #<Java::OrgApacheHadoopHbase::HRegionLocation:0x6814262d>
```

show_filters

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Shows all the filters available in HBase.

Usage

```
show_filters
```

Examples

```
hbase(main):010:0> show_filters
DependentColumnFilter
KeyOnlyFilter
ColumnCountGetFilter
SingleColumnValueFilter
PrefixFilter
SingleColumnValueExcludeFilter
FirstKeyOnlyFilter
ColumnRangeFilter
ColumnValueFilter
TimestampsFilter
FamilyFilter
QualifierFilter
ColumnPrefixFilter
RowFilter
MultipleColumnPrefixFilter
InclusiveStopFilter
PageFilter
ValueFilter
ColumnPaginationFilter
Took 0.0026 seconds
=> #<Java::JavaUtil::HashMap::KeySet:0x2aafa84f>
```

alter_namespace

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Adding a new property
  - Changing a property value
  - Deleting a property

Definition

Changes the properties defined while the namespace creation.

Usage

- Adding/modifying the namespace properties:

```
alter_namespace '<namespace_name>',
                {METHOD => 'set',
                '<property_name>' => '<new_property_value>'}
```

- Deleting the namespace properties:

```
alter_namespace '<namespace_name>',
                {METHOD => 'unset',
                NAME => '<property_name>'}
```

Arguments

Parameter	Description
namespace_name	A namespace name
property_name	A property name
new_property_value	A new value for the specified property

Examples

Adding a new property

```
hbase(main):009:0> alter_namespace 'ns2', {METHOD => 'set', 'prop3' => 'value3'}
Took 0.2304 seconds
hbase(main):010:0> describe_namespace 'ns2'
DESCRIPTION
{NAME => 'ns2', prop1 => 'val1', prop2 => 'val2', prop3 => 'value3'}
Took 0.0039 seconds
=> 1
```

Changing a property value

```
hbase(main):011:0> alter_namespace 'ns2', {METHOD => 'set', 'prop3' => 'value3_new'}
Took 0.2364 seconds
hbase(main):012:0> describe_namespace 'ns2'
DESCRIPTION
{NAME => 'ns2', prop1 => 'val1', prop2 => 'val2', prop3 => 'value3_new'}
Took 0.0072 seconds
=> 1
```

Deleting a property

```
hbase(main):013:0>  alter_namespace 'ns2', {METHOD => 'unset', NAME => 'prop3'}
Took 0.2491 seconds
hbase(main):014:0> describe_namespace 'ns2'
DESCRIPTION
{NAME => 'ns2', prop1 => 'val1', prop2 => 'val2'}
Took 0.0069 seconds
=> 1
```

create_namespace

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Creating a new namespace without properties
  - Including a table into the created namespace
  - Creating a new namespace with properties

Definition

Creates a new namespace — a logical grouping of tables that is similar to a database in relation database systems. Requires a namespace name and optionally a dictionary of namespace configuration properties.

Usage

```
create_namespace '<namespace_name>'[,
                 {'<property_name>' => '<property_value>'[,...]}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
property_name	A property name
property_value	A property value

Examples

Creating a new namespace without properties

```
hbase(main):020:0> create_namespace 'ns1'
Took 0.2599 seconds
```

Including a table into the created namespace

Namespace membership is defined while table creation by specifying a fully-qualified table name in the form: `<namespace_name>:<table_name>`.

> **NOTE**
>
> For all subsequent accesses to the table, it is required to specify the full table name: `<namespace_name>:<table_name>`.

```
hbase(main):021:0> create 'ns1:table1','f1'
Created table ns1:table1
Took 1.3259 seconds
=> Hbase::Table - ns1:table1
```

Creating a new namespace with properties

```
hbase(main):022:0> create_namespace 'ns2', {'prop1' => 'val1', 'prop2' => 'val2'}
Took 0.2446 seconds
```

describe_namespace

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Returns the namespace description.

Usage

```
describe_namespace '<namespace_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name

Examples

```
hbase(main):024:0> describe_namespace 'ns1'
DESCRIPTION
{NAME => 'ns1'}
Took 0.0162 seconds
=> 1
hbase(main):025:0> describe_namespace 'ns2'
DESCRIPTION
{NAME => 'ns2', prop1 => 'val1', prop2 => 'val2'}
Took 0.0029 seconds
=> 1
```

drop_namespace

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Dropping an empty namespace
  - Dropping a non-empty namespace

Definition

Deletes a namespace permanently.

> **CAUTION**
>
> The namespace must be empty. You cannot drop the namespace that includes one or more tables.

Usage

```
drop_namespace '<namespace_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name

Examples

Dropping an empty namespace

```
hbase(main):002:0> drop_namespace 'ns2'
Took 0.2507 seconds
hbase(main):003:0> describe_namespace 'ns2'

ERROR: Unknown namespace ns2!

Describe the named namespace. For example:
  hbase> describe_namespace 'ns1'

Took 0.0096 seconds
```

Dropping a non-empty namespace

```
hbase(main):001:0> drop_namespace 'ns1'

ERROR: org.apache.hadoop.hbase.constraint.ConstraintException: Only empty namespaces can be removed. Namespace ns1 has 1 tables
        at org.apache.hadoop.hbase.master.procedure.DeleteNamespaceProcedure.prepareDelete(DeleteNamespaceProcedure.java:217)
        at org.apache.hadoop.hbase.master.procedure.DeleteNamespaceProcedure.executeFromState(DeleteNamespaceProcedure.java:78)
        at org.apache.hadoop.hbase.master.procedure.DeleteNamespaceProcedure.executeFromState(DeleteNamespaceProcedure.java:45)
        at org.apache.hadoop.hbase.procedure2.StateMachineProcedure.execute(StateMachineProcedure.java:184)
        at org.apache.hadoop.hbase.procedure2.Procedure.doExecute(Procedure.java:910)
        at org.apache.hadoop.hbase.procedure2.ProcedureExecutor.execProcedure(ProcedureExecutor.java:1674)
        at org.apache.hadoop.hbase.procedure2.ProcedureExecutor.executeProcedure(ProcedureExecutor.java:1461)
        at org.apache.hadoop.hbase.procedure2.ProcedureExecutor.access$900(ProcedureExecutor.java:76)
        at org.apache.hadoop.hbase.procedure2.ProcedureExecutor$WorkerThread.run(ProcedureExecutor.java:1973)

Drop the named namespace. The namespace must be empty.

Took 1.0703 seconds
```

list_namespace

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Listing of all namespaces
  - Listing of namespaces matching the regular expression

Definition

Returns all namespaces in HBase. An optional regular expression parameter can be used to filter the output.

Usage

```
list_namespace ['namespace_name' | 'namespace_regex']
```

Arguments

Parameter	Description
namespace_name	A namespace name
namespace_regex	A regular expression for searching namespaces

Examples

Listing of all namespaces

```
hbase(main):001:0> list_namespace
NAMESPACE
default
hbase
ns1
ns2
4 row(s)
Took 0.3307 seconds
```

Listing of namespaces matching the regular expression

```
hbase(main):002:0> list_namespace 'ns*'
NAMESPACE
ns1
ns2
2 row(s)
Took 0.0074 seconds
```

list_namespace_tables

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns all the tables included into the namespace.

Usage

```
list_namespace_tables '<namespace_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name

Examples

```
hbase(main):001:0> list_namespace_tables 'ns1'
TABLE
table1
1 row(s)
Took 0.3366 seconds
=> ["table1"]
```

append

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- Appending the previous cell value with some postfix
- Changing attributes
- Changing visibility

Definition

Appends a cell value at the specified table/row/column coordinates with the given postfix. It also allows to modify the custom attributes or visibility of this value (only one thing at a time).

Usage

```
append '[<namespace_name>:]<table_name>',
       '<row_key>',
       '<column_family>:<column_qualifier>',
       '<cell_value_postfix>'[,
       {ATTRIBUTES => {'<cell_attribute_name>' => '<cell_attribute_value>'[,...]}}] | [, {VISIBILITY => '<visibility_expression>'}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
column_family	A column family name
column_qualifier	A column qualifier
cell_value_postfix	A postfix for appending the specified cell value
cell_attribute_name	A cell attribute name
cell_attribute_value	A value of the specified cell attribute
visibility_expression	A logical expression that refers to existing visibility labels. Can use the following logical operators: `|` (or), `&` (and). Examples: `SECRET|PRIVATE`, `(SECRET&PRIVATE)|PUBLIC`, `SECRET&PRIVATE&PUBLIC`

Examples

Appending the previous cell value with some postfix

```
hbase(main):029:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
 cf1:c3                                  timestamp=1637324768581, value=value3
 cf1:c4                                  timestamp=1637324859512, value=value4
1 row(s)
Took 0.0092 seconds
hbase(main):030:0> append 't4', 'r1', 'cf1:c3', '_new'
CURRENT VALUE = value3_new
Took 0.0118 seconds
hbase(main):031:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
 cf1:c3                                  timestamp=1637326066133, value=value3_new
 cf1:c4                                  timestamp=1637324859512, value=value4
1 row(s)
Took 0.0114 seconds
```

Changing attributes

```
hbase(main):004:0> append 't4', 'r1', 'cf1:c3', '', ATTRIBUTES => {'mykey' => 'myvalue'}
CURRENT VALUE = value3_new
Took 0.5597 seconds
hbase(main):005:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
 cf1:c3                                  timestamp=1637326657852, value=value3_new
 cf1:c4                                  timestamp=1637324859512, value=value4
```

Changing visibility

```
hbase(main):006:0> append 't4', 'r1', 'cf1:c3', '', {VISIBILITY => 'PRIVATE'}
CURRENT VALUE = value3_new
Took 0.0220 seconds
hbase(main):007:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
 cf1:c3                                  timestamp=1637326735791, value=value3_new
 cf1:c4                                  timestamp=1637324859512, value=value4
1 row(s)
Took 0.0117 seconds
```

count

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- Counting a number of all rows in the table
- Counting a number of rows having the specified column
- Counting a number of rows matching the defined filter
- Counting a number of rows having the keys between STARTROW and STOPROW
- Using INTERVAL and CACHE parameters

Definition

Returns the number of rows in the table. As calculating can take a long time, the current count is shown every 1000 rows by default — this interval can be changed by setting the input parameter `INTERVAL`.

Scan caching is also enabled. The default cache size is 10 rows — this value can be changed by setting the input parameter `CACHE`. If your rows are small in size, you can increase its default value.

Usage

- The basic syntax:

```
count '[<namespace_name>:]<table_name>'[,
      COLUMNS => <columns_array[]>][,
      STARTROW => '<start_row_key>'][,
      STOPROW => '<stop_row_key>']
```

- Using optimization parameters:

```
count '[<namespace_name>:]<table_name>'[,
      INTERVAL => <interval_value>][,
      CACHE => <cache_value>]
```

- Using filters:

```
count '[<namespace_name>:]<table_name>',
      {FILTER => "<filter_description>"}
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
columns_array	An array of strings specified in the format `'<column_family>:<column_qualifier>'`, where `<column_family>` — a column family name; `<column_qualifier>` — a column qualifier
start_row_key	A start value for searching row keys
stop_row_key	A stop value for searching row keys
interval_value	An interval for showing the intermediate result
cache_value	A cache size
filter_description	A filter used for searching the table cells

Examples

Counting a number of all rows in the table

```
hbase(main):024:0> count 't4'
1 row(s)
Took 0.0175 seconds
=> 1
```

Counting a number of rows having the specified column

```
hbase(main):029:0> count 't4', COLUMNS => ['cf1:1']
0 row(s)
Took 0.0079 seconds
=> 0
hbase(main):030:0> count 't4', COLUMNS => ['cf1:c1']
1 row(s)
Took 0.0053 seconds
=> 1
```

Counting a number of rows matching the defined filter

```
hbase(main):027:0> count 't4', {FILTER => "ValueFilter(=, 'regexstring:value*')"}
1 row(s)
Took 0.0346 seconds
=> 1
```

Counting a number of rows having the keys between STARTROW and STOPROW

```
hbase(main):033:0> count 't4', COLUMNS => ['cf1:c1'], STARTROW => 'r1', STOPROW => 'r2'
1 row(s)
Took 0.0064 seconds
=> 1
```

Using INTERVAL and CACHE parameters

```
hbase(main):025:0> count 't4', INTERVAL => 100000, CACHE => 1000
1 row(s)
Took 0.0112 seconds
=> 1
```

delete

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Deleting a value at the specified table/row/column coordinates that has no previous versions
  - Deleting a value at the specified table/row/column coordinates that has previous versions
  - Deleting the specified value version

Definition

Deletes a cell value at the specified table/row/column and optionally timestamp coordinates.

Usage

```
delete '[<namespace_name>:]<table_name>',
       '<row_key>',
       '<column_family:column_qualifier>'[,
       <cell_timestamp>][,
       {VISIBILITY => '<visibility_expression>'}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
column_family	A column family name
column_qualifier	A column qualifier
cell_timestamp	A cell timestamp
visibility_expression	A logical expression that refers to existing visibility labels. Can use the following logical operators: `|` (or), `&` (and). Examples: `SECRET|PRIVATE`, `(SECRET&PRIVATE)|PUBLIC`, `SECRET&PRIVATE&PUBLIC`

Examples

Deleting a value at the specified table/row/column coordinates that has no previous versions

```
hbase(main):002:0> scan 't4'
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
1 row(s)
Took 0.3796 seconds
hbase(main):003:0> delete 't4', 'r1', 'cf1:c1'
Took 0.0750 seconds
hbase(main):004:0> scan 't4'
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
1 row(s)
Took 0.0086 seconds
```

Deleting a value at the specified table/row/column coordinates that has previous versions

```
hbase(main):009:0> delete 't4', 'r1', 'cf1:c5'
Took 0.0181 seconds
hbase(main):010:0> scan 't4'
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329190124, value=7738718a
1 row(s)
Took 0.0074 seconds
```

Deleting the specified value version

```
hbase(main):011:0> scan 't4', {VERSIONS => 5}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329190124, value=7738718a
 r1                                      column=cf1:c5, timestamp=1637329187604, value=7738718W
 r1                                      column=cf1:c5, timestamp=1637328326920, value=7738718M
1 row(s)
Took 0.0091 seconds
hbase(main):012:0> delete 't4', 'r1', 'cf1:c5', 1637329187604
Took 0.0047 seconds
hbase(main):013:0> scan 't4', {VERSIONS => 5}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329190124, value=7738718a
 r1                                      column=cf1:c5, timestamp=1637328326920, value=7738718M
1 row(s)
Took 0.0078 seconds
```

deleteall

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Deleting all cells values from the specified row
  - Using ROWPREFIXFILTER
  - Deleting all cells values from the specified column

Definition

Deletes all cells with previous versions from the specified row. Requires a table name, a row key, and optionally — a column name and a timestamp. Allows to use a prefix filter instead of a row key.

> CAUTION
>
> Even if you specify a timestamp, the command deletes all cells of the specified column, not only having the defined timestamp.

Usage

```
deleteall '[<namespace_name>:]<table_name>',
          '<row_key>' | {ROWPREFIXFILTER => '<row_prefix>'}[,
          '<column_family>:<column_qualifier>'[,
          <cell_timestamp>]][,
          {VISIBILITY => '<visibility_expression>'}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
row_prefix	A prefix for searching row keys
column_family	A column family name
column_qualifier	A column qualifier
cell_timestamp	A cell timestamp
visibility_expression	A logical expression that refers to existing visibility labels. Can use the following logical operators: `|` (or), `&` (and). Examples: `SECRET|PRIVATE`, `(SECRET&PRIVATE)|PUBLIC`, `SECRET&PRIVATE&PUBLIC`

Examples

Deleting all cells values from the specified row

```
hbase(main):079:0> scan 't6', VERSIONS => 5
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637426986008, value=value3
 r1                                      column=cf1:c1, timestamp=1637426972064, value=value2
 r1                                      column=cf1:c1, timestamp=1637426961063, value=value1
 r1                                      column=cf1:c2, timestamp=1637426521080, value=value4
 r2                                      column=cf1:c3, timestamp=1637427026420, value=value6
 r2                                      column=cf1:c3, timestamp=1637426999070, value=value5
2 row(s)
Took 0.0066 seconds
hbase(main):080:0> deleteall 't6', 'r1'
Took 0.0067 seconds
hbase(main):081:0> scan 't6', VERSIONS => 5
ROW                                      COLUMN+CELL
 r2                                      column=cf1:c3, timestamp=1637427026420, value=value6
 r2                                      column=cf1:c3, timestamp=1637426999070, value=value5
1 row(s)
Took 0.0054 seconds
```

Using ROWPREFIXFILTER

```
hbase(main):048:0> scan 't6', VERSIONS => 5
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637426380000, value=value3
 r1                                      column=cf1:c1, timestamp=1637426378038, value=value2
 r1                                      column=cf1:c1, timestamp=1637426371683, value=value1
 r1                                      column=cf1:c2, timestamp=1637426387651, value=value4
 r2                                      column=cf1:c3, timestamp=1637426397389, value=value6
 r2                                      column=cf1:c3, timestamp=1637426395621, value=value5
2 row(s)
Took 0.0437 seconds
hbase(main):049:0> deleteall 't6', {ROWPREFIXFILTER => 'r'}
Took 0.0546 seconds
hbase(main):050:0> scan 't6', VERSIONS => 5
ROW                                      COLUMN+CELL
0 row(s)
Took 0.0051 seconds
```

Deleting all cells values from the specified column

```
hbase(main):057:0> scan 't6', VERSIONS => 5
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637426519471, value=value3
 r1                                      column=cf1:c1, timestamp=1637426517669, value=value2
 r1                                      column=cf1:c1, timestamp=1637426515563, value=value1
 r1                                      column=cf1:c2, timestamp=1637426521080, value=value4
 r2                                      column=cf1:c3, timestamp=1637426525624, value=value6
 r2                                      column=cf1:c3, timestamp=1637426523363, value=value5
2 row(s)
Took 0.0121 seconds
hbase(main):058:0> deleteall 't6', 'r1', 'cf1:c1'
Took 0.0076 seconds
hbase(main):059:0> scan 't6', VERSIONS => 5
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c2, timestamp=1637426521080, value=value4
 r2                                      column=cf1:c3, timestamp=1637426525624, value=value6
 r2                                      column=cf1:c3, timestamp=1637426523363, value=value5
2 row(s)
Took 0.0061 seconds
```

get

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Getting the whole row
  - Getting the specified column cell
  - Getting multiple columns
  - Using TIMERANGE
  - Getting several value versions of one column
  - Getting the specified value version
  - Using a filter

Definition

Returns the content of the specified row or column cell.

Usage

- Basic syntax:

```
get '[<namespace_name>:]<table_name>',
    '<row_key>'[,
    '<column_family>:<column_qualifier>'[,...]]
```

- Using dictionaries:

```
get '[<namespace_name>:]<table_name>',
    '<row_key>',
    {COLUMN => <columns_array[]>[,
    VERSIONS => <number_displayed_versions>][,
    TIMESTAMP => <cell_timestamp>][,
    TIMERANGE => [<timestamp_start>, <timestamp_end>][,
    ATTRIBUTES => {'<cell_attribute_name>' => '<cell_attribute_value>'[,...]}]}
```

- Using row-scope time ranges:

```
get '[<namespace_name>:]<table_name>',
    '<row_key>',
    {TIMERANGE => [<timestamp_start>, <timestamp_end>]}
```

- Using filters:

```
get '[<namespace_name>:]<table_name>',
    '<row_key>',
    {FILTER => "<filter_description>"}
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
column_family	A column family name
column_qualifier	A column qualifier
columns_array	An array of strings specified in the format `'<column_family>:<column_qualifier>'`, where `<column_family>` — a column family name; `<column_qualifier>` — a column qualifier
number_displayed_versions	A count of the cell versions to be displayed
cell_timestamp	A cell timestamp
timestamp_start	A start value for searching cells timestamps
timestamp_end	A stop value for searching cells timestamps
cell_attribute_name	A cell attribute name
cell_attribute_value	A value of the specified cell attribute
filter_description	A filter used for searching the table cells

Examples

Getting the whole row

```
hbase(main):013:0> get 't4', 'r1'
COLUMN                CELL
 cf1:c1               timestamp=1637324524743, value=value1
 cf1:c2               timestamp=2222221111111, value=value2
 cf1:c3               timestamp=1637326735791, value=value3_new
 cf1:c4               timestamp=1637324859512, value=value4
 cf1:c5               timestamp=1637329193429, value=7738718b
1 row(s)
Took 0.0125 seconds
```

Getting the specified column cell

```
hbase(main):014:0> get 't4', 'r1', 'cf1:c1'
COLUMN                CELL
 cf1:c1               timestamp=1637324524743, value=value1
1 row(s)
Took 0.0239 seconds
```

Getting multiple columns

```
hbase(main):016:0> get 't4', 'r1', {COLUMN => ['cf1:c1', 'cf1:c5']}
COLUMN                CELL
 cf1:c1               timestamp=1637324524743, value=value1
 cf1:c5               timestamp=1637329193429, value=7738718b
1 row(s)
Took 0.0075 seconds
hbase(main):017:0> get 't4', 'r1', 'cf1:c1', 'cf1:c3'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c3                                  timestamp=1637326735791, value=value3_new
1 row(s)
Took 0.0171 seconds
```

Using TIMERANGE

```
hbase(main):019:0> get 't4', 'r1', {TIMERANGE => [1637324524743, 1637326735791]}
COLUMN                CELL
 cf1:c1               timestamp=1637324524743, value=value1
 cf1:c4               timestamp=1637324859512, value=value4
1 row(s)
Took 0.0127 seconds
```

Getting several value versions of one column

```
hbase(main):021:0> get 't4', 'r1', {COLUMN => 'cf1:c5', VERSIONS => 5}
COLUMN                CELL
 cf1:c5               timestamp=1637329193429, value=7738718b
 cf1:c5               timestamp=1637329190124, value=7738718a
 cf1:c5               timestamp=1637329187604, value=7738718W
 cf1:c5               timestamp=1637328326920, value=7738718M
1 row(s)
Took 0.0148 seconds
```

Getting the specified value version

```
hbase(main):022:0> get 't4', 'r1', {COLUMN => 'cf1:c5', TIMESTAMP => 1637329190124}
COLUMN                CELL
 cf1:c5               timestamp=1637329190124, value=7738718a
1 row(s)
Took 0.0146 seconds
```

Using a filter

```
hbase(main):015:0> get 't4', 'r1', {FILTER => "ValueFilter(=, 'regexstring:value*')"}
COLUMN                CELL
 cf1:c1               timestamp=1637324524743, value=value1
 cf1:c2               timestamp=2222221111111, value=value2
 cf1:c3               timestamp=1637326735791, value=value3_new
 cf1:c4               timestamp=1637324859512, value=value4
1 row(s)
Took 0.0973 seconds
```

get_counter

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Returns a counter for the cell value with the specified table/row/column coordinates. A counter should be managed with atomic increment functions (see incr), and the data value should be binary encoded (as long).

Usage

```
get_counter '[<namespace_name>:]<table_name>',
            '<row_key>',
            '<column_family>:<column_qualifier>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
column_family	A column family name
column_qualifier	A column qualifier

Examples

```
hbase(main):049:0> incr 't4', 'r1', 'cf1:c5', 11
COUNTER VALUE = 3978705112342280269
Took 0.0079 seconds
hbase(main):050:0> get 't4','r1','cf1:c5'
COLUMN                                    CELL
 cf1:c5                                   timestamp=1637328326920, value=7738718M
1 row(s)
Took 0.0091 seconds
hbase(main):001:0> get_counter 't4', 'r1', 'cf1:c5'
COUNTER VALUE = 3978705112342280269
Took 0.4473 seconds
```

get_splits

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- A table without splitting
- A table with splitting

Definition

Gets **splits** of the given table — row key values used to define Region boundaries. See more details in the HBase architecture article.

Usage

```
get_splits '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

A table without splitting

```
hbase(main):001:0> get_splits 't4'
Total number of splits = 1
Took 0.4184 seconds
=> []
```

A table with splitting

```
hbase(main):014:0> get_splits 'people_ages'
Total number of splits = 5
F
K
P
W
Took 0.0128 seconds
=> ["F", "K", "P", "W"]
```

incr

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
- Incrementing with the default step 1
- Incrementing with the defined step

Definition

Increments a cell value at the specified table/row/column coordinates. The default incrementing step is `1`, you can also define your own. Additionally, the command allows to modify the custom attributes or visibility of the cell value.

> **CAUTION**
>
> The `incr` command can be applied only to cells containing binary encoded (long) values.

Usage

```
incr '[<namespace_name>:]<table_name>',
     '<row_key>',
     '<column_family>:<column_qualifier>'[,
     '<incr_step>'][,
     {ATTRIBUTES => {'<cell_attribute_name>' => '<cell_attribute_value>'[,...]}}] | [, {VISIBILITY => '<visibility_expression>'}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
column_family	A column family name
column_qualifier	A column qualifier
incr_step	An incrementing step. The default value is `1`
cell_attribute_name	A cell attribute name
cell_attribute_value	A value of the specified cell attribute
visibility_expression	A logical expression that refers to existing visibility labels. Can use the following logical operators: `|` (or), `&` (and). Examples: `SECRET|PRIVATE`, `(SECRET&PRIVATE)|PUBLIC`, `SECRET&PRIVATE&PUBLIC`

Examples

Incrementing with the default step 1

```
hbase(main):043:0> put 't4', 'r1', 'cf1:c5', 77387187
Took 0.0055 seconds
hbase(main):044:0> incr 't4', 'r1', 'cf1:c5'
COUNTER VALUE = 3978705112342280248
Took 0.0074 seconds
hbase(main):045:0> get 't4', 'r1', 'cf1:c5'
COLUMN                                   CELL
 cf1:c5                                  timestamp=1637328241031, value=77387188
1 row(s)
Took 0.0050 seconds
```

Incrementing with the defined step

```
hbase(main):047:0> incr 't4', 'r1', 'cf1:c5', 10
COUNTER VALUE = 3978705112342280258
Took 0.0074 seconds
hbase(main):048:0> get 't4', 'r1', 'cf1:c5'
COLUMN                                   CELL
 cf1:c5                                  timestamp=1637328290099, value=7738718B
1 row(s)
Took 0.0098 seconds
```

put

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Putting a value without timestamp and any attributes
  - Putting a value with the predefined timestamp
  - Putting a value with additional attributes
  - Putting a value with visibility assigning

Definition

Inserts a cell value at the specified table, row, column, and optionally timestamp coordinates.

Usage

```
put '[<namespace_name>:]<table_name>',
    '<row_key>',
    '<column_family>:<column_qualifier>',
    '<cell_value>'[,
    <cell_timestamp>][,
    {ATTRIBUTES => {'<cell_attribute_name>' => '<cell_attribute_value>'[,...]}}][,
    {VISIBILITY => '<visibility_expression>'}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
row_key	A row key
column_family	A column family name
column_qualifier	A column qualifier
cell_value	A value for the inserted cell
cell_timestamp	A cell timestamp
cell_attribute_name	A cell attribute name
cell_attribute_value	A value of the specified cell attribute
visibility_expression	A logical expression that refers to existing visibility labels. Can use the following logical operators: `|` (or), `&` (and). Examples: `SECRET|PRIVATE`, `(SECRET&PRIVATE)|PUBLIC`, `SECRET&PRIVATE&PUBLIC`

Examples

Putting a value without timestamp and any attributes

```
hbase(main):001:0> put 't4', 'r1', 'cf1:c1', 'value1'
Took 0.5673 seconds
hbase(main):002:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
1 row(s)
Took 0.0396 seconds
```

Putting a value with the predefined timestamp

```
hbase(main):009:0> put 't4', 'r1', 'cf1:c2', 'value2', 2222221111111
Took 0.0073 seconds
hbase(main):010:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
1 row(s)
Took 0.0104 seconds
```

Putting a value with additional attributes

```
hbase(main):011:0> put 't4', 'r1', 'cf1:c3', 'value3', {ATTRIBUTES => {'my_prop_name' => 'my_prop_value'}}
Took 0.0123 seconds
hbase(main):012:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
 cf1:c3                                  timestamp=1637324768581, value=value3
1 row(s)
Took 0.0086 seconds
```

Putting a value with visibility assigning

```
hbase(main):013:0> put 't4', 'r1', 'cf1:c4', 'value4', {VISIBILITY => 'SECRET'}
Took 0.0231 seconds
hbase(main):014:0> get 't4', 'r1'
COLUMN                                   CELL
 cf1:c1                                  timestamp=1637324524743, value=value1
 cf1:c2                                  timestamp=2222221111111, value=value2
 cf1:c3                                  timestamp=1637324768581, value=value3
 cf1:c4                                  timestamp=1637324859512, value=value4
1 row(s)
Took 0.0109 seconds
```

scan

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
  - Scanning all the table
  - Scanning the specified column cell
  - Scanning several columns
  - Using TIMERANGE
  - Scanning several value versions of one column
  - Scanning the specified value version
  - Using a filter for column values
  - Using a prefix for searching row keys
  - Getting all statistical metrics
  - Getting the defined statistical metrics
  - Including the cells marked for deleting

Definition

Scans the table and returns all the rows matching the given specifications.

Usage

- Basic syntax:

```
scan '[<namespace_name>:]<table_name>'[,
     '<column_family>:<column_qualifier>']
```

- Using dictionaries:

```
scan '[<namespace_name>:]<table_name>',
     {COLUMN => <columns_array[]>[,
     VERSIONS => <number_displayed_versions>][,
     TIMESTAMP => <cell_timestamp>][,
     TIMERANGE => [<timestamp_start>, <timestamp_end>][,
     ATTRIBUTES => {'<cell_attribute_name>' => '<cell_attribute_value>'[,...]}]}
```

- Using table-scope time ranges:

```
scan '[<namespace_name>:]<table_name>',
     {TIMERANGE => [<timestamp_start>, <timestamp_end>]}
```

- Using filters:

```
scan '[<namespace_name>:]<table_name>',
     {[ROWPREFIXFILTER => '<row_prefix>'][,
     FILTER => "<filter_description>"]}`
```

- Including metrics:

```
scan '[<namespace_name>:]<table_name>',
     {ALL_METRICS => true} | {METRICS => <metrics_array[]>}
```

- Including the cells marked as deleted:

```
scan '[<namespace_name>:]<table_name>',
     {RAW => true, VERSIONS => <number_displayed_versions>}
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
column_family	A column family name
column_qualifier	A column qualifier
columns_array	An array of strings specified in the format `'<column_family>:<column_qualifier>'`, where `<column_family>` — a column family name; `<column_qualifier>` — a column qualifier
number_displayed_versions	A count of the cell versions to be displayed
cell_timestamp	A cell timestamp
timestamp_start	A start value for searching cells timestamps
timestamp_end	A stop value for searching cells timestamps
cell_attribute_name	A cell attribute name
cell_attribute_value	A value of the specified cell attribute
row_prefix	A prefix for searching row keys
filter_description	A filter used for searching the table cells
metrics_array	An array of statistical metrics names

Examples

Scanning all the table

```
hbase(main):003:0> scan 't4'
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
1 row(s)
Took 0.0744 seconds
```

Scanning the specified column cell

```
hbase(main):010:0> scan 't4', {COLUMN => 'cf1:c1'}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
1 row(s)
Took 0.0103 seconds
```

Scanning several columns

```
hbase(main):008:0> scan 't4', {COLUMN => ['cf1:c1', 'cf1:c2']}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
1 row(s)
Took 0.0149 seconds
```

Using TIMERANGE

```
hbase(main):009:0> scan 't4', {TIMERANGE => [1637324524743, 1637326735791]}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
1 row(s)
Took 0.0129 seconds
```

Scanning several value versions of one column

```
hbase(main):011:0> scan 't4', {COLUMN => 'cf1:c5', VERSIONS => 5}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
 r1                                      column=cf1:c5, timestamp=1637329190124, value=7738718a
 r1                                      column=cf1:c5, timestamp=1637329187604, value=7738718W
 r1                                      column=cf1:c5, timestamp=1637328326920, value=7738718M
1 row(s)
Took 0.0085 seconds
```

Scanning the specified value version

```
hbase(main):016:0> scan 't4', {COLUMN => 'cf1:c5', TIMESTAMP => 1637329190124}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c5, timestamp=1637329190124, value=7738718a
1 row(s)
Took 0.0070 seconds
```

Using a filter for column values

```
hbase(main):019:0> scan 't4', {FILTER => "ValueFilter(=, 'regexstring:value*')"}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
1 row(s)
Took 0.0380 seconds
```

Using a prefix for searching row keys

```
hbase(main):026:0> scan 't4', {ROWPREFIXFILTER => 'r'}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
1 row(s)
Took 0.0077 seconds
```

Getting all statistical metrics

```
hbase(main):022:0> scan 't4', {ALL_METRICS => true}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
1 row(s)

METRIC                                   VALUE
 BYTES_IN_REMOTE_RESULTS                 191
 BYTES_IN_RESULTS                        191
 MILLIS_BETWEEN_NEXTS                    5
 NOT_SERVING_REGION_EXCEPTION            0
 REGIONS_SCANNED                         1
 REMOTE_RPC_CALLS                        1
 REMOTE_RPC_RETRIES                      0
 ROWS_FILTERED                           0
 ROWS_SCANNED                            1
 RPC_CALLS                               1
 RPC_RETRIES                             0
Took 0.0079 seconds
```

Getting the defined statistical metrics

```
hbase(main):028:0> scan 't4', {METRICS => ['RPC_RETRIES', 'ROWS_FILTERED']}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
1 row(s)

METRIC                                   VALUE
 ROWS_FILTERED                           0
 RPC_RETRIES                             0
Took 0.0131 seconds
```

Including the cells marked for deleting

```
hbase(main):001:0> scan 't4', {RAW => true, VERSIONS => 10}
ROW                                      COLUMN+CELL
 r1                                      column=cf1:c1, timestamp=1637324524743, value=value1
 r1                                      column=cf1:c2, timestamp=2222221111111, value=value2
 r1                                      column=cf1:c3, timestamp=1637326735791, value=value3_new
 r1                                      column=cf1:c4, timestamp=1637324859512, value=value4
 r1                                      column=cf1:c5, timestamp=1637329193429, value=7738718b
 r1                                      column=cf1:c5, timestamp=1637329190124, value=7738718a
 r1                                      column=cf1:c5, timestamp=1637329187604, value=7738718W
 r1                                      column=cf1:c5, timestamp=1637328326920, value=7738718M
 r1                                      column=cf2:c1, timestamp=1637329303310, type=Delete
 r1                                      column=cf2:c1, timestamp=1637329303310, value=last
 r1                                      column=cf2:c1, timestamp=1637329264711, type=Delete
 r1                                      column=cf2:c1, timestamp=1637329264711, value=value_new_new2
 r1                                      column=cf2:c1, timestamp=1637329262403, type=Delete
 r1                                      column=cf2:c1, timestamp=1637329262403, value=value_new
 r1                                      column=cf2:c1, timestamp=1637329253123, type=Delete
 r1                                      column=cf2:c1, timestamp=1637329253123, value=value
1 row(s)
Took 0.3665 seconds
```

truncate

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Disables, drops, and recreates the specified table.

Usage

```
truncate '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):004:0> truncate 't4'
Truncating 't4' table (it may take a while):
Disabling table...
Truncating table...
Took 2.1331 seconds
```

truncate_preserve

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Disables, drops, and recreates the specified table, maintaining the previous Region boundaries.

Usage

```
truncate_preserve '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):005:0> truncate_preserve 't4'
Truncating 't4' table (it may take a while):
Disabling table...
Truncating table...
Took 2.0099 seconds
```

assign

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Assigns the defined Region to one of Region Servers. Read more details about Regions assignment in the HBase documentation.

CAUTION

If the defined Region is already assigned, this command will force its reassign. Use with caution.

Usage

```
assign '<region_name>' | '<encoded_region_name>'
```

Arguments

Parameter	Description
region_name	A Region name without its encoded part. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the `region_name` is `wordcount2,m,1637833871529`. You can get the full Region name and exclude its encoded part via the list_regions command output (the `REGION_NAME` field)
encoded_region_name	An encoded part of the Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)

Examples

```
hbase(main):007:0> assign '2ff3683a90c4106432c540ddd0f57cba'
Took 1.2233 seconds
hbase(main):008:0> assign 'wordcount2,m,1637833871529'
Took 1.1642 seconds
```

balancer

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Triggers the cluster Balancer. Returns `true` if the Balancer is running and is able to tell Region Servers to unassign all Regions to balance (the re-assignment itself is async). Otherwise returns `false` — in most cases, when some Regions are in transition (the input parameter `force` can be used to force the command in such cases).

> **CAUTION**
>
> For experts only. Forcing a balance may do more damage than repair if Regions assignment is corrupted.

Usage

```
balancer ['force']
```

Arguments

Parameter	Description
force	Use it to force balancing even if some Regions are in transition. By default, the command does not run in such a case

Examples

```
hbase(main):050:0> balancer
true
Took 0.0196 seconds
=> 1
```

balancer_enabled

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Checks the Balancer state. Returns `true` if the Balancer is enabled, otherwise returns `false`.

Usage

```
balancer_enabled
```

Examples

```
hbase(main):009:0> balancer_enabled
true
Took 0.0161 seconds
=> true
```

balance_switch

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Disabling
  - Enabling

Definition

Enables/disables the cluster Balancer. Returns the previous Balancer state.

Usage

```
balance_switch <new_state>
```

Arguments

Parameter	Description
new_state	A boolean value that defines a new Balancer state. Use `true` to enable, `false` to disable

Examples

Disabling

```
hbase(main):047:0> balance_switch false
Previous balancer state : true
Took 0.0249 seconds
=> "true"
```

Enabling

```
hbase(main):048:0> balance_switch true
Previous balancer state : false
Took 0.0089 seconds
=> "false"
```

catalogjanitor_enabled

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Checks the CatalogJanitor state. Returns `true` if the CatalogJanitor is enabled, otherwise returns `false`.

Usage

```
catalogjanitor_enabled
```

Examples

```
hbase(main):038:0> catalogjanitor_enabled
true
Took 0.0112 seconds
=> 1
```

catalogjanitor_run

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Runs the CatalogJanitor in order to check and clean up the hbase:meta table (previously called `.META.`).

Usage

```
catalogjanitor_run
```

Examples

```
hbase(main):043:0> catalogjanitor_run
Took 0.0118 seconds
=> 0
```

catalogjanitor_switch

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Disabling
  - Enabling

Definition

Enables/disables the CatalogJanitor. Returns the previous CatalogJanitor state.

Usage

```
catalogjanitor_switch <new_state>
```

Arguments

Parameter	Description
new_state	A boolean value that defines a new CatalogJanitor state. Use `true` to enable, `false` to disable

Examples

Disabling

```
hbase(main):040:0> catalogjanitor_switch false
true
Took 0.0179 seconds
=> 1
```

Enabling

```
hbase(main):041:0> catalogjanitor_switch true
false
Took 0.0033 seconds
=> 1
```

cleaner_chore_enabled

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Checks the Cleaner chore state. Returns `true`, if the Cleaner chore is enabled, otherwise returns `false`.

Usage

```
cleaner_chore_enabled
```

Examples

```
hbase(main):008:0> cleaner_chore_enabled
true
Took 0.0081 seconds
=> 1
```

cleaner_chore_run

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Runs garbage collecting of HFiles and WAL files via the Cleaner chore.

Usage

```
cleaner_chore_run
```

Examples

```
hbase(main):002:0> cleaner_chore_run
Took 0.3230 seconds
=> false
```

cleaner_chore_switch

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Disabling
  - Enabling

Definition

Enables/disables the Cleaner chore. Returns the previous Cleaner chore state.

Usage

```
cleaner_chore_switch <new_state>
```

Arguments

Parameter	Description
new_state	A boolean value that defines a new Cleaner chore state. Use `true` to enable, `false` to disable

Examples

Disabling

```
hbase(main):010:0> cleaner_chore_switch false
true
Took 0.0123 seconds
=> 1
```

Enabling

```
hbase(main):011:0> cleaner_chore_switch true
false
Took 0.0034 seconds
=> 1
```

clear_block_cache

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Removes all blocks corresponding to the specified table from BlockCache.

CAUTION

This command can significantly impact the query performance, as the subsequent queries will have to retrieve the blocks from the underlying filesystem. Use with caution.

Usage

```
clear_block_cache '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):006:0> clear_block_cache 'ns1:temp2'
CacheEvictionStats{evictedBlocks=0, maxCacheSize=1638085504, failedRegionsSize=0, failedRegions=[]}
Took 0.1503 seconds
=> 1
```

clear_compaction_queues

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Clears compaction queues at the specified Region Server.

Usage

```
clear_compaction_queues '<region_server_name>,<port_number>'[, <queue_names_array[]>]
```

Arguments

Parameter	Description
region_server_name	A Region Server name
port_number	A port number
queue_names_array	An array with possible string values: ▪ `long` — longCompactions queue (Large Compaction Thread); ▪ `short` — shortCompactions queue (Small Compaction Thread).

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

```
hbase(main):002:0> clear_compaction_queues 'bds-adh-1.ru-central1.internal,16020'
Took 0.0233 seconds
hbase(main):003:0> clear_compaction_queues 'bds-adh-1.ru-central1.internal,16020', ['long']
Took 0.0047 seconds
```

clear_deadservers

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Clears all dead Region Servers from HBase or the specified ones. The optional input parameter can be used for filtering dead servers. Returns an error if the list of dead servers is empty.

Usage

```
clear_deadservers ['<region_server_name>,<port_number>,<start_code>'[,...]]
```

Arguments

Parameter	Description
region_server_name	A Region Server name
port_number	A port number
start_code	A Server start code

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

```
hbase(main):008:0> clear_deadservers

ERROR: servers cannot be null or empty

          Clear the dead region servers that are never used.
          Examples:
          Clear all dead region servers:
          hbase> clear_deadservers
          Clear the specified dead region servers:
          hbase> clear_deadservers 'host187.example.com,60020,1289493121758'
          or
          hbase> clear_deadservers 'host187.example.com,60020,1289493121758',
                                   'host188.example.com,60020,1289493121758'

Took 0.0166 seconds
```

close_region

Daria Barysheva

To Table of Contents

IMPORTANT

The command is deprecated. Please, use unassign instead.

compact

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Table
  - Column family (within a table)
  - Region
  - Column family (within a Region)

Definition

Runs compaction for the defined table/Region/column family. See also major_compact.

Usage

```
compact '[<namespace_name>:]<table_name>' | '<region_name>' | '<encoded_region_name>'[,
        '<column_family>' | nil[,
        '<compact_type>']]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
region_name	A Region name without its encoded part. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the `region_name` is `wordcount2,m,1637833871529`. You can get the full Region name and exclude its encoded part via the list_regions command output (the `REGION_NAME` field)
encoded_region_name	An encoded part of the Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)
column_family	A column family name
compact_type	Possible values: - `NORMAL` — normal compaction; - `MOB` — MOB (medium-sized objects) compaction. The default value is `NORMAL`. See more details in the HBase documentation

Examples

Table

```
hbase(main):023:0> compact 'ns1:temp2'
Took 0.0483 seconds
```

Column family (within a table)

```
hbase(main):026:0> compact 'ns1:temp2', 'cf1'
Took 0.0252 seconds
```

Region

```
hbase(main):027:0> compact 'wordcount2,m,1637833871529'
Took 0.2334 seconds
hbase(main):028:0> compact '2ff3683a90c4106432c540ddd0f57cba'
Took 0.0151 seconds
```

Column family (within a Region)

```
hbase(main):030:0> compact '2ff3683a90c4106432c540ddd0f57cba', 'f'
Took 0.0121 seconds
```

compaction_state

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Gets compaction status for the defined table.

Usage

```
compaction_state '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):002:0> compaction_state 'ns1:temp2'
Took 0.1477 seconds
=> "NONE"
```

compact_rs

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Compacts all Regions within the defined Region Server.

Usage

```
compact_rs '<region_server_name>,<port_number>[,<start_code>]'[, <is_major>]
```

Arguments

Parameter	Description
region_server_name	A Region Server name
port_number	A port number
start_code	A Server start code
is_major	Use `true` to run Major compaction

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

```
hbase(main):003:0> compact_rs 'bds-adh-1.ru-central1.internal,16020,1642487399069'
Took 0.1098 seconds
hbase(main):004:0> compact_rs 'bds-adh-1.ru-central1.internal,16020,1642487399069', true
Took 0.0795 seconds
```

flush

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Table
  - Region Server
  - Region name
  - Encoded Region name

Definition

Flushes all Regions for the defined table/Region Server or all Regions with the specified name. Flushing means moving information from MemStore to HFiles. By default, this process is performed automatically when the MemStore fills up.

Usage

```
flush '[<namespace_name>:]<table_name>' | '<region_server_name>,<port_number>,<start_code>' | '<region_name>' |
'<encoded_region_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
region_server_name	A Region Server name
port_number	A port number
start_code	A Server start code
region_name	A Region name without its encoded part. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the `region_name` is `wordcount2,m,1637833871529`. You can get the full Region name and exclude its encoded part via the list_regions command output (the `REGION_NAME` field)
encoded_region_name	An encoded part of the Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

Table

```
hbase(main):019:0> flush 'BEST_BOOKS'
Took 0.2079 seconds
```

Region Server

```
hbase(main):020:0> flush 'bds-adh-2.ru-central1.internal,16020,1642400781522'
Took 0.1736 seconds
```

Region name

```
hbase(main):021:0> flush 'BEST_BOOKS,K,1642431463389'
Took 0.0034 seconds
```

Encoded Region name

```
hbase(main):022:0> flush 'd6e6be2ad047bc3f2ea5ed6842f851b0'
Took 0.0076 seconds
```

is_in_maintenance_mode

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Checks whether the Master Server is in maintenance mode.

Usage

```
is_in_maintenance_mode
```

Examples

```
hbase(main):004:0> is_in_maintenance_mode
false
Took 0.3552 seconds
=> false
```

list_deadservers

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Lists all dead Region Servers.

Usage

```
list_deadservers
```

Examples

```
hbase(main):006:0> list_deadservers
SERVERNAME
0 row(s)
Took 0.0316 seconds
```

major_compact

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Table
  - Column family (within a table)
  - Region
  - Column family (within a Region)

Definition

Runs Major compaction for the defined table/Region/column family. See also compact.

Usage

```
major_compact '[<namespace_name>:]<table_name>' | '<region_name>' | '<encoded_region_name>'[,
              '<column_family>' | nil[,
              '<compact_type>']]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
region_name	A Region name without its encoded part. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the `region_name` is `wordcount2,m,1637833871529`. You can get the full Region name and exclude its encoded part via the list_regions command output (the `REGION_NAME` field)
encoded_region_name	An encoded part of the Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)
column_family	A column family name
compact_type	Possible values: ■ `NORMAL` — normal compaction; ■ `MOB` — MOB (medium-sized objects) compaction. The default value is `NORMAL`. See more details in the HBase documentation

Examples

Table

```
hbase(main):031:0> major_compact 'ns1:temp2'
Took 0.0349 seconds
```

Column family (within a table)

```
hbase(main):032:0> major_compact 'ns1:temp2', 'cf1'
Took 0.0175 seconds
```

Region

```
hbase(main):033:0> major_compact 'wordcount2,m,1637833871529'
Took 0.0179 seconds
hbase(main):034:0> major_compact '2ff3683a90c4106432c540ddd0f57cba'
Took 0.0105 seconds
```

Column family (within a Region)

```
hbase(main):035:0> major_compact '2ff3683a90c4106432c540ddd0f57cba', 'f'
Took 0.0125 seconds
```

merge_region

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Merges two Regions. Takes full or encoded Region names as input.

Usage

```
merge_region '<full_region_name1>' | '<encoded_region_name1>',
             '<full_region_name2>' | '<encoded_region_name2>'[,
             '<force_merge>']
```

Arguments

Parameter	Description
full_region_name1	A full name of the first Region. Can be obtained via the list_regions command output (the `REGION_NAME` field)
encoded_region_name1	An encoded part of the first Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)
full_region_name2	A full name of the second Region
encoded_region_name2	An encoded part of the second Region name
force_merge	Use `true` to force the merge process, even if not adjacent Regions are passed (for experts only)

Examples

```
hbase(main):001:0> list_regions 'BEST_BOOKS'
                                        SERVER_NAME |                                                  REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ------------------------------------------------------------ | ---------- |
---------- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1642400781522 |  BEST_BOOKS,,1642431463389.dc0dd6bee8c1b1d7db462a17fbd33673. |            |
K |     0 |     0 |        1.0 |
 bds-adh-2.ru-central1.internal,16020,1642400781522 | BEST_BOOKS,K,1642431463389.d6e6be2ad047bc3f2ea5ed6842f851b0. |          K |
|     0 |     0 |        1.0 |
 2 rows
Took 0.5506 seconds
hbase(main):002:0> merge_region 'BEST_BOOKS,,1642431463389.dc0dd6bee8c1b1d7db462a17fbd33673.', 'BEST_BOOKS,K,1642431463389.
d6e6be2ad047bc3f2ea5ed6842f851b0.'
Took 0.1575 seconds
hbase(main):004:0> list_regions 'BEST_BOOKS'
                                        SERVER_NAME |                                                 REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ----------------------------------------------------------- | ---------- |
---------- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1642400781522 | BEST_BOOKS,,1642432455266.182c4a28649924be67aba3eeb9d15a04. |            |
|     0 |     0 |        1.0 |
 1 rows
Took 0.0185 seconds
```

move

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Moves the specified Region. If the target Region Server is not defined, it is selected randomly.

TIP

You can check the new location of the specified Region via the locate_region command. Notice, that it may take some time for the changes to be returned by this command.

Usage

```
move '<encoded_region_name>'[, '<region_server_name>,<port_number>,<start_code>']
```

Arguments

Parameter	Description
encoded_region_name	An encoded part of the Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)
region_server_name	A Region Server name
port_number	A port number
start_code	A Server start code

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

```
hbase(main):025:0> locate_region 'wordcount2','m'
HOST                                     REGION
 bds-adh-1.ru-central1.internal:16020    {ENCODED => 2ff3683a90c4106432c540ddd0f57cba, NAME => 'wordcount2,m,1637833871529.
2ff3683a90c4106432c540ddd0f57cba.'
                                         , STARTKEY => 'm', ENDKEY => 'r'}
1 row(s)
Took 0.0011 seconds
=> #<Java::OrgApacheHadoopHbase::HRegionLocation:0x403f7aa2>
hbase(main):026:0> move '2ff3683a90c4106432c540ddd0f57cba', 'bds-adh-3.ru-central1.internal,16020,1642400781435'
Took 1.1472 seconds
hbase(main):030:0> locate_region 'wordcount2','m'
HOST                                     REGION
 bds-adh-3.ru-central1.internal:16020    {ENCODED => 2ff3683a90c4106432c540ddd0f57cba, NAME => 'wordcount2,m,1637833871529.
2ff3683a90c4106432c540ddd0f57cba.'
                                         , STARTKEY => 'm', ENDKEY => 'r'}
1 row(s)
Took 0.0056 seconds
=> #<Java::OrgApacheHadoopHbase::HRegionLocation:0x42b43a6>
```

normalize

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Triggers the Region Normalizer for all tables that have the `NORMALIZATION_ENABLED` flag set. Returns `true` if the Normalizer ran successfully, otherwise returns `false`.

IMPORTANT

This command has no effect if Region Normalizer is disabled. Make sure, that the Region Normalizer is turned on using the normalizer_switch command.

Usage

```
normalize
```

Examples

```
hbase(main):036:0> normalize
true
Took 0.0198 seconds
=> 1
```

normalizer_enabled

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Checks the Region Normalizer state. Returns `true` if the Region Normalizer is enabled, otherwise returns `false`.

Usage

```
normalizer_enabled
```

Examples

```
hbase(main):029:0> normalizer_enabled
true
Took 0.0177 seconds
=> 1
```

normalizer_switch

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Disabling
  - Enabling

Definition

Enables/disables the Region Normalizer. Returns the previous Normalizer state.

Usage

```
normalizer_switch <new_state>
```

Arguments

Parameter	Description
new_state	A boolean value that defines a new Region Normalizer state. Use `true` to enable, `false` to disable

Examples

Disabling

```
hbase(main):033:0> normalizer_switch false
true
Took 0.0098 seconds
=> 1
```

Enabling

```
hbase(main):034:0> normalizer_switch true
false
Took 0.0060 seconds
=> 1
```

split

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Splitting a Region
  - Splitting a table

Definition

Splits the entire table or the specified Region into two new Regions.

Usage

```
split '[<namespace_name>:]<table_name>' | '<region_name>'[,
      '<split_key>']
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
region_name	A Region name without its encoded part. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the `region_name` is `wordcount2,m,1637833871529`. You can get the full Region name and exclude its encoded part via the list_regions command output (the `REGION_NAME` field)
split_key	The row key value for dividing the table into two parts

Examples

Splitting a Region

```
hbase(main):005:0> list_regions 'people_ages'
                                        SERVER_NAME |                                                   REGION_NAME |  START_KEY
|    END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ------------------------------------------------------------- | ----------
| ---------- | ----- | ----- | ---------- |
 bds-adh-3.ru-central1.internal,16020,1642400781435 |  people_ages,,1638455640269.6375fc55e8ad8f9948b22cfcc34a4563. |            |
F |     0 |   238 |        1.0 |
 bds-adh-3.ru-central1.internal,16020,1642400781435 | people_ages,F,1638455640269.508ad2a04ec106517eaa8ec6fd01fe83. |          F |
K |     0 |   218 |        1.0 |
 bds-adh-1.ru-central1.internal,16020,1642400781296 | people_ages,K,1638455640269.42f95f9f14c8345d2c9940ea68fc7c30. |          K |
P |     0 |   195 |        1.0 |
 bds-adh-1.ru-central1.internal,16020,1642400781296 | people_ages,P,1638455640269.edf909e0f6785817b4517f044a3ebfd4. |          P |
W |     0 |   254 |        1.0 |
 bds-adh-2.ru-central1.internal,16020,1642400781522 | people_ages,W,1638455640269.bc245646e72b0dcf1a06ca061ed538f4. |          W |
|     0 |    92 |        1.0 |
 5 rows
Took 0.0332 seconds
hbase(main):006:0> split 'people_ages,P,1638455640269', 'S'
Took 0.2193 seconds
hbase(main):007:0> list_regions 'people_ages'
                                        SERVER_NAME |                                                   REGION_NAME |  START_KEY
|    END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ------------------------------------------------------------- | ----------
| ---------- | ----- | ----- | ---------- |
 bds-adh-3.ru-central1.internal,16020,1642400781435 |  people_ages,,1638455640269.6375fc55e8ad8f9948b22cfcc34a4563. |            |
F |     0 |   238 |        1.0 |
 bds-adh-3.ru-central1.internal,16020,1642400781435 | people_ages,F,1638455640269.508ad2a04ec106517eaa8ec6fd01fe83. |          F |
K |     0 |   218 |        1.0 |
 bds-adh-1.ru-central1.internal,16020,1642400781296 | people_ages,K,1638455640269.42f95f9f14c8345d2c9940ea68fc7c30. |          K |
P |     0 |   195 |        1.0 |
 bds-adh-1.ru-central1.internal,16020,1642400781296 | people_ages,P,1642431183346.3636b23eae41ba162fde908cb245d325. |          P |
S |     0 |     0 |        1.0 |
 bds-adh-1.ru-central1.internal,16020,1642400781296 | people_ages,S,1642431183346.f3a24f3f04bd9d126d8f30fe40e466a9. |          S |
W |     0 |     0 |        1.0 |
 bds-adh-2.ru-central1.internal,16020,1642400781522 | people_ages,W,1638455640269.bc245646e72b0dcf1a06ca061ed538f4. |          W |
|     0 |    92 |        1.0 |
 6 rows
Took 0.0355 seconds
```

Splitting a table

```
hbase(main):013:0> list_regions 'BEST_BOOKS'
                                        SERVER_NAME |                                                 REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ----------------------------------------------------------- | ---------- |
---------- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1642400781522 | BEST_BOOKS,,1638430003108.36ee4f69dc1ab9d4a075091ff9ea3b80. |            |
|     0 |     0 |        1.0 |
 1 rows
Took 0.0159 seconds
hbase(main):016:0> split 'BEST_BOOKS', 'K'
Took 0.1780 seconds
hbase(main):017:0> list_regions 'BEST_BOOKS'
                                        SERVER_NAME |                                                  REGION_NAME |  START_KEY |
END_KEY |  SIZE |   REQ |   LOCALITY |
 -------------------------------------------------- | ------------------------------------------------------------ | ---------- |
---------- | ----- | ----- | ---------- |
 bds-adh-2.ru-central1.internal,16020,1642400781522 |  BEST_BOOKS,,1642431463389.dc0dd6bee8c1b1d7db462a17fbd33673. |            |
K |     0 |     0 |        1.0 |
 bds-adh-2.ru-central1.internal,16020,1642400781522 | BEST_BOOKS,K,1642431463389.d6e6be2ad047bc3f2ea5ed6842f851b0. |          K |
|     0 |     0 |        1.0 |
 2 rows
Took 0.0164 seconds
```

splitormerge_enabled

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Checks whether `split` or `merge` operations are enabled.

Usage

```
splitormerge_enabled '<operation_type>'
```

Arguments

Parameter	Description
operation_type	Possible values: - `SPLIT` — checks whether split operations are enabled; - `MERGE` — checks whether merge operations are enabled.

Examples

```
hbase(main):001:0> splitormerge_enabled 'SPLIT'
true
Took 0.3520 seconds
=> 1
hbase(main):002:0> splitormerge_enabled 'MERGE'
true
Took 0.0046 seconds
=> 1
```

splitormerge_switch

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Disabling
  - Enabling

Definition

Enables/disables `split` or `merge` operations. Returns the previous state.

Usage

```
splitormerge_switch '<operation_type>', <new_state>
```

Arguments

Parameter	Description
operation_type	Possible values: ▪ `SPLIT` — split operations; ▪ `MERGE` — merge operations.
new_state	A boolean value that defines a new state for split or merge operations. Use `true` to enable, `false` to disable

Examples

Disabling

```
hbase(main):012:0> splitormerge_switch 'SPLIT', false
true
Took 0.0080 seconds
=> 1
hbase(main):013:0> splitormerge_switch 'MERGE', false
true
Took 0.0080 seconds
=> 1
```

Enabling

```
hbase(main):014:0> splitormerge_switch 'SPLIT', true
false
Took 0.0085 seconds
=> 1
hbase(main):015:0> splitormerge_switch 'MERGE', true
false
Took 0.0087 seconds
=> 1
```

trace

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Start tracing
  - Stop tracing
  - Using a span name

Definition

Starts/stops/checks tracing via HTrace. The kind of operation is defined by the first argument. Returns `true` if tracing is running, otherwise returns `false`.

Usage

```
trace '<trace_type>'[, '<trace_span_name>']
```

Arguments

Parameter	Description
trace_type	Possible values: - `start` — starts tracing. Returns `true` in the case of success. - `stop` — stops the current running tracing span. Returns `false` in the case of success. - `status` — shows whether or not tracing is running.
trace_span_name	An optional name for a new tracing span, which can be defined when starting tracing. The default name is `HBaseShell`

Examples

Start tracing

```
hbase(main):006:0> trace 'status'
Took 0.0004 seconds
=> false
hbase(main):007:0> trace 'start'
Took 0.1438 seconds
=> true
hbase(main):008:0> trace 'status'
Took 0.0002 seconds
=> true
```

Stop tracing

```
hbase(main):009:0> trace 'stop'
Took 0.0003 seconds
=> false
hbase(main):010:0> trace 'status'
Took 0.0001 seconds
=> false
```

Using a span name

```
hbase(main):024:0> trace 'start', 'my'
Took 0.0003 seconds
=> true
hbase(main):025:0> trace 'status', 'my'
Took 0.0002 seconds
=> true
hbase(main):026:0> trace 'stop', 'my'
Took 0.0002 seconds
=> false
hbase(main):027:0> trace 'status', 'my'
Took 0.0002 seconds
=> false
```

unassign

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Unassigns the defined Region. The command closes the Region in its current location and then reopens it again. Can be applied only to assigned Regions.

CAUTION

This command is for expert use only. In the case of double assignments, use `hbck -fix` to resolve conflicts.

Usage

```
unassign '<region_name>' | '<encoded_region_name>'[,
         <force_unassign>]
```

Arguments

Parameter	Description
region_name	A Region name without its encoded part. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the `region_name` is `wordcount2,m,1637833871529`. You can get the full Region name and exclude its encoded part via the list_regions command output (the `REGION_NAME` field)
encoded_region_name	An encoded part of the Region name. For example, if the full Region name is `wordcount2,m,1637833871529.2ff3683a90c4106432c540ddd0f57cba.`, then the encoded part is `2ff3683a90c4106432c540ddd0f57cba`. You can get the full Region name and obtain its encoded part via the list_regions command output (the `REGION_NAME` field)
force_unassign	Use `true` to force unassignment, i.e. to clear all in-memory states in Master before reassign (for experts only)

Examples

```
hbase(main):011:0> unassign 'wordcount2,m,1637833871529'
Took 1.1734 seconds
hbase(main):013:0> assign 'wordcount2,m,1637833871529'
Took 1.1425 seconds
hbase(main):014:0> unassign '2ff3683a90c4106432c540ddd0f57cba'
Took 1.1506 seconds
hbase(main):015:0> assign '2ff3683a90c4106432c540ddd0f57cba'
Took 1.1368 seconds
```

wal_roll

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Rolls the WAL writer, i.e. starts writing log messages into a new file. Needs the Region Server name.

Usage

```
wal_roll '<region_server_name>,<port_number>,<start_code>'
```

Arguments

Parameter	Description
region_server_name	A Region Server name
port_number	A port number
start_code	A Server start code

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

```
hbase(main):004:0> wal_roll 'bds-adh-1.ru-central1.internal,16020,1642400781296'
Took 0.0119 seconds
```

zk_dump

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Shows the dump status of the HBase cluster, as seen by ZooKeeper.

Usage

```
zk_dump
```

Examples

```
hbase(main):001:0> zk_dump
HBase is rooted at /hbase
Active master address: bds-adh-2.ru-central1.internal,16000,1642400778136
Backup master addresses:
Region server holding hbase:meta: bds-adh-2.ru-central1.internal,16020,164240078
1522
Region servers:
 bds-adh-3.ru-central1.internal,16020,1642400781435
 bds-adh-2.ru-central1.internal,16020,1642400781522
 bds-adh-1.ru-central1.internal,16020,1642400781296
/hbase/replication:
/hbase/replication/peers:
/hbase/replication/rs:
/hbase/replication/rs/bds-adh-1.ru-central1.internal,16020,1642400781296:
/hbase/replication/rs/bds-adh-2.ru-central1.internal,16020,1642400781522:
/hbase/replication/rs/bds-adh-3.ru-central1.internal,16020,1642400781435:
Quorum Server Statistics:
 bds-adh-1.ru-central1.internal:2181
  Zookeeper version: 3.4.14-f42ef4b166f1e5c63117c75aed651589a0cce1b2, built on 0
8/06/2019 14:49 GMT
  Clients:
   /10.92.6.9:54898[1](queued=0,recved=655,sent=677)
   /10.92.6.217:50280[0](queued=0,recved=1,sent=0)
   /10.92.6.217:43832[1](queued=0,recved=687,sent=710)

  Latency min/avg/max: 0/0/21
  Received: 1531
  Sent: 1575
  Connections: 3
  Outstanding: 0
  Zxid: 0x2600000118
  Mode: follower
  Node count: 283
 bds-adh-2.ru-central1.internal:2181
  Zookeeper version: 3.4.14-f42ef4b166f1e5c63117c75aed651589a0cce1b2, built on 0
8/06/2019 14:49 GMT
  Clients:
   /10.92.6.217:39292[0](queued=0,recved=1,sent=0)
   /10.92.6.217:60694[1](queued=0,recved=2258,sent=2258)
   /10.92.6.217:39256[1](queued=0,recved=2,sent=2)
   /10.92.6.217:39288[1](queued=0,recved=16,sent=16)
   /10.92.6.9:59798[1](queued=0,recved=1581,sent=1619)
   /10.92.6.90:42248[1](queued=0,recved=651,sent=671)

  Latency min/avg/max: 0/0/17
  Received: 4680
  Sent: 4737
  Connections: 6
  Outstanding: 0
  Zxid: 0x2600000118
  Mode: follower
  Node count: 283
 bds-adh-3.ru-central1.internal:2181
  Zookeeper version: 3.4.14-f42ef4b166f1e5c63117c75aed651589a0cce1b2, built on 0
8/06/2019 14:49 GMT
  Clients:
   /10.92.6.9:53374[1](queued=0,recved=632,sent=632)
   /10.92.6.90:53890[1](queued=0,recved=2260,sent=2260)
   /10.92.6.217:40910[0](queued=0,recved=1,sent=0)

  Latency min/avg/max: 0/0/12
  Received: 3060
  Sent: 3059
  Connections: 3
  Outstanding: 0
  Zxid: 0x2600000118
  Mode: leader
  Node count: 283
  Proposal sizes last/min/max: 36/32/842
Took 0.1005 seconds
```

add_peer

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
  HBase clusters peer
  Custom replication endpoint

Definition

Adds a new replication relationship (peer) between the current HBase cluster and either another HBase cluster or a custom replication endpoint, based on the predefined class. Remember the following features and restrictions:

- To configure replication between two clusters, describe the second cluster using the input parameter `CLUSTER_KEY`. Otherwise use the parameter `ENDPOINT_CLASSNAME` to define the custom replication endpoint. Either `CLUSTER_KEY` or `ENDPOINT_CLASSNAME` should be specified. If `ENDPOINT_CLASSNAME` is defined, then `CLUSTER_KEY` is optional and should be specified only if a particular custom replication endpoint requires it.
- By default, only the selected namespaces and tables are replicated. They should be defined by the `NAMESPACES` and `TABLE_CFS` parameters. But you can configure replication for all tables by setting the `REPLICATE_ALL` flag via the set_peer_replicate_all command.
- Regardless of the `REPLICATE_ALL` flag value, only the tables with the set flag `REPLICATION_SCOPE` are replicated. To set this flag and make a table replicable, use the enable_table_replication command. To reset the flag and disable replication for a table, use the disable_table_replication command. Notice, that replication process can't start for tables with disabled replication, even if these tables are selected in the peer configuration.
- Input parameters `NAMESPACES` and `TABLE_CFS` are related. You can't set tables from the namespaces that are already defined by the `NAMESPACES` parameter.
- You can set replication peer namespaces later via the append_peer_namespaces or set_peer_namespaces commands. To remove namespaces, run the remove_peer_namespaces command.
- You can set replication peer tables/column families later via the append_peer_tableCFs or set_peer_tableCFs commands. To remove tables/column families, run the remove_peer_tableCFs command.

NOTE

For more details on configuring replication, refer to the HBase documentation.

Usage

- HBase clusters peer:

```
add_peer '<peer_id>',
         CLUSTER_KEY => '<cluster_key>'[,
         STATE => '<state>'][,
         NAMESPACES => <namespaces_array[]>][,
         TABLE_CFS => {'[<namespace_name>:]<table_name>' => <column_families_array[]>[,...]}]
```

- Custom replication endpoint:

```
add_peer '<peer_id>',
         ENDPOINT_CLASSNAME => '<class_name>'[,
         CLUSTER_KEY => '<cluster_key>'][,
         STATE => '<state>'][,
         DATA => {'<key_name>' => <key_value>, [...]}][,
         CONFIG => {'<config_name>' => '<config_value>', [...]}][,
         NAMESPACES => <namespaces_array[]>][,
         TABLE_CFS => {'[<namespace_name>:]<table_name>' => <column_families_array[]>[,...]}]
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
cluster_key	A full address of the second HBase cluster in the replication relationship. Mandatory in case of replication being configured between two clusters. Takes the following format: `<hbase.zookeeper.quorum>:<hbase.zookeeper.property.clientPort>:<zookeeper.znode.parent>`. All arguments can be obtained via Master UI (page ***HBase configuration***)
state	A replication peer state. Possible values are `ENABLED` and `DISABLED`. The default value is `ENABLED`. Can be changed later via the disable_peer and enable_peer commands
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name
namespace_name	A namespace name
table_name	A table name
column_families_array	An array of strings specified in the format `'<column_family>'`, where `<column_family>` — a column family name
class_name	A name of the custom class. Mandatory in case of replication being configured for a custom replication endpoint. Example: `org.apache.hadoop.hbase.MyReplicationEndpoint`
key_name	A name of the peer data parameter
key_value	A value of the peer data parameter
config_name	A name of the peer configuration parameter
config_value	A value of the peer configuration parameter

Examples

HBase clusters peer

```
hbase(main):001:0> add_peer '1', CLUSTER_KEY => "serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase", TABLE_CFS => { "articles" => []}
Took 0.3667 seconds
```

Custom replication endpoint

```
hbase(main):018:0> add_peer '2', ENDPOINT_CLASSNAME => 'test', STATE => 'DISABLED', DATA => {'key1' => 'value1'}, CONFIG => {'config1' => 'value1'}, NAMESPACES => ['ns1'], TABLE_CFS => { "articles" => []}
Took 0.0306 seconds
```

append_peer_namespaces

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Successful namespace appending
  - Unsuccessful namespace appending

Definition

Appends replicable namespaces to the specified replication relationship (peer). Remember the following restrictions:

- The `REPLICATE_ALL` flag should have the `false` value in the replication peer configuration (see set_peer_replicate_all).
- Setting a namespace means that all tables in this namespace with the `REPLICATION_SCOPE` flag != 0 are replicated. If the peer configuration already contains some table (added via append_peer_tableCFs or set_peer_tableCFs), then setting its namespace with this command is not allowed.

NOTE

This command does not remove the previously defined namespaces from the peer configuration (unlike set_peer_namespaces).

Usage

```
append_peer_namespaces '<peer_id>', <namespaces_array[]>
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name

Examples

Successful namespace appending

A peer without namespaces:

```
hbase(main):019:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
default.articles 0
1 row(s)
Took 0.0119 seconds
=> #<Java::JavaUtil::ArrayList:0xe886caf>
hbase(main):022:0> append_peer_namespaces '1', ['ns1']
Took 0.0250 seconds
hbase(main):025:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1 default.articles 0
1 row(s)
Took 0.0063 seconds
=> #<Java::JavaUtil::ArrayList:0x6a902015>
```

A peer with namespaces:

```
hbase(main):031:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1 default.articles 0
1 row(s)
Took 0.0063 seconds
=> #<Java::JavaUtil::ArrayList:0x6a902015>
hbase(main):032:0> append_peer_namespaces '1', ['ns2']
Took 0.0114 seconds
hbase(main):034:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1;ns2 default.articles 0
1 row(s)
Took 0.0084 seconds
=> #<Java::JavaUtil::ArrayList:0x2c1ea7be>
```

Unsuccessful namespace appending

```
hbase(main):053:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED false
ns1 default.articles 0
1 row(s)
Took 0.0113 seconds
=> #<Java::JavaUtil::ArrayList:0x6dde1bf5>
hbase(main):054:0> append_peer_namespaces '1', ['default']

ERROR: java.io.IOException: Table-cfs articles is conflict with namespaces default in peer config
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:472)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)
Caused by: org.apache.hadoop.hbase.replication.ReplicationException: Table-cfs articles is conflict with namespaces default in
peer config
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.
checkNamespacesAndTableCfsConfigConflict(ReplicationManager.java:181)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.checkPeerConfig(ReplicationManager.java:148)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.updatePeerConfig(ReplicationManager.java:103)
        at org.apache.hadoop.hbase.master.HMaster.updateReplicationPeerConfig(HMaster.java:3535)
        at org.apache.hadoop.hbase.master.MasterRpcServices.updateReplicationPeerConfig(MasterRpcServices.java:1943)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        ... 3 more

  Append some namespaces to be replicable for the specified peer.

  Set a namespace in the peer config means that all tables in this
  namespace (with replication_scope != 0 ) will be replicated.

  Examples:

    # append ns1,ns2 to be replicable for peer '2'.
    hbase> append_peer_namespaces '2', ["ns1", "ns2"]


Took 8.1496 seconds
```

append_peer_tableCFs

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Successful table appending
  - Unsuccessful table appending

Definition

Appends replicable tables/column families to the specified replication relationship (peer). Remember the following restrictions:

- The `REPLICATE_ALL` flag should have the `false` value in the replication peer configuration (see set_peer_replicate_all).
- If the peer configuration already contains some namespace (added via append_peer_namespaces or set_peer_namespaces), then setting its tables with this command is not allowed.

NOTE

This command does not remove the previously defined tables/column families from the replication peer configuration (unlike set_peer_tableCFs).

Usage

```
append_peer_tableCFs '<peer_id>',
                     {'[<namespace_name>:]<table_name>' => <column_families_array[]>[,...]}
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespace_name	A namespace name
table_name	A table name
column_families_array	An array of strings specified in the format `'<column_family>'`, where `<column_family>` — a column family name

Examples

Successful table appending

Without column families

```
hbase(main):037:0> show_peer_tableCFs '1'
default.articles
Took 0.0077 seconds
=> "default.articles"
hbase(main):042:0> append_peer_tableCFs '1', {'table10' => []}
Took 0.0079 seconds
hbase(main):043:0> show_peer_tableCFs '1'
default.articles;default.table10
Took 0.0033 seconds
=> "default.articles;default.table10"
```

With column families

```
hbase(main):045:0> append_peer_tableCFs '1', {'temp1' => ['cf1']}
Took 0.0087 seconds
hbase(main):046:0> show_peer_tableCFs '1'
default.articles;default.table10;default.temp1:cf1
Took 0.0036 seconds
=> "default.articles;default.table10;default.temp1:cf1"
```

Unsuccessful table appending

```
hbase(main):057:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED false
ns1 default.articles 0
1 row(s)
Took 0.0112 seconds
=> #<Java::JavaUtil::ArrayList:0x1ba7db2a>
hbase(main):058:0> append_peer_tableCFs '1', {'ns1:temp2' => []}

ERROR: java.io.IOException: Table-cfs ns1:temp2 is conflict with namespaces ns1 in peer config
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:472)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)
Caused by: org.apache.hadoop.hbase.replication.ReplicationException: Table-cfs ns1:temp2 is conflict with namespaces ns1 in peer
config
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.
checkNamespacesAndTableCfsConfigConflict(ReplicationManager.java:181)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.checkPeerConfig(ReplicationManager.java:148)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.updatePeerConfig(ReplicationManager.java:103)
        at org.apache.hadoop.hbase.master.HMaster.updateReplicationPeerConfig(HMaster.java:3535)
        at org.apache.hadoop.hbase.master.MasterRpcServices.updateReplicationPeerConfig(MasterRpcServices.java:1943)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        ... 3 more

Append a replicable table-cf config for the specified peer
Examples:

  # append a table / table-cf to be replicable for a peer
  hbase> append_peer_tableCFs '2', { "ns1:table4" => ["cfA", "cfB"]}


Took 8.1481 seconds
```

disable_peer

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Disables the specified replication relationship (peer). HBase will no longer send the replication stream to the destination cluster (specified in the peer), but it will still keep track of all new WALs, which are necessary for replication if and when the peer will be re-enabled. When enabling or disabling replication, the WALs are retained as long as the peer exists.

TIP

If you want to stop the replication stream permanently, use the remove_peer command instead.

Usage

```
disable_peer '<peer_id>'
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens

Examples

```
hbase(main):013:0> disable_peer '1'
Took 0.0179 seconds
```

disable_table_replication

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Disables replication for the specified table. The `REPLICATION_SCOPE` flag is set to `0` for all table column families. Also, the table stops being output via the list_replicated_tables command.

Usage

```
disable_table_replication '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):010:0> disable_table_replication 'articles'
Replication of table 'articles' successfully disabled.
Took 1.4515 seconds
hbase(main):069:0> list_replicated_tables
TABLE:COLUMNFAMILY                                  ReplicationType
0 row(s)
Took 0.0066 seconds
hbase(main):070:0> describe 'articles'
Table articles is ENABLED
articles
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => '
false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false'
, IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE
=> 'true', BLOCKSIZE => '6553
6'}
{NAME => 'tags', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'f
alse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '0', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
 IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536
'}
2 row(s)
Took 0.0208 seconds
```

enable_peer

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Enables the previously disabled replication relationship (peer). After the command execution, the replication stream to the destination cluster (specified in the peer) starts at the point the peer was disabled.

Usage

```
enable_peer '<peer_id>'
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens

Examples

```
hbase(main):023:0> enable_peer '1'
Took 0.0148 seconds
```

enable_table_replication

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Enables replication for the specified table. The `REPLICATION_SCOPE` flag is set to `1` for all table column families. Also, the table starts being output via the list_replicated_tables command.

Usage

```
enable_table_replication '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):021:0> enable_table_replication 'articles'
The replication of table 'articles' successfully enabled
Took 1.2512 seconds
hbase(main):064:0> list_replicated_tables
TABLE:COLUMNFAMILY                                  ReplicationType
 articles:basic                                     GLOBAL
 articles:tags                                      GLOBAL
2 row(s)
Took 0.0381 seconds
hbase(main):065:0> describe 'articles'
Table articles is ENABLED
articles
COLUMN FAMILIES DESCRIPTION
{NAME => 'basic', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => '
false', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '1', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false'
, IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE
=> 'true', BLOCKSIZE => '6553
6'}
{NAME => 'tags', VERSIONS => '5', EVICT_BLOCKS_ON_CLOSE => 'false', NEW_VERSION_BEHAVIOR => 'false', KEEP_DELETED_CELLS =>
'FALSE', CACHE_DATA_ON_WRITE => 'f
alse', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', MIN_VERSIONS => '0', REPLICATION_SCOPE => '1', BLOOMFILTER => 'ROW',
CACHE_INDEX_ON_WRITE => 'false',
 IN_MEMORY => 'false', CACHE_BLOOMS_ON_WRITE => 'false', PREFETCH_BLOCKS_ON_OPEN => 'false', COMPRESSION => 'NONE', BLOCKCACHE =>
'true', BLOCKSIZE => '65536
'}
2 row(s)
Took 0.1300 seconds
```

get_peer_config

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - HBase clusters peer
  - Custom replication endpoint

Definition

Lists configuration parameters of the defined replication relationship (peer).

Usage

```
get_peer_config '<peer_id>'
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens

Examples

HBase clusters peer

```
hbase(main):002:0> get_peer_config '1'
 Cluster Key          serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.i
                      nternal,serov-adh-3.ru-central1.internal:2181:/hbase
Took 0.0017 seconds
=> #<Java::OrgApacheHadoopHbaseReplication::ReplicationPeerConfig:0x6dc2279c>
```

Custom replication endpoint

```
hbase(main):019:0> get_peer_config '2'
 Cluster Key                         bds-adh-1.ru-central1.internal,bds-adh-2.ru-central1.internal,bds-adh-3.ru-central1.
internal:2181:/hbase
 Replication Endpoint                test
 config1                             value1
Took 0.0008 seconds
=> #<Java::OrgApacheHadoopHbaseReplication::ReplicationPeerConfig:0x4a4ff5ed>
```

list_peers

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists all replication relationships (peers) between the source and destination HBase clusters or custom replication endpoints.

Usage

```
list_peers
```

Output parameters

Parameter	Description
PEER_ID	A unique identifier of the replication peer
CLUSTER_KEY	A full address of the second HBase cluster in the replication relationship. Uses the following format based on HBase configuration parameters: `<hbase.zookeeper.quorum>:<hbase.zookeeper.property.clientPort>:<zookeeper.znode.parent>`
ENDPOINT_CLASSNAME	A name of the custom class. Filled in if replication is configured for a custom replication endpoint
STATE	A replication peer state. Can be changed via the disable_peer and enable_peer commands
REPLICATE_ALL	A boolean value of the `REPLICATE_ALL` flag. Can be changed via the set_peer_replicate_all command
NAMESPACES	Namespaces that are included or excluded from the peer configuration. Excluded entities are displayed with a prefix `!` (for example, `!ns1;ns2`)
TABLE_CFS	Tables/column families, included or excluded from the peer configuration. Excluded entities are displayed with a prefix `!` (for example, `!default.articles`)
BANDWIDTH	A source bandwidth (per Region Server). Can be changed via the set_peer_bandwidth command

Examples

```
hbase(main):033:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
default.articles 0
1 row(s)
Took 0.0060 seconds
=> #<Java::JavaUtil::ArrayList:0x41059616>
```

list_peer_configs

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Lists configuration parameters of all replication relationships (peers) that are defined for the current HBase cluster.

Usage

```
list_peer_configs
```

Examples

```
hbase(main):011:0> list_peer_configs
 PeerId                                  1
 Cluster Key                             serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase

Took 0.0076 seconds
=> {"1"=>#<Java::OrgApacheHadoopHbaseReplication::ReplicationPeerConfig:0x37cf91d8>}
```

list_replicated_tables

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- Without regular expressions
- With regular expressions

Definition

Lists all tables and column families that are replicated from the current HBase cluster.

Usage

```
list_replicated_tables ['<table_regex>']
```

Arguments

Parameter	Description
table_regex	A regular expression for searching tables

Examples

Without regular expressions

```
hbase(main):017:0> list_replicated_tables
TABLE:COLUMNFAMILY                                    ReplicationType
 articles:basic                                       GLOBAL
 articles:tags                                        GLOBAL
2 row(s)
Took 0.0130 seconds
```

With regular expressions

```
hbase(main):031:0> list_replicated_tables 'art.*'
TABLE:COLUMNFAMILY                                    ReplicationType
 articles:basic                                       GLOBAL
 articles:tags                                        GLOBAL
2 row(s)
Took 0.0102 seconds
```

remove_peer

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Stops replication stream for the specified relationship (peer), then disables and removes it. It also clears all stored meta information on that peer.

Usage

```
remove_peer '<peer_id>'
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens

Examples

```
hbase(main):002:0> remove_peer '1'
Took 0.0281 seconds
hbase(main):003:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
0 row(s)
Took 0.0049 seconds
=> #<Java::JavaUtil::ArrayList:0x35329a05>
```

remove_peer_namespaces

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Removes namespaces from the replication relationship (peer) configuration.

Usage

```
remove_peer_namespaces '<peer_id>', <namespaces_array[]>
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name

Examples

```
hbase(main):022:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1 default.articles 0
1 row(s)
Took 0.0090 seconds
=> #<Java::JavaUtil::ArrayList:0x1e5b33e5>
hbase(main):002:0> remove_peer_namespaces '1', ['ns1']
Took 0.0269 seconds
hbase(main):003:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
default.articles 0
1 row(s)
Took 0.0060 seconds
=> #<Java::JavaUtil::ArrayList:0x7645b7d>
```

remove_peer_tableCFs

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Removes tables/column families from the replication relationship (peer) configuration.

Usage

```
remove_peer_tableCFs '<peer_id>',
                     {'[<namespace_name>:]<table_name>' => <column_families_array[]>[,...]}
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespace_name	A namespace name
table_name	A table name
column_families_array	An array of strings specified in the format `'<column_family>'`, where `<column_family>` — a column family name

Examples

```
hbase(main):014:0> show_peer_tableCFs '1'
default.articles
Took 0.0066 seconds
=> "default.articles"
hbase(main):014:0> remove_peer_tableCFs '1', {'articles' => []}
Took 0.0306 seconds
hbase(main):017:0> show_peer_tableCFs '1'

Took 0.0037 seconds
```

set_peer_bandwidth

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Setting a bandwidth equal to 2 MB
  - Resetting a bandwidth

Definition

Sets the source bandwidth (per Region Server) for the specified replication relationship (peer).

TIP

To return to the default bandwidth value configured at the server side, call this command without arguments.

Usage

```
set_peer_bandwidth '<peer_id>'[, <bandwidth_value>]
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
bandwidth_value	A desired bandwidth value (in bytes)

Examples

Setting a bandwidth equal to 2 MB

```
hbase(main):049:0> set_peer_bandwidth '1', 2097152
Took 0.0241 seconds
hbase(main):050:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1;ns2 default.articles;default.table10;default.temp1:cf1 2097152
1 row(s)
Took 0.0069 seconds
=> #<Java::JavaUtil::ArrayList:0x780c0>
```

Resetting a bandwidth

```
hbase(main):051:0> set_peer_bandwidth '1'
Took 0.0096 seconds
hbase(main):052:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1;ns2 default.articles;default.table10;default.temp1:cf1 0
1 row(s)
Took 0.0080 seconds
=> #<Java::JavaUtil::ArrayList:0x4ce18cec>
```

set_peer_exclude_namespaces

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Adding excluded namespaces
  - Removing excluded namespaces

Definition

Sets not replicable namespaces for the replication relationship (peer). Remember the following restrictions:

- The `REPLICATE_ALL` flag should have the `true` value in the replication peer configuration (see set_peer_replicate_all).
- Setting an excluded namespace means that all tables in this namespace are not replicated. If the peer configuration already contains some excluded table (added via set_peer_exclude_tableCFs), then setting its namespace with this command is not allowed.
- If you need to clear excluded namespaces, run the current command with an empty array as the second input parameter. Do not use the remove_peer_namespaces command.

Usage

```
set_peer_exclude_namespaces '<peer_id>', <namespaces_array[]>
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name

Examples

Adding excluded namespaces

```
hbase(main):013:0> set_peer_exclude_namespaces '1', ['ns1', 'ns2']
Took 0.0135 seconds
hbase(main):014:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED true
!ns1;ns2  0
1 row(s)
Took 0.0148 seconds
=> #<Java::JavaUtil::ArrayList:0x7d977a20>
```

Removing excluded namespaces

```
hbase(main):017:0> set_peer_exclude_namespaces '1', []
Took 0.0095 seconds
hbase(main):018:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED true
0
1 row(s)
Took 0.0112 seconds
=> #<Java::JavaUtil::ArrayList:0x71a2df1>
```

set_peer_exclude_tableCFs

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Excluding tables/column families
  - Removing excluded tables/column families

Definition

Sets not replicable tables/column families for the replication relationship (peer). Remember the following restrictions:

- The `REPLICATE_ALL` flag should have the `true` value in the replication peer configuration (see set_peer_replicate_all).
- If the peer configuration already contains some excluded namespace (added via set_peer_exclude_namespaces), then setting its tables with this command is not allowed.
- If you need to clear excluded tables/column families, run the current command with an empty dictionary as the second input parameter. Do not use the remove_peer_tableCFs command.

Usage

```
set_peer_exclude_tableCFs '<peer_id>',
                          {'[<namespace_name>:]<table_name>' => <column_families_array[]>[,...]}
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespace_name	A namespace name
table_name	A table name
column_families_array	An array of strings specified in the format `'<column_family>'`, where `<column_family>` — a column family name

Examples

Excluding tables/column families

```
hbase(main):022:0> set_peer_exclude_tableCFs '1', {'articles' => []}
Took 0.0090 seconds
hbase(main):023:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED true
!default.articles 0
1 row(s)
Took 0.0088 seconds
=> #<Java::JavaUtil::ArrayList:0x1d585fb>
```

Removing excluded tables/column families

```
hbase(main):027:0> set_peer_exclude_tableCFs '1', {}
Took 0.0085 seconds
hbase(main):028:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED true
0
1 row(s)
Took 0.0057 seconds
=> #<Java::JavaUtil::ArrayList:0x26c59348>
```

set_peer_namespaces

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
  Successful namespace setting
  Unsuccessful namespace setting

Definition

Sets all replicable namespaces for the specified replication relationship (peer). Remember the following restrictions:

- The `REPLICATE_ALL` flag should have the `false` value in the replication peer configuration (see set_peer_replicate_all).
- Setting a namespace means that all tables in this namespace with the `REPLICATION_SCOPE` flag != 0 are replicated. If the peer configuration already contains some table (added via append_peer_tableCFs or set_peer_tableCFs), then setting its namespace with this command is not allowed.
- If you need to clear included namespaces, run the current command with an empty array as the second input parameter. You can also use the remove_peer_namespaces command.

CAUTION

This command removes the previously defined namespaces from the replication peer configuration (unlike append_peer_namespaces).

Usage

```
set_peer_namespaces '<peer_id>', <namespaces_array[]>
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name

Examples

Successful namespace setting

```
hbase(main):054:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1;ns2 default.articles;default.table10;default.temp1:cf1 0
1 row(s)
Took 0.0090 seconds
=> #<Java::JavaUtil::ArrayList:0x51c008fd>
hbase(main):055:0> set_peer_namespaces '1', ['ns1']
Took 0.0130 seconds
hbase(main):056:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1 default.articles;default.table10;default.temp1:cf1 0
1 row(s)
Took 0.0084 seconds
=> #<Java::JavaUtil::ArrayList:0x3c98981e>
```

Unsuccessful namespace setting

```
hbase(main):051:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED false
ns1 default.articles 0
1 row(s)
Took 0.0081 seconds
=> #<Java::JavaUtil::ArrayList:0x126af200>
hbase(main):052:0> set_peer_namespaces '1', ['default']

ERROR: java.io.IOException: Table-cfs articles is conflict with namespaces default in peer config
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:472)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)
Caused by: org.apache.hadoop.hbase.replication.ReplicationException: Table-cfs articles is conflict with namespaces default in
peer config
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.
checkNamespacesAndTableCfsConfigConflict(ReplicationManager.java:181)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.checkPeerConfig(ReplicationManager.java:148)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.updatePeerConfig(ReplicationManager.java:103)
        at org.apache.hadoop.hbase.master.HMaster.updateReplicationPeerConfig(HMaster.java:3535)
        at org.apache.hadoop.hbase.master.MasterRpcServices.updateReplicationPeerConfig(MasterRpcServices.java:1943)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        ... 3 more

  Set the replicable namespaces config for the specified peer.

  1. The replicate_all flag need to be false when set the replicable namespaces.
  2. Set a namespace in the peer config means that all tables in this namespace
     will be replicated to the peer cluster. If peer config already has a table,
     then not allow set this table's namespace to the peer config.

  Examples:

    # set namespaces config is null, then the table-cfs config decide
    # which table to be replicated.
    hbase> set_peer_namespaces '1', []
    # set namespaces to be replicable for a peer.
    # set a namespace in the peer config means that all tables in this
    # namespace (with replication_scope != 0 ) will be replicated.
    hbase> set_peer_namespaces '2', ["ns1", "ns2"]


Took 8.1730 seconds
```

set_peer_replicate_all

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
  Setting true
  Setting false

Definition

Sets the `REPLICATE_ALL` flag to `true` or `false` for the specified replication relationship (peer). Remember the following restrictions:

- Before setting the `REPLICATE_ALL` flag to `true`, clear the previously added namespaces and tables/column families from the replication peer configuration. Use the remove_peer_namespaces and remove_peer_tableCFs commands for this purpose. Otherwise you will receive errors.
- If the `REPLICATE_ALL` flag is `true`, then all user tables with the `REPLICATION_SCOPE` flag != 0 are replicated. But you can use the set_peer_exclude_namespaces and set_peer_exclude_tableCFs commands to exclude some namespaces and tables/column families from the replication process.
- Before setting the `REPLICATE_ALL` flag to `false`, clear the previously excluded namespaces and tables/column families from the replication peer configuration. Use the set_peer_exclude_namespaces and set_peer_exclude_tableCFs commands with empty second parameter for this purpose. Otherwise you will receive errors.
- If the `REPLICATE_ALL` flag is `false`, only the selected tables are replicated. To select replicable namespaces, use the set_peer_namespaces and append_peer_namespaces commands. To select replicable tables/column families, use set_peer_tableCFs and append_peer_tableCFs.

Usage

```
set_peer_replicate_all <flag_value>
```

Arguments

Parameter	Description
flag_value	A new value of the `REPLICATE_ALL` flag

Examples

Setting true

```
hbase(main):039:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED false
ns1;ns2 default.articles 0
1 row(s)
Took 0.0108 seconds
=> #<Java::JavaUtil::ArrayList:0x2633d09c>
hbase(main):040:0> set_peer_replicate_all '1', true

ERROR: java.io.IOException: Need clean namespaces or table-cfs config firstly when replicate_all flag is true
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:472)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)
Caused by: org.apache.hadoop.hbase.replication.ReplicationException: Need clean namespaces or table-cfs config firstly when
replicate_all flag is true
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.checkPeerConfig(ReplicationManager.java:134)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.updatePeerConfig(ReplicationManager.java:103)
        at org.apache.hadoop.hbase.master.HMaster.updateReplicationPeerConfig(HMaster.java:3535)
        at org.apache.hadoop.hbase.master.MasterRpcServices.updateReplicationPeerConfig(MasterRpcServices.java:1943)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        ... 3 more

  Set the replicate_all flag to true or false for the specified peer.

  If replicate_all flag is true, then all user tables (REPLICATION_SCOPE != 0)
  will be replicate to peer cluster. But you can use 'set_peer_exclude_namespaces'
  to set which namespaces can't be replicated to peer cluster. And you can use
  'set_peer_exclude_tableCFs' to set which tables can't be replicated to peer
  cluster.

  If replicate_all flag is false, then all user tables cannot be replicate to
  peer cluster. Then you can use 'set_peer_namespaces' or 'append_peer_namespaces'
  to set which namespaces will be replicated to peer cluster. And you can use
  'set_peer_tableCFs' or 'append_peer_tableCFs' to set which tables will be
  replicated to peer cluster.

  Notice: When you want to change a peer's replicate_all flag from false to true,
          you need clean the peer's NAMESPACES and TABLECFS config firstly.
          When you want to change a peer's replicate_all flag from true to false,
          you need clean the peer's EXCLUDE_NAMESPACES and EXCLUDE_TABLECFS
          config firstly.

  Examples:

    # set replicate_all flag to true
    hbase> set_peer_replicate_all '1', true
    # set replicate_all flag to false
    hbase> set_peer_replicate_all '1', false

Took 8.1956 seconds
hbase(main):042:0> remove_peer_namespaces '1', ['ns1', 'ns2']
Took 0.0100 seconds
hbase(main):043:0> remove_peer_tableCFs '1', {'articles' => []}
Took 0.0082 seconds
hbase(main):044:0> set_peer_replicate_all '1', true
Took 0.0081 seconds
hbase(main):045:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED true
0
1 row(s)
Took 0.0086 seconds
=> #<Java::JavaUtil::ArrayList:0x4f82248f>
```

Setting false

```
hbase(main):032:0> set_peer_replicate_all '1', false

ERROR: java.io.IOException: Need clean exclude-namespaces or exclude-table-cfs config firstly when replicate_all flag is false
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:472)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)
Caused by: org.apache.hadoop.hbase.replication.ReplicationException: Need clean exclude-namespaces or exclude-table-cfs config
firstly when replicate_all flag is false
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.checkPeerConfig(ReplicationManager.java:144)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.updatePeerConfig(ReplicationManager.java:103)
        at org.apache.hadoop.hbase.master.HMaster.updateReplicationPeerConfig(HMaster.java:3535)
        at org.apache.hadoop.hbase.master.MasterRpcServices.updateReplicationPeerConfig(MasterRpcServices.java:1943)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        ... 3 more

  Set the replicate_all flag to true or false for the specified peer.

  If replicate_all flag is true, then all user tables (REPLICATION_SCOPE != 0)
  will be replicate to peer cluster. But you can use 'set_peer_exclude_namespaces'
  to set which namespaces can't be replicated to peer cluster. And you can use
  'set_peer_exclude_tableCFs' to set which tables can't be replicated to peer
  cluster.

  If replicate_all flag is false, then all user tables cannot be replicate to
  peer cluster. Then you can use 'set_peer_namespaces' or 'append_peer_namespaces'
  to set which namespaces will be replicated to peer cluster. And you can use
  'set_peer_tableCFs' or 'append_peer_tableCFs' to set which tables will be
  replicated to peer cluster.

  Notice: When you want to change a peer's replicate_all flag from false to true,
          you need clean the peer's NAMESPACES and TABLECFS config firstly.
          When you want to change a peer's replicate_all flag from true to false,
          you need clean the peer's EXCLUDE_NAMESPACES and EXCLUDE_TABLECFS
          config firstly.

  Examples:

    # set replicate_all flag to true
    hbase> set_peer_replicate_all '1', true
    # set replicate_all flag to false
    hbase> set_peer_replicate_all '1', false

Took 8.6671 seconds
hbase(main):033:0> set_peer_exclude_tableCFs '1', {}
Took 0.0110 seconds
hbase(main):034:0> set_peer_exclude_namespaces '1', []
Took 0.0080 seconds
hbase(main):035:0> set_peer_replicate_all '1', false
Took 0.0077 seconds
hbase(main):036:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED false
0
1 row(s)
Took 0.0052 seconds
=> #<Java::JavaUtil::ArrayList:0x3909a854>
```

set_peer_tableCFs

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- Successful table setting
- Unsuccessful table setting

Definition

Sets all replicable tables/column families for the replication relationship (peer). Remember the following restrictions:

- The `REPLICATE_ALL` flag should have the `false` value in the replication peer configuration (see set_peer_replicate_all).
- If the peer configuration already contains some namespace (added via append_peer_namespaces or set_peer_namespaces), then setting its tables with this command is not allowed.
- If you need to clear included tables/column families, run the current command with an empty dictionary as the second input parameter. You can also use the remove_peer_tableCFs command.

CAUTION

This command removes the previously defined tables/column families from the replication peer configuration (unlike append_peer_tableCFs).

Usage

```
set_peer_tableCFs '<peer_id>',
                  {'[<namespace_name>:]<table_name>' => <column_families_array[]>[,...]}
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
namespace_name	A namespace name
table_name	A table name
column_families_array	An array of strings specified in the format `'<column_family>'`, where `<column_family>` — a column family name

Examples

Successful table setting

```
hbase(main):060:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1 default.articles;default.table10;default.temp1:cf1 0
1 row(s)
Took 0.0089 seconds
=> #<Java::JavaUtil::ArrayList:0x572b4072>
hbase(main):061:0> set_peer_tableCFs '1', {'articles' => []}
Took 0.0147 seconds
hbase(main):062:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  ENABLED false
ns1 default.articles 0
1 row(s)
Took 0.0079 seconds
=> #<Java::JavaUtil::ArrayList:0x53e166ad>
```

Unsuccessful table setting

```
hbase(main):055:0> list_peers
 PEER_ID CLUSTER_KEY ENDPOINT_CLASSNAME STATE REPLICATE_ALL NAMESPACES TABLE_CFS BANDWIDTH
 1 serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-central1.internal:2181:/hbase  DISABLED false
ns1 default.articles 0
1 row(s)
Took 0.0089 seconds
=> #<Java::JavaUtil::ArrayList:0x785d6a98>
hbase(main):056:0> set_peer_tableCFs '1', {'ns1:temp2' => []}

ERROR: java.io.IOException: Table-cfs ns1:temp2 is conflict with namespaces ns1 in peer config
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:472)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)
Caused by: org.apache.hadoop.hbase.replication.ReplicationException: Table-cfs ns1:temp2 is conflict with namespaces ns1 in peer
config
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.
checkNamespacesAndTableCfsConfigConflict(ReplicationManager.java:181)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.checkPeerConfig(ReplicationManager.java:148)
        at org.apache.hadoop.hbase.master.replication.ReplicationManager.updatePeerConfig(ReplicationManager.java:103)
        at org.apache.hadoop.hbase.master.HMaster.updateReplicationPeerConfig(HMaster.java:3535)
        at org.apache.hadoop.hbase.master.MasterRpcServices.updateReplicationPeerConfig(MasterRpcServices.java:1943)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        ... 3 more

  Set the replicable table-cf config for the specified peer.

  Note:
  1. The replicate_all flag need to be false when set the replicable table-cfs.
  2. Can't set a table to table-cfs config if it's namespace already was in
     namespaces config of this peer.

  Examples:

    # set table-cfs config is null, then the namespaces config decide which
    # table to be replicated.
    hbase> set_peer_tableCFs '1'
    # set table / table-cf to be replicable for a peer, for a table without
    # an explicit column-family list, all replicable column-families (with
    # replication_scope == 1) will be replicated
    hbase> set_peer_tableCFs '2',
     { "ns1:table1" => [],
     "ns2:table2" => ["cf1", "cf2"],
     "ns3:table3" => ["cfA", "cfB"]}


Took 8.1736 seconds
```

show_peer_tableCFs

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Shows replicable tables/column families for the specified replication relationship (peer).

Usage

```
show_peer_tableCFs '<peer_id>'
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens

Examples

```
hbase(main):014:0> show_peer_tableCFs '1'
default.articles
Took 0.0066 seconds
=> "default.articles"
```

update_peer_config

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Updates configuration parameters of the specified replication relationship (peer). Can be useful to configure custom replication endpoints.

Usage

```
update_peer_config '<peer_id>'[,
                   DATA => {'<key_name>' => <key_value>, [...]}][,
                   CONFIG => {'<config_name>' => '<config_value>', [...]}]
```

Arguments

Parameter	Description
peer_id	A unique identifier of the replication peer. Should not contain hyphens
key_name	A name of the peer data parameter
key_value	A value of the peer data parameter
config_name	A name of the peer configuration parameter
config_value	A value of the peer configuration parameter

Examples

```
hbase(main):010:0> update_peer_config '1', CONFIG => {'config1' => 'value1'}
Took 0.0139 seconds
hbase(main):011:0> get_peer_config '1'
 Cluster Key                             serov-adh-1.ru-central1.internal,serov-adh-2.ru-central1.internal,serov-adh-3.ru-
central1.internal:2181:/hbase
 config1                                 value1
Took 0.0047 seconds
=> #<Java::OrgApacheHadoopHbaseReplication::ReplicationPeerConfig:0x1ff463bb>
```

clone_snapshot

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Creates a new table by cloning the specified snapshot. The subsequent writing on the newly created table will not influence the snapshot data.

Usage

```
clone_snapshot '<snapshot_name>',
               '[<namespace_name>:][table_name]'[,
               {RESTORE_ACL => <restore_acl_value>}]
```

Arguments

Parameter	Description
snapshot_name	A snapshot name
namespace_name	A namespace name
table_name	A table name
restore_acl_value	A boolean field that defines whether or not to restore ACL

Examples

```
hbase(main):044:0> create 'temp10', 'cf1'
Created table temp10
Took 1.3420 seconds
=> Hbase::Table - temp10
hbase(main):045:0> clone_snapshot 'snap1', 'table10'
Took 1.2917 seconds
hbase(main):046:0> scan 'table10'
ROW                   COLUMN+CELL
 r1                   column=cf1:c1, timestamp=1637434534105, value=value3
 r1                   column=cf1:c2, timestamp=1637434285911, value=value4
 r2                   column=cf1:c1, timestamp=1637434293457, value=value5
2 row(s)
Took 0.0304 seconds
```

delete_all_snapshot

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Deletes all snapshots with the name matching the given regular expression.

Usage

```
delete_all_snapshot '<snapshot_regex>'
```

Arguments

Parameter	Description
snapshot_regex	A regular expression for searching snapshots

Examples

```
hbase(main):056:0> delete_all_snapshot 'snap.*'
SNAPSHOT              TABLE + CREATION TIME
No snapshots matched the regex snap.*
Took 0.0093 seconds
```

delete_snapshot

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Deletes a snapshot with the specified name.

Usage

```
delete_snapshot '<snapshot_name>'
```

Arguments

Parameter	Description
snapshot_name	A snapshot name

Examples

```
hbase(main):048:0> list_snapshots
SNAPSHOT              TABLE + CREATION TIME
 snap1                temp1 (2021-11-24 06:31:27 UTC)
 snap2                ns1:temp2 (2021-11-24 06:32:21 UTC)
2 row(s)
Took 0.0153 seconds
=> ["snap1", "snap2"]
hbase(main):049:0> delete_snapshot 'snap2'
Took 0.0160 seconds
hbase(main):050:0> list_snapshots
SNAPSHOT              TABLE + CREATION TIME
 snap1                temp1 (2021-11-24 06:31:27 UTC)
1 row(s)
Took 0.0086 seconds
=> ["snap1"]
```

delete_table_snapshots

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
- Without using regular expressions
- Using regular expressions

Definition

Deletes snapshots of the specified table. Allows to use regular expressions for defining a table name and — optionally — a snapshot name.

Usage

```
delete_table_snapshots '<table_regex>'[, '<snapshot_regex>']
```

Arguments

Parameter	Description
table_regex	A regular expression for searching tables
snapshot_regex	A regular expression for searching snapshots

Examples

Without using regular expressions

```
hbase(main):053:0> delete_table_snapshots 'temp1'
SNAPSHOT              TABLE + CREATION TIME
 snap1                temp1 (2021-11-24 06:31:27 UTC)

Delete the above 1 snapshots (y/n)?
y
Successfully deleted snapshot: snap1

Took 0.0207 seconds
=> [#<Java::OrgApacheHadoopHbaseClient::SnapshotDescription:0x3578a45b>]
```

Using regular expressions

```
hbase(main):054:0> delete_table_snapshots '.*', 'snap.*'
SNAPSHOT              TABLE + CREATION TIME
No snapshots matched the table name regular expression .* and the snapshot name regular expression snap.*
Took 0.0066 seconds
```

list_snapshots

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Without using a regular expression
  - Using a regular expression

Definition

Returns a list of snapshots — all or matching the given regular expression.

Usage

```
list_snapshots ['<snapshot_regex>']
```

Arguments

Parameter	Description
snapshot_regex	A regular expression for searching snapshots

Examples

Without using a regular expression

```
hbase(main):011:0> list_snapshots
SNAPSHOT               TABLE + CREATION TIME
 snap1                 temp1 (2021-11-24 06:31:27 UTC)
 snap2                 ns1:temp2 (2021-11-24 06:32:21 UTC)
2 row(s)
Took 0.0403 seconds
=> ["snap1", "snap2"]
```

Using a regular expression

```
hbase(main):013:0> list_snapshots 'snap1*'
SNAPSHOT               TABLE + CREATION TIME
 snap1                 temp1 (2021-11-24 06:31:27 UTC)
1 row(s)
Took 0.0157 seconds
=> ["snap1"]
```

list_table_snapshots

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Without using regular expressions
  - Using regular expressions

Definition

Returns snapshots of the specified table. Allows to use regular expressions for defining a table name and — optionally — a snapshot name.

Usage

```
list_table_snapshots '<table_regex>'[, '<snapshot_regex>']
```

Arguments

Parameter	Description
table_regex	A regular expression for searching tables
snapshot_regex	A regular expression for searching snapshots

Examples

Without using regular expressions

```
hbase(main):015:0> list_table_snapshots 'temp1'
SNAPSHOT              TABLE + CREATION TIME
 snap1                temp1 (2021-11-24 06:31:27 UTC)
1 row(s)
Took 0.0285 seconds
=> ["snap1"]
```

Using regular expressions

```
hbase(main):022:0> list_table_snapshots '.*', 'snap.*'
SNAPSHOT              TABLE + CREATION TIME
 snap1                temp1 (2021-11-24 06:31:27 UTC)
 snap2                ns1:temp2 (2021-11-24 06:32:21 UTC)
2 row(s)
Took 0.0180 seconds
=> ["snap1", "snap2"]
```

restore_snapshot

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Restores the specified snapshot. Can be applied only to disabled tables.

Usage

```
restore_snapshot '<snapshot_name>'[, {RESTORE_ACL => <restore_acl_value>}]
```

Arguments

Parameter	Description
snapshot_name	A snapshot name
restore_acl_value	A boolean field that defines whether or not to restore ACL

Examples

```
hbase(main):026:0> delete 'temp1', 'r2', 'cf1:c1'
Took 0.0769 seconds
hbase(main):028:0> delete 'temp1', 'r1', 'cf1:c2'
Took 0.0087 seconds
hbase(main):029:0> scan 'temp1'
ROW                   COLUMN+CELL
 r1                   column=cf1:c1, timestamp=1637434534105, value=value3
1 row(s)
Took 0.0236 seconds
hbase(main):038:0> disable 'temp1'
Took 0.8044 seconds
hbase(main):039:0> restore_snapshot 'snap1'
Took 1.3029 seconds
hbase(main):041:0> enable 'temp1'
Took 1.2668 seconds
hbase(main):042:0> scan 'temp1'
ROW                   COLUMN+CELL
 r1                   column=cf1:c1, timestamp=1637434534105, value=value3
 r1                   column=cf1:c2, timestamp=1637434285911, value=value4
 r2                   column=cf1:c1, timestamp=1637434293457, value=value5
2 row(s)
Took 0.0321 seconds
```

snapshot

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Takes a snapshot of the specified table.

Usage

```
snapshot '[<namespace_name>:]<table_name>',
         '<snapshot_name>'[,
         {SKIP_FLUSH => <skip_flush_value>}]
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
snapshot_name	A snapshot name
skip_flush_value	A boolean field that defines whether or not to skip flush operations

Examples

```
hbase(main):004:0> snapshot 'temp1', 'snap1'
Took 0.6258 seconds
hbase(main):009:0> snapshot 'ns1:temp2', 'snap2'
Took 0.5650 seconds
```

update_all_config

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Reloads a subset of configuration parameters on all cluster servers.

Usage

```
update_all_config
```

Examples

```
hbase(main):018:0> update_all_config
Took 0.2048 seconds
```

update_config

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Reloads a subset of configuration parameters on the specified cluster servers.

Usage

```
update_config '<server_name>,<port_number>,<start_code>'
```

Arguments

Parameter	Description
server_name	A Server name
port_number	A port number
start_code	A Server start code

TIP

You can get `<server_name>`, `<port_number>`, and `<start_code>` using the status 'simple' command.

Examples

```
hbase(main):023:0> update_config 'bds-adh-2.ru-central1.internal,16000,1637647837862'
Took 0.0168 seconds
```

list_quotas

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Listing all quotas
  - Listing quotas matching the regular expression

Definition

Returns quotas added to HBase — all or matching the given regular expressions.

Usage

```
list_quotas [USER => '<user_regex>'][,
             TABLE => '<table_regex>'][,
             NAMESPACE => '<namespace_regex>']
```

Arguments

Parameter	Description
namespace_regex	A regular expression for searching namespaces
table_regex	A regular expression for searching tables
user_regex	A regular expression for searching users

Examples

Listing all quotas

```
hbase(main):006:0> list_quotas
OWNER                   QUOTAS
 TABLE => temp1         TYPE => SPACE, TABLE => temp1, LIMIT => 1073741824, VIOLATION_POL
                        ICY => NO_INSERTS
1 row(s)
Took 0.1327 seconds
```

Listing quotas matching the regular expression

```
hbase(main):007:0> list_quotas TABLE => 't.*'
OWNER                   QUOTAS
 TABLE => temp1         TYPE => SPACE, TABLE => temp1, LIMIT => 1073741824, VIOLATION_POL
                        ICY => NO_INSERTS
1 row(s)
Took 0.0670 seconds
hbase(main):008:0> list_quotas NAMESPACE => 'ns.*'
OWNER                   QUOTAS
0 row(s)
Took 0.0241 seconds
```

list_snapshot_sizes

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns the size of snapshots made for tables with defined space quotas. This information is used in the space quota size computation algorithms.

TIP

It may take some time between creating a snapshot and adding it to the command output. The file size can be returned when the table, from which the snapshot was created, no longer refers to this file.

Usage

```
list_snapshot_sizes
```

Examples

```
hbase(main):023:0> snapshot "temp1", "snap1"
Took 0.6061 seconds
hbase(main):026:0> list_snapshot_sizes
SNAPSHOT                 SIZE
 snap1                   0
1 row(s)
Took 0.0095 seconds
```

list_quota_snapshots

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
- Listing without any parameters
- Listing for the specified table
- Listing for the specified namespace
- Requesting information from the specified Region Server

Definition

Returns information about tables with defined space quotas: current space usage, configured limits, quota violation details. By default, the command reads this data from the snapshots stored in the system table `hbase:quota`. As Region Servers maintain a copy of these snapshots (refreshing at a regular interval), you can use the input parameter `REGIONSERVER` — to retrieve information from the particular Region Server instead of `hbase:quota`.

A table name or namespace can be used to filter the command output.

Usage

```
list_quota_snapshots({[TABLE => '<table_name>'][,
                       NAMESPACE => '<namespace_name>'][,
                       REGIONSERVER => '<region_server_name>']})
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
region_server_name	A Region Server name

Output parameters

Parameter	Description
TABLE	A table name
USAGE	The current space usage (in bytes)
LIMIT	The defined space quota (in bytes)
IN_VIOLATION	Whether or not violation is fixed
POLICY	The policy running in the case of violation

Examples

Listing without any parameters

```
hbase(main):095:0> list_quota_snapshots
 TABLE USAGE LIMIT IN_VIOLATION POLICY
 temp1 9834 1073741824 false None
1 row(s)
Took 0.0103 seconds
```

Listing for the specified table

```
hbase(main):100:0> list_quota_snapshots({TABLE => 'temp1'})
 TABLE USAGE LIMIT IN_VIOLATION POLICY
 temp1 9834 1073741824 false None
1 row(s)
Took 0.0049 seconds
```

Listing for the specified namespace

```
hbase(main):102:0> list_quota_snapshots({NAMESPACE => 'default'})
 TABLE USAGE LIMIT IN_VIOLATION POLICY
 temp1 9834 1073741824 false None
1 row(s)
Took 0.0052 seconds
```

Requesting information from the specified Region Server

```
hbase(main):104:0> list_quota_snapshots({REGIONSERVER => 'bds-adh-1.ru-central1.internal,16020,1637744748523'})
 TABLE USAGE LIMIT IN_VIOLATION POLICY
 temp1 9834 1073741824 false None
1 row(s)
Took 0.0224 seconds
```

list_quota_table_sizes

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns the computed size of each table in the cluster. Master Server uses this information to make decisions about space quotas. In most cases, calling the list_quota_snapshots command gives a more informative result.

Usage

```
list_quota_table_sizes
```

Examples

```
hbase(main):001:0> list_quota_table_sizes
TABLE                                    SIZE
 SYSTEM.CATALOG                          34766
 t4                                      0
 SYSTEM.SEQUENCE                         0
 t5                                      0
 hbase:meta                              174888
 t6                                      0
 hbase:namespace                         5116
 temp10                                  0
 SYSTEM.STATS                            0
 hbase:acl                               9474
 temp1                                   9834
 hbase:quota                             15006
 SYSTEM.MUTEX                            4451
 SYSTEM.FUNCTION                         0
 hbase:rsgroup                           5318
 hbase:labels                            4897
 ns1:temp2                               9726
 SYSTEM.LOG                              0
 table10                                 0
19 row(s)
Took 0.3489 seconds
```

set_quota

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Applying the SPACE quota to the table
  - Applying the THROTTLE quota to the namespace
  - Removing the defined THROTTLE quota

Definition

Sets a new quota for the defined user, table, or namespace. Provides two forms of quotas:

- `SPACE` — can be applied either to tables or to namespaces. Each quota limits the target's size in the file system and defines the actions that should be performed when the target exceeds that limit. Different quotas can be applied to one table at the table and namespace levels. Table level quotas take priority over namespace level ones.
- `THROTTLE` — can be applied to users, tables, or namespaces. Each quota limits the number of requests or the amount of data that can be read, written, or both read and written per the defined time range.

Usage

- Setting a `SPACE` quota:

```
set_quota TYPE => SPACE[,
          TABLE => '[<namespace_name>:]<table_name>'] | [, NAMESPACE => '<namespace_name>'],
          LIMIT => '<space_limit_value>',
          POLICY => <space_policy_value>
```

- Removing a `SPACE` quota:

```
set_quota TYPE => SPACE[,
          TABLE => '[<namespace_name>:]<table_name>'] | [, NAMESPACE => '<namespace_name>'],
          LIMIT => NONE
```

- Setting a `THROTTLE` quota:

```
set_quota TYPE => THROTTLE[,
          THROTTLE_TYPE => READ | WRITE][,
          USER => '<user_name>'][,
          TABLE => '[<namespace_name>:]<table_name>'] | [, NAMESPACE => '<namespace_name>'],
          LIMIT => '<throttle_limit_value>'
```

- Removing a `THROTTLE` quota:

```
set_quota TYPE => THROTTLE[,
          THROTTLE_TYPE => READ | WRITE][,
          USER => '<user_name>'][,
          TABLE => '[<namespace_name>:]<table_name>'] | [, NAMESPACE => '<namespace_name>'],
          LIMIT => NONE
```

> CAUTION
>
> `SPACE` quotas require a table name or a namespace name, not both of them. `THROTTLE` quotas require a user name, or a table name, or a namespace name, or a combination user/namespace, or a combination user/table. Other variants are disallowed.

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
space_limit_value	A space limit defined for the table or the namespace. Can use standard metric suffixes: `B`, `K`, `M`, `G`, `T`, `P`. The default metric — bytes (without any shortcuts). Examples: `1G`, `2T`
space_policy_value	A policy that should be taken when the space quota violation is fixed. Possible values: - `NO_INSERTS` — disallows data ingesting (e.g. put, incr, append); - `NO_WRITES` — works in the same way as `NO_INSERTS`, but also disallows delete operations; - `NO_WRITES_COMPACTIONS` — works in the same way as `NO_WRITES`, but also disallows compactions; - `DISABLE` — disables the table.
user_name	A user name
throttle_limit_value	A limit that restricts the number of requests or data amount for read/write operations. Provides two forms of syntax: - `<requests_number>req/sec|min|hour|day` — the number of requests per the defined time range. Examples: `100req/min`, `10req/sec`; - `<data_amount>B|K|M|G|T|P/sec|min|hour|day` — the data amount per the defined time range, where `B`, `K`, `M`, `G`, `T`, `P` — standard metric suffixes. Examples: `10M/sec`, `5K/min`.

Examples

Applying the SPACE quota to the table

```
hbase(main):002:0> set_quota TYPE => SPACE, TABLE => 'temp1', LIMIT => '1G', POLICY => NO_INSERTS
Took 0.4164 seconds
```

Applying the THROTTLE quota to the namespace

```
hbase(main):003:0> set_quota TYPE => THROTTLE, NAMESPACE => 'ns1', LIMIT => '10req/sec'
Took 0.0240 seconds
```

Removing the defined THROTTLE quota

```
hbase(main):004:0> set_quota TYPE => THROTTLE, NAMESPACE => 'ns1', LIMIT => NONE
Took 0.0129 seconds
```

grant

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
- Granting without using namespaces and table names
- Granting rights on the specified namespace
- Granting rights on the specified table column

Definition

Grants a user or a group of users the specific rights.

IMPORTANT

Notice, that this command can be executed only by users with `ADMIN` rights. So, run it under superuser `hbase` or grant the appropriate rights to your user:

```
sudo -u hbase hbase shell
grant '<user_name>', 'A'
```

Usage

```
grant '<user_name>' | '@<group_name>',
      '<permissions>' [,
      '@<namespace_name>'] | [, '[<namespace_name>:]<table_name>' [, '<column_family>' [, '<column_qualifier>']]]
```

Arguments

Parameter	Description
user_name	A user name
group_name	A users group name
permissions	Either zero or more letters from the set `RWXCA`, where `R` means READ, `W` — WRITE, `X` — EXEC, `C` — CREATE, `A` — ADMIN. Examples: `RW`, `RWX`, `RWXCA`, etc.
namespace_name	A namespace name
table_name	A table name
column_family	A column family name
column_qualifier	A column qualifier

NOTE

A namespace and a group name should be preceded with `@` character.

Examples

Granting without using namespaces and table names

```
hbase(main):003:0> grant 'dasha', 'RWXCA'
Took 0.2991 seconds
```

Granting rights on the specified namespace

```
hbase(main):004:0> grant 'dasha', 'RWXCA', '@ns1'
Took 0.0422 seconds
hbase(main):010:0> user_permission '@ns1'
User                     Namespace,Table,Family,Qualifier:Permission
 dasha                   ns1,,,: [Permission: actions=READ,WRITE,EXEC,CREATE,ADMIN]
1 row(s)
Took 0.0419 seconds
```

Granting rights on the specified table column

```
hbase(main):009:0> grant 'dasha', 'RW', 'ns1:temp2', 'cf1', 'c1'
Took 0.0795 seconds
hbase(main):011:0> user_permission 'ns1:temp2'
User                     Namespace,Table,Family,Qualifier:Permission
 dasha                   ns1,ns1:temp2,,: [Permission: actions=READ,WRITE,EXEC,CREATE,ADMIN]
 dasha                   ns1,ns1:temp2,cf1,c1: [Permission: actions=READ,WRITE]
2 row(s)
Took 0.0315 seconds
```

list_security_capabilities

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns supported security capabilities.

Usage

```
list_security_capabilities
```

Examples

```
hbase(main):001:0> list_security_capabilities
AUTHORIZATION
SIMPLE_AUTHENTICATION
CELL_VISIBILITY
CELL_AUTHORIZATION
Took 0.3205 seconds
=> ["AUTHORIZATION", "SIMPLE_AUTHENTICATION", "CELL_VISIBILITY", "CELL_AUTHORIZATION"]
```

revoke

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Revoking rights on the specified table column
  - Revoking rights on the specified table
  - Revoking rights on the specified namespace
  - Revoking rights without using namespaces and tables

Definition

Revokes a user access rights.

IMPORTANT

Notice, that this command can be executed only by users with `ADMIN` rights. So, run it under superuser `hbase` or grant the appropriate rights to your user:

```
sudo -u hbase hbase shell
grant '<user_name>', 'A'
```

Usage

```
revoke '<user_name>' | '@<group_name>' [,
       '@<namespace_name>'] | [, '[<namespace_name>:]<table_name>' [, '<column_family>' [, '<column_qualifier>']]]
```

Arguments

Parameter	Description
user_name	A user name
group_name	A users group name
namespace_name	A namespace name
table_name	A table name
column_family	A column family name
column_qualifier	A column qualifier

NOTE

A namespace and a group name should be preceded with the `@` character.

Examples

Revoking rights on the specified table column

```
hbase(main):011:0> user_permission 'ns1:temp2'
User                     Namespace,Table,Family,Qualifier:Permission
 dasha                   ns1,ns1:temp2,,: [Permission: actions=READ,WRITE,EXEC,CREATE,ADMIN]
 dasha                   ns1,ns1:temp2,cf1,c1: [Permission: actions=READ,WRITE]
2 row(s)
Took 0.0315 seconds
hbase(main):001:0> revoke 'dasha', 'ns1:temp2', 'cf1', 'c1'
Took 0.5811 seconds
hbase(main):003:0> user_permission 'ns1:temp2'
User                  Namespace,Table,Family,Qualifier:Permission
 dasha                ns1,ns1:temp2,,: [Permission: actions=READ,WRITE,EXEC,CREA
                      TE,ADMIN]
1 row(s)
Took 0.0458 seconds
```

Revoking rights on the specified table

```
hbase(main):003:0> user_permission 'ns1:temp2'
User                  Namespace,Table,Family,Qualifier:Permission
 dasha                ns1,ns1:temp2,,: [Permission: actions=READ,WRITE,EXEC,CREA
                      TE,ADMIN]
1 row(s)
Took 0.0458 seconds
hbase(main):008:0> revoke 'dasha', 'ns1:temp2'
Took 0.0295 seconds
hbase(main):009:0> user_permission 'ns1:temp2'
User                  Namespace,Table,Family,Qualifier:Permission
0 row(s)
Took 0.0374 seconds
```

Revoking rights on the specified namespace

```
hbase(main):010:0> user_permission '@ns1'
User                     Namespace,Table,Family,Qualifier:Permission
 dasha                   ns1,,,: [Permission: actions=READ,WRITE,EXEC,CREATE,ADMIN]
1 row(s)
Took 0.0419 seconds
hbase(main):004:0> revoke 'dasha', '@ns1'
Took 0.0268 seconds
hbase(main):006:0> user_permission '@ns1'
User                  Namespace,Table,Family,Qualifier:Permission
0 row(s)
Took 0.0355 seconds
```

Revoking rights without using namespaces and tables

```
hbase(main):011:0> revoke 'dasha'
Took 0.0125 seconds
hbase(main):012:0> user_permission
User                  Namespace,Table,Family,Qualifier:Permission

ERROR: org.apache.hadoop.hbase.security.AccessDeniedException: Insufficient permissions for user 'dasha' (global, action=ADMIN)
        at org.apache.hadoop.hbase.security.access.AccessChecker.requireGlobalPermission(AccessChecker.java:158)
        at org.apache.hadoop.hbase.security.access.AccessChecker.requirePermission(AccessChecker.java:129)
        at org.apache.hadoop.hbase.security.access.AccessController.getUserPermissions(AccessController.java:2182)
        at org.apache.hadoop.hbase.protobuf.generated.AccessControlProtos$AccessControlService$1.
getUserPermissions(AccessControlProtos.java:10039)
        at org.apache.hadoop.hbase.protobuf.generated.AccessControlProtos$AccessControlService.callMethod(AccessControlProtos.
java:10197)
        at org.apache.hadoop.hbase.regionserver.HRegion.execService(HRegion.java:8049)
        at org.apache.hadoop.hbase.regionserver.RSRpcServices.execServiceOnRegion(RSRpcServices.java:2409)
        at org.apache.hadoop.hbase.regionserver.RSRpcServices.execService(RSRpcServices.java:2391)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.ClientProtos$ClientService$2.callBlockingMethod(ClientProtos.
java:42010)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)

Show all permissions for the particular user.
Syntax : user_permission <table>

Note: A namespace must always precede with '@' character.

For example:

    hbase> user_permission
    hbase> user_permission '@ns1'
    hbase> user_permission '@.*'
    hbase> user_permission '@^[a-c].*'
    hbase> user_permission 'table1'
    hbase> user_permission 'namespace1:table1'
    hbase> user_permission '.*'
    hbase> user_permission '^[A-C].*'

Took 0.5466 seconds
```

user_permission

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Shows all permissions for the particular user.

IMPORTANT

Notice, that this command can be executed only by users with `ADMIN` rights. So, run it under superuser `hbase` or grant the appropriate rights to your user:

```
sudo -u hbase hbase shell
grant '<user_name>', 'A'
```

Usage

```
user_permission ['@<namespace_name>' | '@<namespace_regex>' | '<table_name>' | '<table_regex>']
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
namespace_regex	A regular expression for searching namespaces
table_regex	A regular expression for searching tables

NOTE

A namespace should be preceded with the `@` character.

Examples

```
hbase(main):003:0> user_permission
User                     Namespace,Table,Family,Qualifier:Permission
 dasha                   hbase,hbase:acl,,: [Permission: actions=READ,WRITE,EXEC,CREATE,ADMIN
                         ]
 hbase                   hbase,hbase:acl,,: [Permission: actions=READ,WRITE,EXEC,CREATE,ADMIN
                         ]
2 row(s)
Took 0.6043 seconds
```

abort_procedure

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Aborts the procedure with the specified ID. Returns `true`, if the command is accepted and the procedure is in the process of aborting. Returns `false`, if the procedure could not be aborted, e.g. if the defined procedure does not exist, or it is already completed, or the command can cause corruption.

CAUTION

Use the command with caution. It is for experts only. Some procedures can be not aborted.

Usage

```
abort_procedure <proc_id>[, <may_interrupt_if_running>]
```

Arguments

Parameter	Description
proc_id	A procedure identifier (can be received by calling list_procedures)
may_interrupt_if_running	An optional boolean value, defining whether or not it is allowed to interrupt the running procedure. The default value is `true`

Examples

```
hbase(main):036:0> abort_procedure 123
false
Took 0.0157 seconds
=> 1
```

list_locks

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns all the locks in HBase.

Usage

```
list_locks
```

Examples

```
hbase(main):035:0> list_locks
Took 0.0158 seconds
=> []
```

list_procedures

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns all the procedures in HBase.

Usage

```
list_procedures
```

Examples

```
hbase(main):001:0> list_procedures
 Id Name State Submitted_Time Last_Update Parameters
 84 org.apache.hadoop.hbase.procedure2.ProcedureExecutor$FailedProcedure ROLLEDBACK 2021-11-29 11:35:46 UTC 2021-11-29 11:35:46
UTC
1 row(s)
Took 0.5023 seconds
```

add_labels

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- Adding one label
- Adding a set of labels

Definition

Adds a set of visibility labels.

Usage

```
add_labels <labels_array[]>
```

Arguments

Parameter	Description
labels_array	An array of string names for new visibility labels

Examples

Adding one label

```
hbase(main):002:0> add_labels ['B1']
Took 0.1502 seconds
```

Adding a set of labels

```
hbase(main):005:0> add_labels ['SECRET','PRIVATE']
Took 0.0127 seconds
```

clear_auths

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Clears visibility labels assigned to the defined user or group.

Usage

```
clear_auths '@<group_name>' | '<user_name>', <labels_array[]>
```

Arguments

Parameter	Description
group_name	A users group name
user_name	A user name
labels_array	An array of visibility labels

CAUTION

Calling the command without the `labels_array` parameter is available, but does not lead to any changes.

Examples

```
hbase(main):005:0> clear_auths 'dasha', ['SECRET']
Took 0.1608 seconds
hbase(main):006:0> get_auths 'dasha'
PRIVATE
Took 0.0129 seconds
=> #<Java::JavaUtil::Collections::UnmodifiableRandomAccessList:0x526893f>
```

get_auths

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Returns visibility labels assigned to the defined user or group.

Usage

```
get_auths '@<group_name>' | '<user_name>'
```

Arguments

Parameter	Description
group_name	A users group name
user_name	A user name

NOTE

A group name should be preceded with the `@` character.

Examples

```
hbase(main):006:0> get_auths 'dasha'
SECRET
PRIVATE
Took 0.0176 seconds
=> #<Java::JavaUtil::Collections::UnmodifiableRandomAccessList:0x56f2c9e8>
```

list_labels

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

- Listing of all visibility labels
- Listing of labels matching the regular expression

Definition

Returns a list of visibility labels defined in HBase. An optional regular expression parameter can be used to filter the output.

Usage

```
list_labels ['<label_regex>']
```

Arguments

Parameter	Description
label_regex	A regular expression for searching visibility labels

Examples

Listing of all visibility labels

```
hbase(main):001:0> list_labels
A1
SECRET
B1
PRIVATE
Took 0.6317 seconds
=> #<Java::JavaUtil::Collections::UnmodifiableRandomAccessList:0x534d0e20>
```

Listing of labels matching the regular expression

```
hbase(main):002:0> list_labels '.1'
A1
B1
Took 0.0087 seconds
=> #<Java::JavaUtil::Collections::UnmodifiableRandomAccessList:0x30c8c6ab>
```

set_auths

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Assigns a set of visibility labels to the defined user or group.

Usage

```
set_auths '@<group_name>' | '<user_name>', <labels_array[]>
```

Arguments

Parameter	Description
group_name	A users group name
user_name	A user name
labels_array	An array of visibility labels

NOTE

A group name should be preceded with the `@` character.

Examples

```
hbase(main):004:0> set_auths 'dasha', ['SECRET','PRIVATE']
Took 0.2063 seconds
```

set_visibility

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples
- Set visibility for all existing cells of the defined column
- Attempt to use all available parameters
- Attempt to use non-existent visibility labels

Definition

Sets the visibility expression on one or more existing table cells. Requires a table name, an expression, using available visibility labels, and a dictionary, containing specifications for scanning the desired cells.

NOTE

This command affects only existing cells and is expected to be mainly useful for feature testing and functional verification. You can also define a visibility expression for new cells, using commands put or append.

Usage

```
set_visibility '[<namespace_name>:]<table_name>',
               '<visibility_expression>',
               {COLUMNS => <columns_array[]>[,
               TIMERANGE => [<timestamp_start>, <timestamp_end>]][,
               TIMESTAMP => <cell_timestamp>][,
               ROWPREFIXFILTER => '<row_prefix>'][,
               STARTROW => '<start_row_key>'][,
               STOPROW => '<stop_row_key>'][,
               FILTER => "<filter_description>"]}
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name
visibility_expression	A logical expression that refers to existing visibility labels. Can use the following logical operators: `|` (or), `&` (and). Examples: `SECRET|PRIVATE`, `(SECRET&PRIVATE)|PUBLIC`, `SECRET&PRIVATE&PUBLIC`
columns_array	An array of strings specified in the format `'<column_family>:<column_qualifier>'`, where `<column_family>` — a column family name; `<column_qualifier>` — a column qualifier
timestamp_start	A start value for searching cells timestamps
timestamp_end	A stop value for searching cells timestamps
cell_timestamp	A cell timestamp
row_prefix	A prefix for searching row keys
start_row_key	A start value for searching row keys
stop_row_key	A stop value for searching row keys
filter_description	A filter used for searching the table cells

Examples

Set visibility for all existing cells of the defined column

```
hbase(main):010:0> set_visibility 'ns1:temp2', 'PRIVATE|SECRET', {COLUMNS => ['cf1:c1']}
1 row(s)
Took 0.0841 seconds
```

Attempt to use all available parameters

```
hbase(main):009:0> set_visibility 'ns1:temp2', 'PRIVATE|SECRET', {COLUMNS => ['cf1:c1'],  TIMERANGE => [1303668804000,
1303668904000], TIMESTAMP => 1303668804000, ROWPREFIXFILTER => 'r', STARTROW => 'r1', STOPROW => 'r2', FILTER => "ValueFilter(=,
'regexstring:value*')"}
0 row(s)
Took 0.0427 seconds
```

Attempt to use non-existent visibility labels

```
hbase(main):013:0> set_visibility 'ns1:temp2', 'Z&Y', {COLUMNS => ['cf1:c1']}

ERROR: org.apache.hadoop.hbase.exceptions.FailedSanityCheckException: Invalid visibility label Z
        at org.apache.hadoop.hbase.regionserver.HRegion.doBatchMutate(HRegion.java:4119)
        at org.apache.hadoop.hbase.regionserver.HRegion.put(HRegion.java:2975)
        at org.apache.hadoop.hbase.regionserver.RSRpcServices.mutate(RSRpcServices.java:2806)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.ClientProtos$ClientService$2.callBlockingMethod(ClientProtos.
java:42000)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)

Set the visibility expression on one or more existing cells.

Pass table name, visibility expression, and a dictionary containing
scanner specifications.  Scanner specifications may include one or more
of: TIMERANGE, FILTER, STARTROW, STOPROW, ROWPREFIXFILTER, TIMESTAMP, or COLUMNS

If no columns are specified, all columns will be included.
To include all members of a column family, leave the qualifier empty as in
'col_family:'.

The filter can be specified in two ways:
1. Using a filterString - more information on this is available in the
Filter Language document attached to the HBASE-4176 JIRA
2. Using the entire package name of the filter.

Examples:

    hbase> set_visibility 't1', 'A|B', {COLUMNS => ['c1', 'c2']}
    hbase> set_visibility 't1', '(A&B)|C', {COLUMNS => 'c1',
        TIMERANGE => [1303668804000, 1303668904000]}
    hbase> set_visibility 't1', 'A&B&C', {ROWPREFIXFILTER => 'row2',
        FILTER => "(QualifierFilter (>=, 'binary:xyz')) AND
        (TimestampsFilter ( 123, 456))"}

This command will only affect existing cells and is expected to be mainly
useful for feature testing and functional verification.

Took 0.5326 seconds
```

add_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Creates a new Region Server group.

Usage

```
add_rsgroup '<rsgroup_name>'
```

Arguments

Parameter	Description
rsgroup_name	A name of a new Region Server group

Examples

```
hbase(main):001:0> add_rsgroup 'test_group'
Took 0.4366 seconds
```

balance_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Balances a Region Server group.

Usage

```
balance_rsgroup '<rsgroup_name>'
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name

Examples

```
hbase(main):030:0> balance_rsgroup 'default'
Ran the balancer.
Took 0.7246 seconds
=> true
```

get_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Gets the Region Server group information.

Usage

```
get_rsgroup '<rsgroup_name>'
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name

Examples

```
hbase(main):031:0> get_rsgroup 'default'
SERVERS
bds-adh-1.ru-central1.internal:16020
bds-adh-2.ru-central1.internal:16020
2 row(s)
TABLES
hbase:meta
hbase:acl
SYSTEM.CATALOG
ns1:temp2
t5
t6
hbase:labels
hbase:namespace
SYSTEM.SEQUENCE
SYSTEM.LOG
SYSTEM.FUNCTION
SYSTEM.MUTEX
SYSTEM.STATS
hbase:rsgroup
14 row(s)
Took 0.0044 seconds
hbase(main):032:0> get_rsgroup 'test_group'
SERVERS
bds-adh-3.ru-central1.internal:16020
1 row(s)
TABLES
t4
temp1
2 row(s)
Took 0.0035 seconds
```

get_server_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Gets a name of the group, which the given Region Server is a member of.

Usage

```
get_server_rsgroup '<region_server_name>:<port_number>'
```

Arguments

Parameter	Description
region_server_name	A Region Server name
port_number	A port number

Examples

```
hbase(main):033:0> get_server_rsgroup 'bds-adh-3.ru-central1.internal:16020'
test_group
1 row(s)
Took 0.0170 seconds
hbase(main):034:0> get_server_rsgroup 'bds-adh-1.ru-central1.internal:16020'
default
1 row(s)
Took 0.0039 seconds
hbase(main):035:0> get_server_rsgroup 'bds-adh-2.ru-central1.internal:16020'
default
1 row(s)
Took 0.0042 seconds
```

get_table_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Gets a name of the group, which the given table is a member of.

Usage

```
get_table_rsgroup '[<namespace_name>:]<table_name>'
```

Arguments

Parameter	Description
namespace_name	A namespace name
table_name	A table name

Examples

```
hbase(main):036:0> get_table_rsgroup 'temp1'
test_group
1 row(s)
Took 0.0124 seconds
hbase(main):037:0> get_table_rsgroup 'ns1:temp2'
default
1 row(s)
Took 0.0041 seconds
```

list_rsgroups

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Listing all Region Server groups
  - Listing the groups matching the regular expression

Definition

Returns all Region Server groups. An optional regular expression parameter can be used to filter the output.

Usage

```
list_rsgroups ['<rsgroup_regex>']
```

Arguments

Parameter	Description
rsgroup_regex	A regular expression for searching Region Server groups

Examples

Listing all Region Server groups

```
hbase(main):004:0> list_rsgroups
NAME                      SERVER / TABLE
 test_group
 default                  server bds-adh-1.ru-central1.internal:16020
                          server bds-adh-2.ru-central1.internal:16020
                          server bds-adh-3.ru-central1.internal:16020
                          table hbase:meta
                          table hbase:acl
                          table SYSTEM.CATALOG
                          table t4
                          table t5
                          table t6
                          table hbase:labels
                          table temp1
                          table hbase:namespace
                          table SYSTEM.SEQUENCE
                          table SYSTEM.LOG
                          table SYSTEM.FUNCTION
                          table SYSTEM.MUTEX
                          table SYSTEM.STATS
                          table hbase:rsgroup
2 row(s)
Took 0.0289 seconds
```

Listing the groups matching the regular expression

```
hbase(main):005:0> list_rsgroups 'test*'
NAME                      SERVER / TABLE
 test_group
1 row(s)
Took 0.0069 seconds
```

move_namespaces_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Assigns the given namespaces to the specified Region Server group.

Usage

```
move_namespaces_rsgroup '<rsgroup_name>', <namespaces_array[]>
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name

Examples

```
hbase(main):022:0> create 'ns1:temp2', 'cf1'
Created table ns1:temp2
Took 1.0197 seconds
=> Hbase::Table - ns1:temp2
hbase(main):023:0> list_namespace_tables 'ns1'
TABLE
temp2
1 row(s)
Took 0.0096 seconds
=> ["temp2"]
hbase(main):024:0> list_rsgroups
NAME                     SERVER / TABLE
 test_group              server bds-adh-2.ru-central1.internal:16020
                         server bds-adh-3.ru-central1.internal:16020
                         table t4
                         table temp1
 default                 server bds-adh-1.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table ns1:temp2
                         table t5
                         table t6
                         table hbase:labels
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
2 row(s)
Took 0.0117 seconds
hbase(main):025:0> move_namespaces_rsgroup 'test_group', ['ns1']
Took 1.1491 seconds
hbase(main):026:0> list_rsgroups
NAME                     SERVER / TABLE
 test_group              server bds-adh-2.ru-central1.internal:16020
                         server bds-adh-3.ru-central1.internal:16020
                         table ns1:temp2
                         table t4
                         table temp1
 default                 server bds-adh-1.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table t5
                         table t6
                         table hbase:labels
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
2 row(s)
Took 0.0055 seconds
```

move_servers_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Assigns the given servers to the specified Region Server group.

Usage

```
move_servers_rsgroup '<rsgroup_name>', <region_servers_array[]>
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name
region_servers_array	An array of strings specified in the format `'<region_server_name>:<port_number>'`, where `<region_server_name>` — a Region Server name; `<port_number>` — a port number

Examples

```
hbase(main):014:0> move_servers_rsgroup 'test_group', ['bds-adh-3.ru-central1.internal:16020']
Took 19.6359 seconds
hbase(main):015:0> list_rsgroups
NAME                     SERVER / TABLE
 test_group              server bds-adh-3.ru-central1.internal:16020
 default                 server bds-adh-1.ru-central1.internal:16020
                         server bds-adh-2.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table t4
                         table t5
                         table t6
                         table hbase:labels
                         table temp1
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
2 row(s)
Took 0.0125 seconds
```

move_servers_namespaces_rsgroup

Daria Barysheva

Contents

To Table of Contents

[Definition](#)

[Usage](#)

[Examples](#)

Definition

Assigns the given servers and namespaces to the specified Region Server group.

Usage

```
move_servers_namespaces_rsgroup '<rsgroup_name>',
                                <region_servers_array[]>,
                                <namespaces_array[]>
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name
region_servers_array	An array of strings specified in the format `'<region_server_name>:<port_number>'`, where `<region_server_name>` — a Region Server name; `<port_number>` — a port number
namespaces_array	An array of strings specified in the format `'<namespace_name>'`, where `<namespace_name>` — a namespace name

Examples

```
hbase(main):027:0> move_servers_namespaces_rsgroup 'default', ['bds-adh-2.ru-central1.internal:16020'], ['ns1']
Took 4.2444 seconds
hbase(main):028:0> list_rsgroups
NAME                     SERVER / TABLE
 test_group              server bds-adh-3.ru-central1.internal:16020
                         table t4
                         table temp1
 default                 server bds-adh-1.ru-central1.internal:16020
                         server bds-adh-2.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table ns1:temp2
                         table t5
                         table t6
                         table hbase:labels
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
2 row(s)
Took 0.0063 seconds
```

move_servers_tables_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Assigns the given servers and tables to the specified Region Server group.

Usage

```
move_servers_rsgroup '<rsgroup_name>',
                     <region_servers_array[]>,
                     <tables_array[]>
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name
region_servers_array	An array of strings specified in the format `'<region_server_name>:<port_number>'`, where `<region_server_name>` — a Region Server name; `<port_number>` — a port number
tables_array	An array of strings specified in the format `'[<namespace_name>:]<table_name>'`, where `<namespace_name>` — a namespace name; `<table_name>` — a table name

Examples

```
hbase(main):019:0> move_servers_tables_rsgroup 'test_group',['bds-adh-2.ru-central1.internal:16020'], ['t4']
Took 27.6992 seconds
hbase(main):020:0> list_rsgroups
NAME                     SERVER / TABLE
 test_group              server bds-adh-2.ru-central1.internal:16020
                         server bds-adh-3.ru-central1.internal:16020
                         table t4
                         table temp1
 default                 server bds-adh-1.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table t5
                         table t6
                         table hbase:labels
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
2 row(s)
Took 0.0063 seconds
```

move_tables_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Assigns the given tables to the specified Region Server group.

Usage

```
move_tables_rsgroup '<rsgroup_name>', <tables_array[]>
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name
tables_array	An array of strings specified in the format `'[<namespace_name>:]<table_name>'`, where `<namespace_name>` — a namespace name; `<table_name>` — a table name

Examples

```
hbase(main):016:0> move_tables_rsgroup 'test_group', ['temp1']
Took 1.1676 seconds
hbase(main):017:0> list_rsgroups
NAME                     SERVER / TABLE
 test_group              server bds-adh-3.ru-central1.internal:16020
                         table temp1
 default                 server bds-adh-1.ru-central1.internal:16020
                         server bds-adh-2.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table t4
                         table t5
                         table t6
                         table hbase:labels
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
2 row(s)
Took 0.0097 seconds
```

remove_rsgroup

Daria Barysheva

Contents

To Table of Contents

- Definition
- Usage
- Examples
  - Attempt to remove a group with tables
  - Attempt to remove a group with servers
  - Removing a group without servers and tables

Definition

Removes the given Region Server group.

> CAUTION
>
> All servers and tables should be removed from the Region Server group before this operation.

Usage

```
remove_rsgroup '<rsgroup_name>'
```

Arguments

Parameter	Description
rsgroup_name	A Region Server group name

Examples

Attempt to remove a group with tables

```
hbase(main):018:0> remove_rsgroup 'test_group'

ERROR: org.apache.hadoop.hbase.constraint.ConstraintException: RSGroup test_group has 2 tables; you must remove these tables from
the rsgroup before the rsgroup can be removed.
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminServer.removeRSGroup(RSGroupAdminServer.java:460)
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint$RSGroupAdminServiceImpl.removeRSGroup(RSGroupAdminEndpoint.
java:254)
        at org.apache.hadoop.hbase.protobuf.generated.RSGroupAdminProtos$RSGroupAdminService.callMethod(RSGroupAdminProtos.
java:13885)
        at org.apache.hadoop.hbase.master.MasterRpcServices.execMasterService(MasterRpcServices.java:816)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)

Remove a RegionServer group.

  hbase> remove_rsgroup 'my_group'


Took 0.0049 seconds
hbase(main):019:0> move_tables_rsgroup 'default', ['t4', 'temp1']
Took 2.2508 seconds
```

Attempt to remove a group with servers

```
hbase(main):020:0> remove_rsgroup 'test_group'

ERROR: org.apache.hadoop.hbase.constraint.ConstraintException: RSGroup test_group has 1 servers; you must remove these servers
from the RSGroup beforethe RSGroup can be removed.
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminServer.removeRSGroup(RSGroupAdminServer.java:466)
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint$RSGroupAdminServiceImpl.removeRSGroup(RSGroupAdminEndpoint.
java:254)
        at org.apache.hadoop.hbase.protobuf.generated.RSGroupAdminProtos$RSGroupAdminService.callMethod(RSGroupAdminProtos.
java:13885)
        at org.apache.hadoop.hbase.master.MasterRpcServices.execMasterService(MasterRpcServices.java:816)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)

Remove a RegionServer group.

  hbase> remove_rsgroup 'my_group'


Took 0.0042 seconds
hbase(main):022:0> move_servers_rsgroup 'default', ['bds-adh-3.ru-central1.internal:16020']
Took 1.0126 seconds
```

Removing a group without servers and tables

```
hbase(main):023:0> remove_rsgroup 'test_group'
Took 0.1284 seconds
hbase(main):024:0> list_rsgroups
NAME                     SERVER / TABLE
 default                 server bds-adh-1.ru-central1.internal:16020
                         server bds-adh-2.ru-central1.internal:16020
                         server bds-adh-3.ru-central1.internal:16020
                         table hbase:meta
                         table hbase:acl
                         table SYSTEM.CATALOG
                         table ns1:temp2
                         table t4
                         table t5
                         table t6
                         table hbase:labels
                         table temp1
                         table hbase:namespace
                         table SYSTEM.SEQUENCE
                         table SYSTEM.LOG
                         table SYSTEM.FUNCTION
                         table SYSTEM.MUTEX
                         table SYSTEM.STATS
                         table hbase:rsgroup
1 row(s)
Took 0.0097 seconds
```

remove_servers_rsgroup

Daria Barysheva

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Removes the defined decommissioned servers from the Server Group. Online servers are disallowed for this operation.

Usage

```
remove_servers_rsgroup <region_servers_array[]>
```

Arguments

Parameter	Description
region_servers_array	An array of strings specified in the format `'<region_server_name>:<port_number>'`, where `<region_server_name>` — a Region Server name; `<port_number>` — a port number

Examples

```
hbase(main):013:0> remove_servers_rsgroup 'bds-adh-3.ru-central1.internal:16020'

ERROR: org.apache.hadoop.hbase.constraint.ConstraintException: Server bds-adh-3.ru-central1.internal:16020 is an online server,
not allowed to remove.
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminServer.checkForDeadOrOnlineServers(RSGroupAdminServer.java:693)
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminServer.removeServers(RSGroupAdminServer.java:609)
        at org.apache.hadoop.hbase.rsgroup.RSGroupAdminEndpoint$RSGroupAdminServiceImpl.removeServers(RSGroupAdminEndpoint.
java:350)
        at org.apache.hadoop.hbase.protobuf.generated.RSGroupAdminProtos$RSGroupAdminService.callMethod(RSGroupAdminProtos.
java:13905)
        at org.apache.hadoop.hbase.master.MasterRpcServices.execMasterService(MasterRpcServices.java:816)
        at org.apache.hadoop.hbase.shaded.protobuf.generated.MasterProtos$MasterService$2.callBlockingMethod(MasterProtos.java)
        at org.apache.hadoop.hbase.ipc.RpcServer.call(RpcServer.java:413)
        at org.apache.hadoop.hbase.ipc.CallRunner.run(CallRunner.java:130)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:324)
        at org.apache.hadoop.hbase.ipc.RpcExecutor$Handler.run(RpcExecutor.java:304)

Remove decommissioned servers from rsgroup.
Dead/recovering/live servers will be disallowed.
Example:
  hbase> remove_servers_rsgroup ['server1:port','server2:port']

Took 0.9030 seconds
```

Hive on Spark parameters

Konstantin Alpashkin

Contents

To Table of Contents

Remote Spark Driver

This section describes the parameters used to control the interaction between Spark and Hive Metastore. For more information on using the parameters, see the Spark and Hive section.

Parameter	Description	Default value
hive.spark.job.monitor.timeout	The timeout for the job monitor to get a Spark job state (in seconds)	60
hive.spark.dynamic.partition.pruning	When set to `true`, enables dynamic partition pruning (DPP) for the Spark engine so that joins on partition keys are processed by writing to a temporary HDFS file, and read later for removing unnecessary partitions	false
hive.spark.dynamic.partition.pruning.map.join.only	Similar to `hive.spark.dynamic.partition.pruning`, but enables DPP only if the join on the partitioned table can be converted to a MapJoin	false
hive.spark.dynamic.partition.pruning.max.data.size	The maximum data size for the dimension table that generates partition pruning information (in megabytes). If a table reaches this limit, the optimization is disabled	100
hive.spark.exec.inplace.progress	Updates Spark job execution progress in-place in the terminal	true
hive.spark.use.ts.stats.for.mapjoin	If set to `true`, the MapJoin optimization in Hive/Spark uses statistics from TableScan operators at the root of the operator tree, instead of parent ReduceSink operators of the Join operator	false
hive.spark.explain.user	Defines whether to show `EXPLAIN` result at the user level for Hive-on-Spark queries. When enabled, logs `EXPLAIN` output for the query at the user level	false
hive.prewarm.spark.timeout	Time to wait to finish pre-warming Spark executors when `hive.prewarm.enabled=true` (in milliseconds)	5000
hive.spark.optimize.shuffle.serde	If set to `true`, Hive on Spark registers custom serializers for data types in shuffle. This should result in less shuffled data	false
hive.merge.sparkfiles	Merges small files at the end of a Spark DAG Transformation	false
hive.spark.use.op.stats	Defines whether to use operator stats to determine reducer parallelism for Hive on Spark. If set to `false`, Hive will use source table stats to determine reducer parallelism for all first level reduce tasks, and the maximum reducer parallelism from all parents for all the rest (second level and onward) reducer tasks. Setting to `false` triggers an alternative algorithm for calculating the number of partitions per Spark shuffle. This new algorithm typically results in an increased number of partitions per shuffle	true
hive.spark.use.ts.stats.for.mapjoin	If set to `true`, MapJoin optimization in Hive/Spark will use statistics from TableScan operators at the root of operator tree, instead of parent ReduceSink operators of the Join operator. Setting this to `true` is useful when the operator statistics used for a common join-to-map join conversion are inaccurate	false
hive.spark.use.groupby.shuffle	When set to `true`, uses Spark's `RDD#groupByKey()` to perform groupings. When set to false, use Spark's `RDD#repartitionAndSortWithinPartitions()` to perform groupings. While `groupByKey()` has better performance when running grouping, it can use an excessive amount of memory. Setting this to `false` may reduce memory usage, but will affect the performance	true

Remote Spark Driver

The remote Spark driver is the application launched in the Spark cluster that submits actual Spark jobs. It is a long-lived application initialized upon the first query of the current user and is running until the user session is closed.

The following properties control the remote communication between the remote Spark driver and the Hive client that spawns it.

Parameter	Description	Default value
hive.spark.client.future.timeout	Timeout for requests from the Hive client to the remote Spark driver (in seconds)	60
hive.spark.client.connect.timeout	Timeout for the remote Spark driver to connect back to the Hive client (in milliseconds)	1000
hive.spark.client.server.connect.timeout	Timeout for the handshake between the Hive client and the remote Spark driver (in milliseconds). Checked by both processes	90000
hive.spark.client.secret.bits	Number of bits of randomness in the generated secret for communication between the Hive client and the remote Spark driver. Rounded down to the nearest multiple of 8	256
hive.spark.client.rpc.server.address	The server address of the HiveServer2 host to be used for communication between the Hive client and the remote Spark driver	hive.spark.client.rpc.server.address, `localhost` if unavailable
hive.spark.client.rpc.threads	The maximum number of threads for the remote Spark driver's RPC event loop	8
hive.spark.client.rpc.max.size	The maximum message size in bytes for communication between the Hive client and the remote Spark driver	52,428,800 (50 * 1024 * 1024, or 50 MB)
hive.spark.client.channel.log.level	Channel logging level for remote Spark driver. Possible values: `DEBUG`, `ERROR`, `INFO`, `TRACE`, `WARN`. If unset, `TRACE` is used	—

Beeline command line

Mikhail Serov

Contents

To Table of Contents

Beeline Hive commands
Beeline command options

Beeline commands

!<SQLLine_command>	Executes an SQLLine command
!delimiter	Sets the delimiter for queries written in Beeline. Multi-character delimiters are allowed, but quotation marks, slashes, and dashes (`--`) are not allowed. Defaults to `;`. Usage: `!delimiter $$` Version: 3.0.0

Beeline properties

fetchsize	The JDBC standard allows you to specify the number of rows fetched with each database round-trip for a query, and this number is referred to as the fetch size. Setting the fetch size in Beeline overrides the JDBC driver's default fetch size and affects subsequent statements executed in the current session. Setting the value to `-1` instructs Beeline to use the JDBC driver's default fetch size (default). Setting the value to `0` or any positive integer is passed to the JDBC driver for each statement. Any other negative value will throw an exception. Usage: `!set fetchsize 200` Version: 4.0.0

Beeline Hive commands

Hive-specific commands (same as Hive CLI commands) can run from Beeline, when the Hive JDBC driver is used. Use `;` (semicolon) to terminate commands. Comments in scripts can be specified using the `--` prefix.

Beeline Hive commands

reset	Resets the configuration to the default values
reset <key>	Resets the value of a particular configuration variable (key) to the default value If you misspell a key name, Beeline does not show an error.
set <key>=<value>	Sets the value of a particular configuration variable (key) If you misspell a variable name, Beeline does not show an error.
set	Prints a list of configuration variables that are overridden by the user or Hive
set -v	Prints all Hadoop and Hive configuration variables
*add FILE[S] <filepath> <filepath>** *add JAR[S] <filepath> <filepath>** *add ARCHIVE[S] <filepath> <filepath>**	Adds one or more files, JARs, or archives to the list of resources in the distributed cache. For more information, see Hive Resources page of the Apache Hive documentation
*add FILE[S] <ivyurl> <ivyurl>** *add JAR[S] <ivyurl> <ivyurl>** *add ARCHIVE[S] <ivyurl> <ivyurl>**	As of Hive 1.2.0, adds one or more files, JARs, or archives to the list of resources in the distributed cache using an Ivy URL of the form *ivy://group:module:version?query_string*. For more information, see Hive Resources page of the Apache Hive documentation
list FILE[S] *list JAR[S]* *list ARCHIVE[S]*	Lists the resources already added to the distributed cache. For more information, see Hive Resources page of the Apache Hive documentation
*list FILE[S] <filepath>** *list JAR[S] <filepath>** *list ARCHIVE[S] <filepath>**	Checks whether the given resources are already added to the distributed cache or not. For more information, see Hive Resources page of the Apache Hive documentation
*delete FILE[S] <filepath>** *delete JAR[S] <filepath>** *delete ARCHIVE[S] <filepath>**	Removes the resource(s) from the distributed cache
*delete FILE[S] <ivyurl> <ivyurl>** *delete JAR[S] <ivyurl> <ivyurl>** *delete ARCHIVE[S] <ivyurl> <ivyurl>**	As of Hive 1.2.0, removes the resource(s) which were added using the <ivyurl> from the distributed cache. See Hive Resources page of the Apache Hive documentation for more information
reload	As of Hive 0.14.0, makes HiveServer2 aware of any JAR changes in the path specified by the configuration parameter hive.reloadable.aux.jars.path (without having to restart HiveServer2). The changes can be adding, removing, or updating JAR files
dfs <dfs command>	Executes a `dfs` command
<query string>	Executes a Hive query and prints results to standard output

Beeline command options

The Beeline CLI supports these command line options:

Option	Description	Usage example
-u <database URL>	The JDBC URL to connect to. Special characters in parameter values should be encoded with URL encoding if needed	`$ beeline -u db_URL`
-r	Reconnects to the last used URL (if you have previously used `!connect` and `!save` commands). Version: 2.1.0	`$ beeline -r`
-n <username>	The username to be used for connection	`$ beeline -n <valid_user>`
-p <password>	The password to be used for connection. Optional password mode: Starting from Hive 2.2.0, the argument for `-p` option is optional. If the password is not provided after `-p`, Beeline will prompt for the password while initiating the connection. When password is provided, Beeline uses it to initiate the connection without prompting	`$ beeline -p <valid_password>`
-d <driver class>	The driver class to use	`$ beeline -d <driver_class>`
-e <query>	Specifies the query to be executed. Double or single quotes enclose the query string. This option can be specified multiple times	`$ beeline -e "query_string"`
-f <file>	Specifies the path to a script file to be executed. Version: 0.12.0 If the script contains tabs, query compilation fails in version 0.12.0. This bug is fixed in version 0.13.0	`$ beeline -f <path/to/file>`
-i <file> --init <files>	Sets the init file(s) for initialization. Version: 0.14.0: single file definition introduced. Version: 2.1.0: multiple files can be specified	`$ beeline -i <path/to/initfile>`
-w <path/to/password_file> --password-file <path/to/password file>	Specifies the password file to read password from. Version: 1.2.0	`$ beeline -u '<JDBC_URL>' -n test_user -w pass`
-a <auth type> --authType <auth_type>	Sets the authentication type passed to the JDBC as an `auth` property. Version: 0.13.0	`$ beeline --authType 'kerberos'`
--property-file <file>	Specifies the file to read configuration properties from. Version: 2.2.0	`$ beeline --property-file <path/to/file>`
--hiveconf property=value	Sets a value for the given configuration property. Properties that are listed in `hive.conf.restricted.list` cannot be reset with `hiveconf` (see Restricted List and Whitelist page of the Apache Hive documentation). Version: 0.13.0	`$ beeline --hiveconf prop1=value1`
--hivevar name=value	Sets a Hive variable name and value. This is a Hive-specific setting in which variables can be set at the session level and referenced in Hive commands or queries	`$ beeline --hivevar var1=value1`
--color=[true/false]	Controls whether color is used for display. Default is `false`. Not supported for Separated-Value Output formats	`$ beeline --color=true`
--showHeader=[true/false]	Shows column names in the query results (`true`) or not (`false`). Default is `true`	`$ beeline --showHeader=false`
--headerInterval=ROWS	The interval for re-displaying column headers, in number of rows, when `outputformat=table`. Default is 100. Not supported for Separated-Value Output formats	`$ beeline --headerInterval=50`
--fastConnect=[true/false]	Controls if the building of a list of all tables and columns for tab-completion of HiveQL statements is skipped (`true`) or not (`false`). Default is `true`	`$ beeline --fastConnect=false`
--autoCommit=[true/false]	Enables/disables automatic transaction commit. Default is `false`	`$ beeline --autoCommit=true`
--verbose=[true/false]	Shows verbose error messages and debug information (`true`) or does not show those messages and information (`false`). Default is `false`	`$ beeline --verbose=true`
--showWarnings=[true/false]	Displays warnings that are reported on the connection after issuing any HiveQL commands. Default is `false`	`$ beeline --showWarnings=true`
--showDbInPrompt=[true/false]	Displays the current database name in prompt. Default is `false` Version: 2.2.0	`$ beeline --showDbInPrompt=true`
--showNestedErrs=[true/false]	Displays nested errors. Default is `false`	`$ beeline --showNestedErrs=true`
--numberFormat=[pattern]	Formats numbers using the DecimalFormat pattern	`$ beeline --numberFormat="#,###,##0.00"`
--force=[true/false]	Continues to run the script even after errors (`true`) or do not continue (`false`). Default is `false`	`$ beeline--force=true`
--maxWidth=MAXWIDTH	Shows the maximum width to display before truncating data, in characters, when `outputformat=table`. Default is to query the terminal for current width, then fall back to 80	`$ beeline --maxWidth=150`
--maxColumnWidth=MAXCOLWIDTH	Shows the maximum column width, in characters, when `outputformat=table`. Default is 50 in Hive version 2.2.0+ or 15 in earlier versions	`$ beeline --maxColumnWidth=25`
--silent=[true/false]	Reduces the amount of informational messages displayed (`true`) or not (`false`). It also stops displaying the log messages for the query from HiveServer2 (Hive 0.14 and later) and the HiveQL commands (Hive 1.2.0 and later). Default is `false`	`$ beeline --silent=true`
--autosave=[true/false]	Automatically saves preferences (`true`) or not (`false`). Default is `false`	`$ beeline --autosave=true`
--outputformat=[table/vertical/csv/tsv/dsv/csv2/tsv2]	Sets the format mode for the result display. Default is table. See Separated-value output formats for description of recommended options. Version 0.14.0: added dsv/csv2/tsv2	`$ beeline --outputformat=tsv`
--truncateTable=[true/false]	If set to `true`, truncates table column in the console when it exceeds console length. Version: 0.14.0	`$ beeline --truncateTable=true`
--delimiterForDSV=DELIMITER	Sets the delimiter for delimiter-separated values output format. Default is `|` character. Version: 0.14.0	`$ beeline --delimiterForDSV='*'`
--isolation=LEVEL	Sets the transaction isolation level to `TRANSACTION_READ_COMMITTED` or `TRANSACTION_SERIALIZABLE`. For more information, see Java Connection Interface documentation	`$ beeline --isolation=TRANSACTION_SERIALIZABLE`
--nullemptystring=[true/false]	Sets for use the historic behavior of printing null as empty string (`true`) or the current behavior of printing null as NULL (`false`). Default is `false`. Version: 0.13.0	`$ beeline --nullemptystring=false`
--incremental=[true/false]	Defaults to `true` starting from Hive 2.3 onwards, before it defaulted to `false`. When set to `false`, the entire result set is fetched and buffered before being displayed, yielding optimal display column sizing. When set to `true`, result rows are displayed immediately as they are fetched, yielding lower latency and memory usage at the price of extra display column padding. Setting `--incremental=true` is recommended if you encounter an OutOfMemoryError on the client side (due to the fetched result set size being large)	`$ beeline --incremental=false`
--incrementalBufferRows=NUMROWS	Sets the number of rows to buffer when printing rows on stdout, defaults to 1000; only applicable if `--incremental=true` and `--outputformat=table`. Version: 2.3.0	`$ beeline --incrementalBufferRows=1000`
--maxHistoryRows=NUMROWS	Sets the maximum number of rows to store Beeline history. Version: 2.3.0	`$ beeline --maxHistoryRows=1000`
--delimiter=;	Sets the delimiter for queries written in Beeline. Multi-char delimiters are allowed, but quotation marks, slashes, and `--` are not allowed. Defaults to `;`. Version: 3.0.0	`$ beeline beeline --delimiter=$$`
--convertBinaryArrayToString=[true/false]	Permits to display binary column data as a string using the platform's default character set. Version: 3.0.0: The default behavior (`false`) is to display binary data using: `Arrays.toString(byte[] columnValue)` Displays binary column data as a string using the UTF-8 character set. Version: 4.0.0: The default behavior (`false`) is to display binary data using Base64 encoding without padding	`$ beeline --convertBinaryArrayToString=true`
--help	Displays usage tips	`$ beeline --help`

Impala shell commands

Eugenia Kuzina

To Table of Contents

You can use the commands described below to pass requests to an impalad daemon. It is possible to enter a command to the prompt, or pass it as the argument to the `-q` option of impala-shell. Most of these commands are passed to the Impala daemon as SQL statements. Refer to the corresponding sections of the Impala SQL Statements article for syntax details.

Impala shell commands

Name	Description
alter	Changes the structure or settings of an Impala table, or a table shared between Impala and Hive. See ALTER TABLE and ALTER VIEW statements for details
compute stats	Displays performance-related information for a table. Impala uses it to optimize queries. See COMPUTE STATS Statement
connect	Connects to an instance of impalad. The command uses the default port `21000` for connections. If you connect to an instance of impalad that works on a different port, specify this port as a parameter. See Use impala-shell to connect to Impala
describe	Shows the column names, data types, and comments for a specified table. Also, the `describe formatted` command displays additional information such as the HDFS data directory, partitions, and internal properties for the table. You can use the abbreviation `desc` for the `describe` statement. See DESCRIBE Statement
drop	Removes an object, and in some cases its associated data files. See DROP TABLE, DROP VIEW, DROP DATABASE, and DROP FUNCTION statements for more information
explain	Provides the execution plan for a query. `explain` represents a query as a series of steps. These steps might be map/reduce stages, metastore operations, or file system operations (move or rename). See EXPLAIN Statement and Using the EXPLAIN Plan for Performance Tuning
help	Displays a list of all available commands and options
history	Shows an enumerated command history list from recent sessions. The history is stored in the *~/.impalahistory* file
insert	Writes a query result to a specified table. This statement either overwrites table data or appends data to the existing table content. See INSERT Statement
invalidate metadata	Updates impalad metadata for tables. Use this command after creating, dropping, or altering databases, tables, or partitions in Hive. See INVALIDATE METADATA Statement
profile	Displays low-level information about the most recent query. It is used for performance diagnosis and tuning. The report starts with the same information as the `EXPLAIN` and `SUMMARY` reports. See Using the Query Profile for Performance Tuning
quit	Exits the impala-shell. It is necessary to add a final semicolon to the command so that the shell recognizes the end of the command
refresh	Refreshes impalad metadata of the HDFS block locations corresponding to Impala data files. Utilize this command after loading new data files into an Impala table through Hive or through HDFS commands. See REFRESH Statement
rerun or @	Executes a previous impala-shell command from the list of commands displayed by the history command. It can be an SQL statement, or a command specific to impala-shell (`quit` or `profile`). The command accepts an integer argument. A positive integer number represents the command number from the `history` command output list. A negative integer `-N` represents the `Nth` command from the end of the list. For example, `-1` executes the most recent command. Commands that are executed again do not produce new entries in the `history` output list
select	Selects data from a database. The returned result-set can be sent to the console output, or a file, or can be a part of another query. See SELECT Statement
set	Specifies query options for an impala-shell session. The available options are listed at Query Options for the SET Statement. These options are used for query tuning and troubleshooting. The command syntax is `set <option>=<value>`. If you need to display the current query options, run `set` without arguments. To modify an option value, run the command with a new value. To restore an option to its default, use the `unset` command. The `set` statement has no effect until impala-shell is connected to an impalad instance. You can also use the `set` as an SQL statement. See SET Statement
shell	Allows you to execute a command in the operating system shell without exiting from impala-shell. You can use the `!` character as shorthand for the `shell` command. To embed comments within the `source` or `!` commands, use the shell comment character `#` before a commented line
show	Displays metastore data for objects (databases, tables, and functions) created and accessed through Impala or Hive. See SHOW Statement
source or src	The command accepts as a parameter the path to a file on the local file system and executes statements from this file. The file can contain SQL statements and impala-shell commands, including other `source` commands. The `source` command allows you to perform batch operations in the same way as impala-shell with the `-f` option, but in interactive mode. Each command or statement in the file, except the last one, must end with a semicolon. See Running Commands and SQL Statements in impala-shell
summary	Displays a summary of the operations performed at different stages of the query. It provides a higher-level view of the information displayed by the `explain` command. See Using the SUMMARY Report for Performance Tuning. You can also see a continuously updated report of the summary information while a query is in progress. See LIVE_SUMMARY Query Option
unset	Removes any user-specified value for a query option and returns the option to its default value. For information about the options for which this command is available, refer to Query Options for the SET Statement. You can also remove substitution variables. For this, use the following notation: `unset var:<variable_name>`
use	Specifies the database against which to execute subsequent commands. The `use` command allows you to avoid utilizing fully qualified names when you need to access a database other than the default one. See USE Statement
version	Displays Impala version information

MapReduce CLI

Konstantin Alpashkin

Contents

To Table of Contents

User commands
Administration commands

All MapReduce commands are called by the *bin/mapred* script. Running the *mapred* script without any arguments prints the description for all commands.

The script usage is as follows:

```
$ mapred [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [COMMAND_OPTIONS]
```

Hadoop has an option parsing framework that employs parsing generic options as well as running classes.

MapReduce shell basic structure

Command options	Description
SHELL_OPTIONS	The common set of shell options
GENERIC_OPTIONS	The common set of options supported by multiple commands
COMMAND_OPTIONS	Various commands with their options

User commands

Command	Description
archive	Creates a Hadoop archive
archives-logs	Combines YARN logs into Hadoop archives
classpath	Prints the classpath
distcp	Used for inter/intra-cluster copying of large files and directories
job	Used to work with MapReduce jobs
pipes	Runs a pipe job
queue	Allows users to interact with queues and to view job queue information
version	Prints the version
envvars	Displays computed Hadoop environment variables

Administration commands

Command	Description
historyserver	Starts JobHistoryServer
hsadmin	Runs a MapReduce HSAdmin client
frameworkuploader	Collects framework JARs

archive

Konstantin Alpashkin

To Table of Contents

Allows to create Hadoop archives (.*har*).

The tool usage is as follows:

```
$ hadoop archive -archiveName name -p <parent> [-r <replication_factor>] <src>* <dest>
```

Arguments

-archiveName <name>	A name for the archive
-p <parent>	Specifies the relative path to which the files should be archived
-r <replication_factor>	Indicates the desired replication factor
<src>	Source directories to include into archive
<dest>	The destination path to store the archive

archive-logs

Konstantin Alpashkin

To Table of Contents

For clusters with a lot of aggregated YARN logs, you can combine logs into Hadoop archives to reduce the number of small files, and hence the NameNode load. This command provides an easy way to do this.

You can read aggregated logs in Hadoop archives by using the Job History Server and the YARN `logs` command.

The tool usage is as follows:

```
$ mapred archive-logs
```

Arguments

-force	Forces re-creating the working directory if an existing one is found. This should only be used if you know that another instance is not currently running
-maxEligibleApps <n>	The maximum number of eligible applications to process Defaults to `-1` (all)
-maxTotalLogsSize <megabytes>	The maximum total logs size required to be eligible for archival (in megabytes). Defaults to `1024`
-memory <megabytes>	The amount of memory for each container (in megabytes). Defaults to `1024`
-minNumberLogFiles <n>	The minimum number of log files required to be eligible for archival. Defaults to `20`
-noProxy	When specified, all processing is done under the user running this command (or the YARN user if DefaultContainerExecutor is in use). If not specified, all processing is done under the user who owns that application; if the user running this command is not allowed to impersonate that user, it will fail
-help	Prints this help message

classpath

Konstantin Alpashkin

To Table of Contents

Prints the classpath needed to get the Hadoop JAR and the required libraries. If called without arguments, prints the classpath set up by the command scripts, which is likely to contain wildcards in the classpath entries. Additional options print the classpath after wildcard expansion or write the classpath into the manifest of a JAR file. The latter is useful in environments where wildcards cannot be used and the expanded classpath exceeds the maximum supported command line length.

The tool usage is as follows:

```
$ mapred classpath [--glob |--jar <path> |-h |--help]
```

Arguments

--glob	Expands wildcards
--jar <path/to/.jar>	Writes the classpath string into the manifest of the JAR named `<path/to/.jar>`
--help	Prints usage options

distcp

Konstantin Alpashkin

To Table of Contents

The tool is used for inter/intra-cluster copying of files and directories.

The tool usage is as follows:

```
$ mapred distcp <src> <dst> [args]
```

Arguments

-append	Allows to reuse existing data in destination files and append new data to them if possible
-async	Runs the `distcp` command asynchronously. Quits as soon as the Hadoop job is started
-atomic	Instructs `distcp` to copy source data to a temporary target location, and then move the temporary target to the final location atomically
-bandwidth <arg>	Specifies a bandwidth per map (in MB/second)
-blocksperchunk <arg>	The number of blocks per chunk. When specified, splits files into chunks to copy in parallel
-copybuffersize	The size of the copy buffer to use (in bytes). Defaults to `8192` bytes
-delete	Deletes files existing in `<dst>` but not in `<src>`
-diff <oldSnapshot> <newSnapshot>	Allows to identify the difference between source and target, and apply the diff to the target to make it in sync with source
-f <urilist_uri>	Specifies a path to a file with a list of URIs to be copied
-filelimit <n>	Limits the total number of files to copy to be <= `n`
-filters	The path to a file containing a list of pattern strings, one string per line, to exclude paths that match the pattern from the copy
-i	Ignores failures
-log <path/to/logdir>	Saves logs to *<path/to/logdir>*
-m	Defines the maximum number of simultaneous copies
-numListstatusThreads	The number of threads to use for building file listings
-overwrite	If provided, overwrites the destination
-p <arg>	Preserve status (replication, block-size, user, group, permission, checksum-type, ACL, XATTR, timestamps). If `-p` is specified with no `<arg>`, then preserves the replication, block size, user, group, permission, checksum type and timestamps. XATTRs are preserved when both the source and destination paths are in the */.reserved/raw* hierarchy (HDFS only)
-rdiff <newSnapshot> <oldSnapshot>	Allows to identify the changes on the target since `<oldSnapshot>` was created on the target, and apply the diff reversely to the target, and copy modified files from the source's `<oldSnapshot>`, to make the target the same as `<oldSnapshot>`
-sizelimit <n>	Deprecated. Limits the total size to be <= `n` (in bytes)
-skipcrccheck	Defines whether to skip CRC checks for source and target paths
-strategy <arg>	The copy strategy to be used in `distcp`. Possible values are `dynamic` and `uniformsize`
-tmp <path/to/dir>	An intermediate work path to be used for atomic commits
-update	Updates the target, copying only missing files or directories
-v	Logs additional info (path, size) in the SKIP/COPY log
-xtrack <path>	Saves information about missing source files to the specified `<path>`

Example:

```
$ mapred distcp hdfs://nn1:8020/foo/bar hdfs://nn2:8020/bar/foo
```

envvars

Konstantin Alpashkin

The command prints computed Hadoop environment variables.

The usage is as follows:

```
$ mapred envvars
```

job

Konstantin Alpashkin

To Table of Contents

The command is used to interact with MapReduce jobs.

The usage is as follows:

```
$ mapred job | [GENERIC_OPTIONS] | [-submit <job-file>] | [-status <job-id>] | [-counter <job-id> <group-name> <counter-name>] | [-kill <job-id>] | [-events <job-id> <from-event-#> <#-of-events>] | [-history [all] <jobHistoryFile|jobId> [-outfile <file>] [-format <human|json>]] | [-list [all]] | [-kill-task <task-id>] | [-fail-task <task-id>] | [-set-priority <job-id> <priority>] | [-list-active-trackers] | [-list-blacklisted-trackers] | [-list-attempt-ids <job-id> <task-type> <task-state>] [-logs <job-id> <task-attempt-id>] [-config <job-id> <file>]
```

Arguments

-submit <job-file>	Submits a job
-status <job-id>	Prints the map and reduce completion percentage and all job counters
-counter <job-id> | <group-name> | <counter-name>	Prints the counter value
-kill <job-id>	Kills a job
-events <job-id> | <from-event-#> | <#-of-events>	Prints events details received by jobtracker for the given range
-history [all] jobHistoryFilejobId [-outfile <file>] [-format human|json]	Prints job details, failed and killed task details. The `all` flag allows to view more details about a job, such as successful tasks, task attempts made for each task, task counters, and so on. `-outfile <file>` allows to direct the output to a file (instead of stdout). The format defaults to human-readable but can also be changed to JSON using the `-format` flag
-list [all]	Displays jobs which are not complete. `-list all` displays all jobs
-kill-task <task-id>	Kills a task. Killed tasks are NOT counted as failed attempts
-fail-task <task-id>	Fails a task. Failed tasks are counted as failed attempts
-set-priority <job-id> <priority>	Changes the priority of a job. Supported priority values are `VERY_HIGH`, `HIGH`, `NORMAL`, `LOW`, `VERY_LOW`
-list-active-trackers	Lists all the active NodeManagers in the cluster
-list-blacklisted-trackers	Lists the blacklisted task trackers in the cluster. This command is not supported in MRv2-based clusters
list-attempt-ids <job-id> <task-type> <task-state>	Lists the attempt IDs based on the provided task type and status. Valid task types are `REDUCE`, `MAP`. Valid task states are `running`, `pending`, `completed`, `failed`, `killed`
-logs <job-id> <task-attempt-id>	Dumps the container log for a job if `<task-attempt-id>` is not provided, otherwise dumps the log for the task with the specified `<task-attempt-id>`. The logs are dumped to stdout
-config <job-id> <file>	Downloads the job configuration file

Example:

```
$ mapred job -status job_141643275736_2152
```

pipes

Konstantin Alpashkin

To Table of Contents

The command is used to run a pipes job.

The usage is as follows:

```
$ mapred pipes [-conf <path>] [-jobconf <key=value>, <key=value>, ...] [-input <path>] [-output <path>] [-jar <jar file>] [-inputformat <class>] [-map <class>] [-partitioner <class>] [-reduce <class>] [-writer <class>] [-program <executable>] [-reduces <num>]
```

Arguments

-conf <path>	The path to a job configuration file
-jobconf <key=value,key=value,...>	Adds/overrides configuration properties for a job
-input <path>	The path to the input directory
-output <path>	The path to the output directory
-jar <jar file>	Specifies the JAR file name
-inputformat <class>	The name of the InputFormat class
-map <class>	The name of the Map class
-partitioner <class>	The name of the Partitioner class
-reduce <class>	The name of the Reducer class
-writer <class>	The name of the RecordWriter class
-program <executable>	The URI to the executable
-reduces <num>	The number of reduces

Example:

```
$ mapred pipes -conf job_463823057_6457
```

queue

Konstantin Alpashkin

To Table of Contents

The command allows to interact with queues and view job queue information.

The usage is as follows:

```
$ mapred queue [-list] | [-info <job-queue-name> [-showJobs]] | [-showacls]
```

Arguments

-list	Returns a list of job queues configured in the system along with scheduling information
-info <job-queue-name> [-showJobs]	Displays the job queue information and associated scheduling information of a particular job queue. If the `-showJobs` flag is provided, a list of jobs submitted to the particular job queue is displayed
-showacls	Displays the queue name and associated queue operations allowed for the current user. The list includes only those queues to which the user has access

Example:

```
$ mapred queue -list
```

version

Konstantin Alpashkin

To Table of Contents

The command prints the current Hadoop version.

The usage is as follows:

```
$ mapred version
```

frameworkuploader

Konstantin Alpashkin

To Table of Contents

The command collects framework JARs and uploads them to HDFS as a tarball.

The usage is as follows:

```
$ mapred frameworkuploader -target <target> args
```

Arguments

-input <classpath>	The input classpath that is searched for JAR files to be included in the tarball
-fs <filesystem>	The target file system. Defaults to the default filesystem set by `fs.defaultFS`
-target <target>	The target location to store the framework tarball, optionally followed by a `#` with the localized alias. An example is `/usr/lib/framework.tar#framework`. Make sure the target directory is readable by all users and is not writable by others than administrators to ensure cluster security
-blacklist <list>	Specifies a comma-separated array of regexps to filter JAR file names to exclude from the classpath. For example, this can be used to exclude test JARs or Hadoop services that are not necessary to localize
-whitelist <list>	Specifies a comma-separated array of regexps to include specific JAR files. This can be used to provide additional security, so that no external source can include malicious code in the classpath when the tool runs
-nosymlink	Used to exclude symlinks that point to the same directory. For example, the path */a/foo.jar* and a symlink */a/bar.jar* that points to */a/foo.jar* would normally add *foo.jar* and *bar.jar* to the tarball as separate files despite them actually being the same file. This flag would make the tool exclude */a/bar.jar* so only one copy of the file is added
-initialReplication <num>	The replication number that is used to create the framework tarball. It is safe to leave this value as the default `3`
-finalReplication <num>	The uploader tool sets the replication factor once all blocks are collected and uploaded. If a quick initial startup is required, then it is recommended to set this value equal to the number of commissioned nodes divided by two but less than `512`
-acceptableReplication <num>	The tool waits until the tarball gets replicated by this number of times before exiting. The value should be a replication number less than or equal to the value of `finalReplication`. An optimal value is 90% of the `finalReplication` value to accommodate failing nodes
-timeout <seconds>	A timeout to wait to reach `acceptableReplication` before the tool exits (in seconds). The tool logs an error otherwise and returns

Example:

```
$ mapred frameworkuploader -timeout 10
```

historyserver

Konstantin Alpashkin

To Table of Contents

The command starts JobHistoryServer.

The usage is as follows:

```
$ mapred historyserver
```

hsadmin

Konstantin Alpashkin

To Table of Contents

The command runs a MapReduce HSAdmin client to execute JobHistoryServer administrative commands.

The usage is as follows:

```
$ mapred hsadmin args
```

Arguments

-refreshUserToGroupsMappings	Refreshes user-to-groups mappings
-refreshSuperUserGroupsConfiguration	Refreshes superuser proxy groups mappings
-refreshAdminAcls	Refreshes ACLs for administration of Job History server
-refreshLoadedJobCache	Refreshes the loaded job cache of Job History server
-refreshJobRetentionSettings	Refreshes the Job History period and job cleaner settings
-refreshLogRetentionSettings	Refreshes the log retention period and log retention check interval
-getGroups <username>	Returns groups to which the `<username>` user belongs
-help <cmd>	Displays help for the `<cmd>` command or all commands if none is specified

Example:

```
$ mapred hsadmin -refreshLogRetentionSettings
```

Ozone CLI

Sergei Tikhomirov

Contents

To Table of Contents

- Overview
- Bucket commands
- Key commands
- Prefix commands
- Snapshot commands
- Token commands
- User commands
- Volume commands

Overview

The general way to invoke an Ozone shell command is as follows:

```
$ ozone sh <object> <action> [<params>]
```

where:

- `<object>` is an Ozone object such as bucket, key, prefix, snapshot, user, token, or volume.
- `<action>` is an action to perform with an object. For example, to create a volume, the action would be `create`.
- `[params]` are additional parameters that some actions require.

For examples of how to use the CLI tool, refer to the Connect to Ozone via CLI article.

Bucket commands

These commands allow you to administer buckets within a volume.

Command	Description
addacl	Adds ACLs to a bucket
clrquota	Clears bucket quota
create	Creates a new bucket
delete	Deletes a bucket
getacl	Returns all ACLs from a bucket
info	Prints information about a bucket
link	Creates a symlink between buckets
list	Prints information about all buckets in a volume
removeacl	Removes ACLs from a bucket
set-replication-config	Sets replication configuration for a bucket
setacl	Overwrites bucket's ACLs
setquota	Sets quotas for an existing bucket
update	Updates the owner of a bucket

Key commands

These commands allow you to administer keys within a bucket.

Command	Description
addacl	Adds ACLs to a key
cat	Prints contents of a specific Ozone key to standard output
checksum	Returns checksum information of a key
cp	Copies an existing key to another one within the same bucket
delete	Deletes a key
get	Downloads contents of a key from a server
getacl	Lists all ACLs for a given key
info	Prints information about a key
list	Lists information about all keys in a given volume, bucket, or snapshot
put	Adds a key with file contents to a bucket
removeacl	Removes an access control list from a key
rename	Renames an existing key
setacl	Sets one or more ACLs, replacing the existing ones

Prefix commands

These commands allow you to manage prefix access settings.

Command	Description
addacl	Adds an access control list for a prefix
getacl	Lists all ACLs for a given prefix
removeacl	Removes an access control list from a prefix
setacl	Sets one or more ACLs, replacing the existing ones

Snapshot commands

These commands allow you to work with bucket snapshots.

Command	Description
create	Creates a snapshot of a bucket
delete	Deletes a snapshot of a bucket
diff	Returns a difference between two snapshots
info	Prints information about a bucket
list	Lists all snapshots of a bucket
listDiff	Lists the SnapshotDiff jobs for a bucket

Token commands

These commands allow you to administer delegation tokens.

Command	Description
cancel	Cancels a delegation token
get	Issues a delegation token and saves it's encoded version to a file
print	Prints information about a delegation token
renew	Renews a delegation token

User commands

These commands allow you to administer users.

Command	Description
assign-admin	Assigns an admin role to an access ID
assign	Assigns a user to a tenant (organization)
get-secret	Returns a secret given to an access ID
info	Prints information about a user
list	Lists all users in a tenant
revoke-admin	Revokes admin privileges from a user
revoke	Removes a user from an organization (tenant)
set-secret	Sets a secret for an access ID

Volume commands

These commands allow you to administer volumes.

Command	Description
addacl	Adds ACLs to a volume
clrquota	Clears volume quota
create	Creates a new volume
delete	Deletes a volume
getacl	Returns all ACLs from a volume
info	Prints information about a volume
list	Prints information about all volumes
removeacl	Removes ACLs from a volume
setacl	Overwrites volume's ACLs
setquota	Sets quotas for an existing volume
update	Updates the owner of a volume

bucket addacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Adds an access control list for a bucket.

Usage

```
$ ozone sh bucket addacl [-hV] [-s=<store>]
                               -a=<list>
                               <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where: ▪ `<type>` is a system entity type like `user` or `group`. ▪ `<entity>` is a user or a group the permissions for which you want to control. ▪ `<permissions>` is a sequence of letters that defines the entity’s permissions. The following values are available: ▪ `r` — read the resource data. ▪ `w` — write data to the resource. ▪ `c` — create objects at the resource. ▪ `d` — delete objects at the resource. ▪ `l` — list objects stored at the resource. ▪ `x` — read the access control list of the resource. ▪ `y` — edit the access control list of the resource. ▪ `a` — all of the above. ▪ `n` — no permissions.
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Adding an ACL:

```
$ ozone sh bucket addacl vol1/bucket1 -s=OZONE -a=user:sergei:all
```

The confirmation output:

```
ACL user:sergei:la[ACCESS] added successfully.
```

bucket clrquota

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Clears quota for a bucket.

Usage

```
$ ozone sh bucket clrquota [-hV] [--namespace-quota]
                               [--space-quota]
                               <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
--namespace-quota	Defines whether to clear the namespace quota (maximum number of keys in a bucket). The new value will be `-1` by default
--space-quota	Defines whether to clear the space quota (maximum number of space that can be used). The new value will be `-1` by default
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Clearing a quota for a bucket:

```
$ ozone sh bucket clrquota --namespace-quota --space-quota vol1/bucket1
```

bucket create

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Creates a new bucket in a given volume.

Usage

```
$ ozone sh bucket create [-ghV] [-k=<bekName>]
                                [-l=<allowedBucketLayout>]
                                [--namespace-quota=<quotaInNamespace>]
                                [--quota=<quotaInBytes>]
                                [-r=<replication>]
                                [-t=<type>]
                                [-u=<ownerName>]
                                <value>
```

Arguments

Parameter	Description
bekName	Bucket encryption key name
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
allowedBucketLayout	A bucket layout. Possible values: `OBJECT_STORE`, `LEGACY`, and the default `FILE_SYSTEM_OPTIMIZED`
quotaInNamespace	Maximum number of keys in a bucket
quotaInBytes	Maximum space quota that can be used (in bytes)
replication	Replication definition. The valid values are based on the replication type. For `RATIS`: `ONE` or `THREE`. In case of `EC`, it should follow the `CODEC-DATA-PARITY-CHUNKSIZE` pattern, e.g. `rs-3-2-1024k`
type	Replication type. The supported types are `RATIS` and `EC`
ownerName	Owner of the bucket. Defaults to the current user if not specified
-g, --enforcegdpr	Indicates to create a GDPR-enforced bucket
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Create a bucket with a specified quota:

```
$ ozone sh bucket create --quota=1073741824 vol1/bucket3
```

Create a bucket with a specified replication config:

```
$ ozone sh bucket create -t=RATIS -r=ONE vol1/bucket4
```

bucket delete

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Deletes a bucket.

Usage

```
$ ozone sh bucket delete [-hrVy] <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-r	Determines wether to delete a bucket recursively
-y, --yes	Determines wether to delete without an interactive user confirmation
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Deleting a bucket:

```
$ ozone sh bucket delete vol1/bucket1
```

bucket getacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists all ACLs for a given bucket.

Usage

```
$ ozone sh bucket getacl [-hV] [--source] [-s=<store>] <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
--source	Specifies wether to display the ACL of a link bucket (`--source=false`) or a source bucket (`--source=true`)
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Listing all ACLs for a source bucket:

```
$ ozone sh bucket getacl --source=true vol1/bucket2
```

The output depends on the present ACLs, but should be similar to this:

```
[ {
  "type" : "USER",
  "name" : "s_tikhomirov_krb1@AD.RANGER-TEST",
  "aclScope" : "ACCESS",
  "aclList" : [ "ALL" ]
}, {
  "type" : "USER",
  "name" : "sergei",
  "aclScope" : "ACCESS",
  "aclList" : [ "LIST", "ALL" ]
} ]
```

bucket info

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Prints information about a bucket.

Usage

```
$ ozone sh bucket info [-hV] <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. To get information about that bucket, call the following command:

```
$ ozone sh bucket info vol1/bucket1
```

The output depends on bucket configuration, but overall should be similar to the following:

```
{
  "metadata" : { },
  "volumeName" : "vol1",
  "name" : "bucket1",
  "storageType" : "DISK",
  "versioning" : false,
  "usedBytes" : 0,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-22T22:06:04.800Z",
  "modificationTime" : "2025-01-22T22:06:55.834Z",
  "sourcePathExist" : true,
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "bucketLayout" : "FILE_SYSTEM_OPTIMIZED",
  "owner" : "s_tikhomirov_krb1",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "link" : false
}
```

bucket link

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Creates a symlink between buckets.

Usage

```
$ ozone sh bucket link [-hV] <source> <target>
```

Arguments

Parameter	Description
source	A source bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
target	A target bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket2/`) or a short URI (starts with a volume name, e.g. `vol1/bucket2`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Creating a symlink between buckets:

```
$ ozone sh bucket link vol1/bucket1 vol1/bucket2
```

bucket list

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Lists information about all buckets in a given volume.

Usage

```
$ ozone sh bucket list [-hV] [--has-snapshot]
                       [-p=<prefix>]
                       [-s=<startItem>]
                       [[-l=<limit>] | [-a]]
                       <value>
```

NOTE

In this command, `list` can be shortened to just `ls`.

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
prefix	A name prefix to filter the items
startItem	The item to start listing from. The chosen item will not appear in the results
limit	Maximum number of the results to list. The default value is `100`
-a, --all	If specified, the command will display all the results. By default, it is specified. To counter it, use the `-l` parameter
--has-snapshot	If specified, the command will display only those buckets that have at least one active snapshot
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with two buckets (`bucket1` and `bucket2`) created within. To list information about all the buckets, call the following command

```
$ ozone sh bucket list vol1
```

The output depends on bucket configuration, but overall should be similar to the following:

```
[ {
  "metadata" : { },
  "volumeName" : "vol1",
  "name" : "bucket1",
  "storageType" : "DISK",
  "versioning" : false,
  "usedBytes" : 0,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-22T22:06:04.800Z",
  "modificationTime" : "2025-01-22T22:06:55.834Z",
  "sourcePathExist" : true,
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "bucketLayout" : "FILE_SYSTEM_OPTIMIZED",
  "owner" : "s_tikhomirov_krb1",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "link" : false
}, {
  "metadata" : { },
  "volumeName" : "vol1",
  "name" : "bucket2",
  "storageType" : "DISK",
  "versioning" : false,
  "usedBytes" : 0,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-22T22:07:32.541Z",
  "modificationTime" : "2025-01-22T22:07:32.541Z",
  "sourcePathExist" : true,
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "bucketLayout" : "FILE_SYSTEM_OPTIMIZED",
  "owner" : "s_tikhomirov_krb1",
  "link" : false
} ]
```

Now, consider that a new bucket `newbucket` was added to `vol1`. To find it among the numerous other buckets using the `list` command, you can utilize the prefix search:

```
$ ozone sh bucket list vol1 -p="new"
```

The output would be:

```
[ {
  "metadata" : { },
  "volumeName" : "vol1",
  "name" : "newbucket",
  "storageType" : "DISK",
  "versioning" : false,
  "usedBytes" : 0,
  "usedNamespace" : 0,
  "creationTime" : "2025-02-12T22:51:39.876Z",
  "modificationTime" : "2025-02-12T22:51:39.876Z",
  "sourcePathExist" : true,
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "bucketLayout" : "FILE_SYSTEM_OPTIMIZED",
  "owner" : "s_tikhomirov_krb1",
  "link" : false
} ]
```

bucket removeacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Removes an access control list from a bucket.

Usage

```
$ ozone sh bucket removeacl [-hV] [-s=<store>]
                            -a=<list>
                            <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where: ▪ `<type>` is a system entity type like `user` or `group`. ▪ `<entity>` is a user or a group the permissions for which you want to control. ▪ `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available: ▪ `r` — read the resource data. ▪ `w` — write data to the resource. ▪ `c` — create objects at the resource. ▪ `d` — delete objects at the resource. ▪ `l` — list objects stored at the resource. ▪ `x` — read the access control list of the resource. ▪ `y` — edit the access control list of the resource. ▪ `a` — all of the above. ▪ `n` — no permissions.
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Removing an ACL from a bucket:

```
$ ozone sh bucket removeacl vol1/newbucket -a=user:sergei:all
```

Confirmation of removal:

```
ACL user:sergei:la[ACCESS] removed successfully.
```

bucket set-replication-config

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Sets the replication configuration for a bucket.

Usage

```
$ ozone sh bucket set-replication-config [-hV] [-r=<replication>]
                                         [-t=<type>]
                                         <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
replication	Replication definition. The valid values are based on the replication type. For `RATIS`: `ONE` or `THREE`. In case of `EC`, it should follow the `CODEC-DATA-PARITY-CHUNKSIZE` pattern, e.g. `rs-3-2-1024k`
type	Replication type. The supported types are `RATIS` and `EC`
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting a new replication configuration for a bucket:

```
$ ozone sh bucket set-replication-config -t=RATIS -r=ONE vol1/bucket2
```

bucket setacl

Sergei Tikhomirov

AD.H

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Sets one or more ACLs, replacing the existing ones.

IMPORTANT

All the existing ACL will be deleted if they are not present in the new ACL list.

Usage

```
$ ozone sh bucket setacl [-hV] [-s=<store>]
                               -a=<list>
                               <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`

list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where:
	▪ `<type>` is a system entity type like `user` or `group`.
	▪ `<entity>` is a user or a group the permissions for which you want to control.
	▪ `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available:
	  ▪ `r` — read the resource data.
	  ▪ `w` — write data to the resource.
	  ▪ `c` — create objects at the resource.
	  ▪ `d` — delete objects at the resource.
	  ▪ `l` — list objects stored at the resource.
	  ▪ `x` — read the access control list of the resource.
	  ▪ `y` — edit the access control list of the resource.
	  ▪ `a` — all of the above.
	  ▪ `n` — no permissions.

-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting an ACL to a bucket:

```
$ ozone sh bucket setacl -s=OZONE -a=user:sergei:all vol1/bucket2
```

The confirmation output:

```
ACLs set successfully.
```

bucket setquota

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Manages storage space quota for a bucket.

Usage

```
$ ozone sh bucket setquota [-hV] [--namespace-quota=<quotaInNamespace>]
                                [--quota=<quotaInBytes>]
                                <value>
```

Arguments

Parameter	Description
quotaInNamespace	Maximum number of keys in a bucket
quotaInBytes	Maximum space quota that can be used (in bytes)
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Specifying quota for a bucket:

```
$ ozone sh bucket setquota --namespace-quota=5 --quota=1073741824 vol1/bucket3
```

bucket update

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Updates the owner of the bucket.

Usage

```
$  ozone sh bucket update [-hV] [-u=<ownerName>] <value>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
ownerName	Name of the new bucket owner
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting a new bucket owner:

```
$ ozone sh bucket update -u=sergei vol1/bucket1
```

As a confirmation, the command outputs the new bucket information:

```
{
  "metadata" : { },
  "volumeName" : "vol1",
  "name" : "bucket1",
  "storageType" : "DISK",
  "versioning" : false,
  "usedBytes" : 0,
  "usedNamespace" : 0,
  "creationTime" : "2025-01-22T22:06:04.800Z",
  "modificationTime" : "2025-02-12T23:58:47.979Z",
  "sourcePathExist" : true,
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "bucketLayout" : "FILE_SYSTEM_OPTIMIZED",
  "owner" : "sergei",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "link" : false
}
```

key addacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Adds an access control list for a key.

Usage

```
$ ozone sh key addacl [-hV] [-s=<store>]
                            -a=<list>
                            <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where: ▪ `<type>` is a system entity type like `user` or `group`. ▪ `<entity>` is a user or a group the permissions for which you want to control. ▪ `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available: ▪ `r` — read the resource data. ▪ `w` — write data to the resource. ▪ `c` — create objects at the resource. ▪ `d` — delete objects at the resource. ▪ `l` — list objects stored at the resource. ▪ `x` — read the access control list of the resource. ▪ `y` — edit the access control list of the resource. ▪ `a` — all of the above. ▪ `n` — no permissions.
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Adding an ACL:

```
$ ozone sh key addacl vol1/bucket1/key1 -s=OZONE -a=user:sergei:all
```

The confirmation output:

```
ACL user:sergei:la[ACCESS] added successfully.
```

key cat

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Prints contents of a specific Ozone key to standard output.

Usage

```
$ ozone sh key cat [-hV] <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket contains a key `key1`. To print the contents of that key, call the following command:

```
$ ozone sh key cat vol1/bucket1/key1
```

key checksum

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Returns checksum information of a key.

Usage

```
$ ozone sh key checksum [-hV] [-c=<mode>]
                        <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
mode	Method of combining chunk checksums. Possible values are `COMPOSITE_CRC` (default) and `MD5MD5CRC`
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket contains a key `key1`. To print the checksum information about that key, call the following command:

```
$ ozone sh key checksum vol1/bucket1/key1
```

The output depends on the key configuration, but overall should be similar to the following:

```
{
  "volumeName" : "vol1",
  "bucketName" : "bucket1",
  "name" : "key1",
  "dataSize" : 14,
  "algorithm" : "COMPOSITE-CRC32",
  "checksum" : "B4B9291A"
}
```

key cp

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Copies an existing key to another one within the same bucket.

Usage

```
$ ozone sh key cp [-hV] [-r=<replication>]
                      [-t=<type>]
                      <value>
                      <fromKey>
                      <toKey>
```

Arguments

Parameter	Description
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
replication	Replication definition. The valid values are based on the replication type. For `RATIS`: `ONE` or `THREE`. In case of `EC`, it should follow the `CODEC-DATA-PARITY-CHUNKSIZE` pattern, e.g. `rs-3-2-1024k`
type	Replication type. The supported types are `RATIS` and `EC`
fromKey	An existing key to be copied
toKey	A name for the new key
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket contains a key `key1`. To copy that key, call the following command:

```
$ ozone sh key cp vol1/bucket1 key1 key2
```

key delete

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Deletes a key.

Usage

```
$ ozone sh key delete [-hV] <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Deleting a key:

```
$ ozone sh key delete vol1/bucket1/key1
```

key get

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Downloads contents of a key from a server.

Usage

```
$ ozone sh key get [-hV] <value>
                       <fileName>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
fileName	Path where to store the contents
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket contains a key `key1`. To download the contents of that key, call the following command:

```
$ ozone sh key get vol1/bucket1/key1 testFile1.txt
```

key getacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists all ACLs for a given key.

Usage

```
$ ozone sh key getacl [-hV] [-s=<store>] <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Listing all ACLs for a key:

```
$ ozone sh key getacl vol1/bucket1/key1
```

The output depends on the present ACLs, but should be similar to this:

```
[ {
  "type" : "USER",
  "name" : "s_tikhomirov_krb1@AD.RANGER-TEST",
  "aclScope" : "ACCESS",
  "aclList" : [ "ALL" ]
}, {
  "type" : "USER",
  "name" : "sergei",
  "aclScope" : "ACCESS",
  "aclList" : [ "LIST", "ALL" ]
} ]
```

key info

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Prints information about a key.

Usage

```
$ ozone sh key info [-hV] <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket contains a key `key1`. To get information about that key, call the following command:

```
$ ozone sh key info vol1/bucket1/key1
```

The output depends on the key configuration, but overall should be similar to the following:

```
{
  "volumeName" : "vol1",
  "bucketName" : "bucket1",
  "name" : "key1",
  "dataSize" : 14,
  "creationTime" : "2025-02-23T21:14:28.098Z",
  "modificationTime" : "2025-02-23T21:17:20.170Z",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "metadata" : { },
  "ozoneKeyLocations" : [ {
    "containerID" : 1,
    "localID" : 115816896921600001,
    "length" : 14,
    "offset" : 0,
    "keyOffset" : 0
  } ],
  "file" : true
}
```

key list

Sergei Tikhomirov

Contents

To Table of Contents

- Definition
- Usage
- Examples

Definition

Lists information about all keys in a given volume, bucket, or snapshot.

Usage

```
$ ozone sh key list [-hV] [-p=<prefix>]
                         [-s=<startItem>]
                         [[-l=<limit>] | [-a]]
                         <value>
```

NOTE

In this command, `list` can be shortened to just `ls`.

Arguments

Parameter	Description
value	URI of a volume or bucket. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
prefix	A name prefix to filter the items
startItem	The item to start listing from. The chosen item will not appear in the results
limit	Maximum number of the results to list. The default value is `100`
-a, --all	If specified, the command will display all the results. By default, it is specified. To counter it, use the `-l` parameter
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket has two keys: `key1` and `key2`. To list information about all the keys, call the following command

```
$ ozone sh key list vol1/bucket1
```

The output depends on key configuration, but overall should be similar to the following:

```
[ {
  "volumeName" : "vol1",
  "bucketName" : "bucket1",
  "name" : "key1",
  "dataSize" : 14,
  "creationTime" : "2025-02-23T21:14:28.098Z",
  "modificationTime" : "2025-02-23T21:17:20.170Z",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "metadata" : { },
  "file" : true
}, {
  "volumeName" : "vol1",
  "bucketName" : "bucket1",
  "name" : "key2",
  "dataSize" : 14,
  "creationTime" : "2025-02-23T22:16:35.235Z",
  "modificationTime" : "2025-02-23T22:16:36.399Z",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "metadata" : { },
  "file" : true
} ]
```

Now, consider that a new key `newkey` was added to the bucket. To find it among the numerous keys using the `list` command, you can utilize the prefix search:

```
$ ozone sh key list vol1/bucket1 -p="new"
```

The output would be:

```
[ {
  "volumeName" : "vol1",
  "bucketName" : "bucket1",
  "name" : "newkey",
  "dataSize" : 14,
  "creationTime" : "2025-02-23T22:38:35.625Z",
  "modificationTime" : "2025-02-23T22:38:36.695Z",
  "replicationConfig" : {
    "replicationFactor" : "ONE",
    "requiredNodes" : 1,
    "replicationType" : "RATIS"
  },
  "metadata" : { },
  "file" : true
} ]
```

key put

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Adds a key with file contents to a bucket.

Usage

```
$ ozone sh key put [-hV] [--stream]
                   [-r=<replication>]
                   [-t=<type>]
                   <value>
                   <fileName>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
replication	Replication definition. The valid values are based on the replication type. For `RATIS`: `ONE` or `THREE`. In case of `EC`, it should follow the `CODEC-DATA-PARITY-CHUNKSIZE` pattern, e.g. `rs-3-2-1024k`
type	Replication type. The supported types are `RATIS` and `EC`
fileName	Path to the file whose contents should be uploaded to a key
--stream	Specifies whether to use Streaming Write Pipeline
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. To add a key to that bucket, call the following command:

```
$ ozone sh key put vol1/bucket1/key1 testFile.txt --stream
```

key removeacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Removes an access control list from a key.

Usage

```
$ ozone sh key removeacl [-hV] [-s=<store>]
                              -a=<list>
                              <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`

list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where:
	- `<type>` is a system entity type like `user` or `group`.
	- `<entity>` is a user or a group the permissions for which you want to control.
	- `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available:
	  - `r` — read the resource data.
	  - `w` — write data to the resource.
	  - `c` — create objects at the resource.
	  - `d` — delete objects at the resource.
	  - `l` — list objects stored at the resource.
	  - `x` — read the access control list of the resource.
	  - `y` — edit the access control list of the resource.
	  - `a` — all of the above.
	  - `n` — no permissions.

-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Removing an ACL from a key:

```
$ ozone sh key removeacl vol1/bucket1/key1 -a=user:sergei:all
```

Confirmation of removal:

```
ACL user:sergei:la[ACCESS] removed successfully.
```

key rename

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Renames an existing key.

Usage

```
$ ozone sh key rename [-hV] <value>
                      <fromKey>
                      <toKey>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
fromKey	An existing key to be renamed
toKey	A new name for the key
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Consider a volume `vol1` with a bucket `bucket1` created within. That bucket contains a key `key1`. To rename that key, call the following command:

```
$ ozone sh key rename vol1/bucket1 key1 key2
```

key setacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Sets one or more ACLs, replacing the existing ones.

IMPORTANT

All the existing ACL will be deleted if they are not present in the new ACL list.

Usage

```
$ ozone sh key setacl [-hV] [-s=<store>]
                            -a=<list>
                            <value>
```

Arguments

Parameter	Description
value	A key URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/key1`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/key1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`

list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where:
	- `<type>` is a system entity type like `user` or `group`.
	- `<entity>` is a user or a group the permissions for which you want to control.
	- `<permissions>` is a sequence of letters that defines the entity’s permissions. The following values are available:
	  - `r` — read the resource data.
	  - `w` — write data to the resource.
	  - `c` — create objects at the resource.
	  - `d` — delete objects at the resource.
	  - `l` — list objects stored at the resource.
	  - `x` — read the access control list of the resource.
	  - `y` — edit the access control list of the resource.
	  - `a` — all of the above.
	  - `n` — no permissions.

-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting an ACL to a key:

```
$ ozone sh key setacl -s=OZONE -a=user:sergei:all vol1/bucket1/key1
```

The confirmation output:

```
ACLs set successfully.
```

prefix addacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Adds an access control list for a prefix.

Usage

```
$ ozone sh prefix addacl [-hV] [-s=<store>]
                               -a=<list>
                               <value>
```

Arguments

Parameter	Description
value	A prefix URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/prefix`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/prefix`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`

list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where:
	- `<type>` is a system entity type like `user` or `group`.
	- `<entity>` is a user or a group the permissions for which you want to control.
	- `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available:
	  - `r` — read the resource data.
	  - `w` — write data to the resource.
	  - `c` — create objects at the resource.
	  - `d` — delete objects at the resource.
	  - `l` — list objects stored at the resource.
	  - `x` — read the access control list of the resource.
	  - `y` — edit the access control list of the resource.
	  - `a` — all of the above.
	  - `n` — no permissions.

-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Adding an ACL:

```
$ ozone sh prefix addacl vol1/bucket1/prefix -s=OZONE -a=user:sergei:all
```

The confirmation output:

```
ACL user:sergei:la[ACCESS] added successfully.
```

prefix getacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists all ACLs for a given prefix.

Usage

```
$ ozone sh prefix getacl [-hV] [-s=<store>] <value>
```

Arguments

Parameter	Description
value	A prefix URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/prefix`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/prefix`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Listing all ACLs for a prefix:

```
$ ozone sh prefix getacl vol1/bucket1/prefix
```

The output depends on the present ACLs, but should be similar to this:

```
[ {
  "type" : "USER",
  "name" : "s_tikhomirov_krb1@AD.RANGER-TEST",
  "aclScope" : "ACCESS",
  "aclList" : [ "ALL" ]
}, {
  "type" : "USER",
  "name" : "sergei",
  "aclScope" : "ACCESS",
  "aclList" : [ "LIST", "ALL" ]
} ]
```

prefix removeacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Removes an access control list from a prefix.

Usage

```
$ ozone sh prefix removeacl [-hV] [-s=<store>]
                              -a=<list>
                              <value>
```

Arguments

Parameter	Description
value	A prefix URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/prefix`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/prefix`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where: ▪ `<type>` is a system entity type like `user` or `group`. ▪ `<entity>` is a user or a group the permissions for which you want to control. ▪ `<permissions>` is a sequence of letters that defines the entity’s permissions. The following values are available: ▪ `r` — read the resource data. ▪ `w` — write data to the resource. ▪ `c` — create objects at the resource. ▪ `d` — delete objects at the resource. ▪ `l` — list objects stored at the resource. ▪ `x` — read the access control list of the resource. ▪ `y` — edit the access control list of the resource. ▪ `a` — all of the above. ▪ `n` — no permissions.
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Removing an ACL from a prefix:

```
$ ozone sh prefix removeacl vol1/bucket1/prefix -a=user:sergei:all
```

Confirmation of removal:

```
ACL user:sergei:la[ACCESS] removed successfully.
```

prefix setacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Sets one or more ACLs, replacing the existing ones.

IMPORTANT

All the existing ACL will be deleted if they are not present in the new ACL list.

Usage

```
$ ozone sh prefix setacl [-hV] [-s=<store>]
                                -a=<list>
                                <value>
```

Arguments

Parameter	Description
value	A prefix URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/prefix`) or a short URI (starts with a volume name, e.g. `vol1/bucket1/prefix`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where: ▪ `<type>` is a system entity type like `user` or `group`. ▪ `<entity>` is a user or a group the permissions for which you want to control. ▪ `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available: ▪ `r` — read the resource data. ▪ `w` — write data to the resource. ▪ `c` — create objects at the resource. ▪ `d` — delete objects at the resource. ▪ `l` — list objects stored at the resource. ▪ `x` — read the access control list of the resource. ▪ `y` — edit the access control list of the resource. ▪ `a` — all of the above. ▪ `n` — no permissions.
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting an ACL to a key:

```
$ ozone sh key setacl -s=OZONE -a=user:sergei:all vol1/bucket1/key1
```

The confirmation output:

```
ACLs set successfully.
```

snapshot create

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Creates a snapshot of a bucket.

Usage

```
$ ozone sh snapshot create [-hV] [<snapshotName>]
                                 <value>
```

Arguments

Parameter	Description
snapshotName	Name of the snapshot. If left empty, a name will be generated
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Create a bucket snapshot:

```
$ ozone sh snapshot create vol1/bucket1 bucket1snapshot
```

snapshot delete

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Deletes a snapshot of a bucket.

Usage

```
$ ozone sh snapshot delete [-hV] <snapshotName>
                                 <value>
```

Arguments

Parameter	Description
snapshotName	Name of the snapshot to delete
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Delete a bucket snapshot:

```
$ ozone sh snapshot delete vol1/bucket1 bucket1snapshot
```

snapshot diff

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Returns a difference between two snapshots.

Usage

```
$ ozone sh snapshot diff [-chV]  [-p=<pageSize>]
                                 [-t=<token>]
                                 <value>
                                 <fromSnapshot>
                                 <toSnapshot>
```

Arguments

Parameter	Description
pageSize	Number of diff entries to be returned
token	Continuation token for the next page
fromSnapshot	An older snapshot of a bucket
toSnapshot	A newer snapshot of a bucket
-c, --cancel	Request to cancel a running SnapshotDiff job. If the job's state is not `IN_PROGRESS`, the request will fail
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Create the first snapshot of a bucket:

```
$ ozone sh snapshot create vol1/bucket2 oldsnap
```

Add a key to that bucket:

```
$ ozone sh key put vol1/bucket2/key42 file.txt
```

Create the second snapshot of a bucket:

```
$ ozone sh snapshot create vol1/bucket2 newsnap
```

Request a diff:

```
$ ozone sh snapshot diff vol1/bucket1 oldsnap newsnap
```

The confirmation that a job has started:

```
Snapshot diff job is IN_PROGRESS. Please retry after 60000 ms.
```

In 60 seconds, repeat the previous command. The result should be as follows:

```
Difference between snapshot: oldsnap and snapshot: newsnap
+       ./key42
```

snapshot info

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Prints information about a bucket.

Usage

```
$ ozone sh snapshot info [-hV] <snapshotName>
                               <value>
```

Arguments

Parameter	Description
snapshotName	Name of a snapshot
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Print information about a snapshot:

```
$ ozone sh snapshot info vol1/bucket2 newsnap
```

The output depends on the snapshot, but should be similar to the following:

```
{
  "volumeName" : "vol1",
  "bucketName" : "bucket2",
  "name" : "newsnap",
  "creationTime" : 1740354183187,
  "snapshotStatus" : "SNAPSHOT_ACTIVE",
  "snapshotId" : "958877db-3d6e-4ff6-9ff6-910f3176b8d7",
  "snapshotPath" : "vol1/bucket2",
  "checkpointDir" : "-958877db-3d6e-4ff6-9ff6-910f3176b8d7",
  "referencedSize" : 5,
  "referencedReplicatedSize" : 17,
  "exclusiveSize" : 0,
  "exclusiveReplicatedSize" : 0
}
```

snapshot list

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists all snapshots of a bucket.

Usage

```
$ ozone sh snapshot list [-hV] [-p=<prefix>]
                                [-s=<startItem>]
                                [[-l=<limit>] | [-a]]
                                <value>
```

Arguments

Parameter	Description
prefix	A name prefix to filter the items
startItem	The item to start listing from. The chosen item will not appear in the results
limit	Maximum number of the results to list. The default value is `100`
-a, --all	If specified, the command will display all the results. By default, it is specified. To counter it, use the `-l` parameter
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

List all snapshots of a bucket:

```
$ ozone sh snapshot list vol1/bucket2
```

The output depends on the bucket, but should be similar to the following:

```
 [ {
  "volumeName" : "vol1",
  "bucketName" : "bucket2",
  "name" : "oldsnap",
  "creationTime" : 1740353795458,
  "snapshotStatus" : "SNAPSHOT_ACTIVE",
  "snapshotId" : "e36a629c-b976-4cb1-b8aa-5b4f4beaf5ca",
  "snapshotPath" : "vol1/bucket2",
  "checkpointDir" : "-e36a629c-b976-4cb1-b8aa-5b4f4beaf5ca",
  "referencedSize" : 0,
  "referencedReplicatedSize" : 0,
  "exclusiveSize" : 0,
  "exclusiveReplicatedSize" : 0
}, {
  "volumeName" : "vol1",
  "bucketName" : "bucket2",
  "name" : "s20250223-232331.183",
  "creationTime" : 1740353011183,
  "snapshotStatus" : "SNAPSHOT_ACTIVE",
  "snapshotId" : "c10f2b45-5f40-4b25-ae0c-ce617e5909a0",
  "snapshotPath" : "vol1/bucket2",
  "checkpointDir" : "-c10f2b45-5f40-4b25-ae0c-ce617e5909a0",
  "referencedSize" : 0,
  "referencedReplicatedSize" : 0,
  "exclusiveSize" : 0,
  "exclusiveReplicatedSize" : 0
} ]
```

snapshot listDiff

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists the SnapshotDiff jobs for a bucket.

Usage

```
$ ozone sh snapshot listDiff [-ahV] [<jobStatus>]
                                    <value>
```

Arguments

Parameter	Description
jobStatus	Allows you to list jobs only with the specified status. Possible values: `queued`, `in_progress`, `done`, `failed`, and `rejected`. By default, returns only the jobs with the `in_progress` status
-a, --all	Allows you to list all jobs regardless of the status
value	A bucket URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/bucket1/`) or a short URI (starts with a volume name, e.g. `vol1/bucket1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

List all the SnapshotDiff jobs for a bucket:

```
$ ozone sh snapshot listDiff vol1/bucket2 -a
```

The output should be similar to:

```
[ {
  "volumeName" : "vol1",
  "bucketName" : "bucket2",
  "fromSnapshot" : "s20250223-232331.183",
  "toSnapshot" : "newsnap",
  "jobStatus" : "DONE"
}, {
  "volumeName" : "vol1",
  "bucketName" : "bucket2",
  "fromSnapshot" : "oldsnap",
  "toSnapshot" : "newsnap",
  "jobStatus" : "DONE"
} ]
```

token cancel

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Cancels a delegation token.

Usage

```
$ ozone sh token cancel [-hV] -t=<tokenFile>
```

Arguments

Parameter	Description
tokenFile	Path to a file containing an encoded token
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Cancelling a token:

```
$ ozone sh token cancel -t=tokenFile
```

The confirmation is as follows:

```
Token canceled successfully.
```

token get

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Issues a delegation token and saves it’s encoded version to a file.

Usage

```
$ ozone sh token get [-hV] [-r=<renewer>]
                           [<uri>]
                           -t=<tokenFile>
```

Arguments

Parameter	Description
renewer	A user who will be able to renew the created token
uri	Resource for which a token will be issued
tokenFile	Path to a file that will contain an encoded token
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Creating a token for a bucket and setting a user as a renewer:

```
$ ozone sh token get -t=bucketKey vol1/bucket1 -r=sergei
```

The output should be similar to:

```
Successfully get token for service 10.92.40.213:9874,10.92.40.239:9874,10.92.42.15:9874
Kind: OzoneToken, Service: 10.92.40.213:9874,10.92.40.239:9874,10.92.42.15:9874, Ident: (OzoneToken owner=s_tikhomirov_krb1@AD.
RANGER-TEST, renewer=sergei, realUser=, issueDate=2025-02-24T10:11:35.453Z, maxDate=2025-03-03T10:11:35.453Z, sequenceNumber=4,
masterKeyId=1, strToSign=null, signature=null, awsAccessKeyId=null, omServiceId=ozone1-mgr, omCertSerialId=7)
```

token print

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Prints information about a delegation token.

Usage

```
$ ozone sh token print -t=<tokenFile>
```

Arguments

Parameter	Description
tokenFile	Path to a file that will contain an encoded token

Examples

Print information about a delegation token:

```
$ ozone sh token print -t=tokenFile
```

The output should be similar to this:

```
Kind: OzoneToken, Service: 10.92.40.213:9874,10.92.40.239:9874,10.92.42.15:9874, Ident: (OzoneToken owner=s_tikhomirov_krb1@AD.
RANGER-TEST, renewer=sergei, realUser=, issueDate=2025-02-24T10:11:35.453Z, maxDate=2025-03-03T10:11:35.453Z, sequenceNumber=4,
masterKeyId=1, strToSign=null, signature=null, awsAccessKeyId=null, omServiceId=ozone1-mgr, omCertSerialId=7)
```

token renew

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Renews a delegation token.

Usage

```
$ ozone sh token renew [-hV] -t=<tokenFile>
```

Arguments

Parameter	Description
tokenFile	Path to a file containing an encoded token
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Renewing a token:

```
$ ozone sh token renew -t=tokenFile
```

The output should be similar to this:

```
Token renewed successfully, expiry time: 2025-02-25T10:51:37.082Z.
```

user assign-admin

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Assigns an admin role to an access ID.

Usage

```
$ ozone sh user assign-admin [-dhV] [--om-service-id=<omServiceID>]
                                   [-t=<tenantId>]
                                   <accessId>
```

Arguments

Parameter	Description
-d, --delegated	Assign delegated admin. By default, a non-delegated admin is assigned
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
tenantId	Tenant (organization) name
accessId	Access ID
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user assign

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Assigns a user to a tenant (organization).

Usage

```
$ ozone sh user assign [-hV] [--om-service-id=<omServiceID>]
                          -t=<tenantId>
                          <userPrincipal>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
tenantId	Tenant (organization) name
userPrincipal	Username
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user get-secret

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Returns a secret given to an access ID. This differs from `ozone s3 getsecret` as this command would not generate a secret when the given access ID doesn’t exist.

Usage

```
$ ozone sh user get-secret [-hV] [--om-service-id=<omServiceID>]
                                <accessId>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
accessId	Access ID to which a secret was given
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user info

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Prints information about a user.

Usage

```
$ ozone sh user info [-hjV] [--om-service-id=<omServiceID>]
                            <userPrincipal>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
userPrincipal	Username
-j, --json	Defines whether to print result as JSON
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user list

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Lists all users in a tenant.

Usage

```
$ ozone sh user list [-hjV] [--om-service-id=<omServiceID>]
                            [-p=<prefix>]
                            <tenantId>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
prefix	A name prefix to filter the items
tenantId	Tenant name
-j, --json	Defines whether to print result as JSON
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user revoke-admin

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage

Definition

Revokes admin privileges from a user.

Usage

```
$ ozone sh user revoke-admin [-hV] [--om-service-id=<omServiceID>]
                                  [-t=<tenantId>]
                                  <accessId>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
accessId	Access ID
tenantId	Tenant name
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user revoke

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Removes a user from an organization (tenant).

Usage

```
$ ozone sh user revoke-admin [-hV] [--om-service-id=<omServiceID>]
                                 [-t=<tenantId>]
                                 <accessId>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
accessId	Access ID
tenantId	Tenant name
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

user set-secret

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Definition

Sets a secret for an access ID.

Usage

```
$ ozone sh user set-secret [-hV] [--om-service-id=<omServiceID>]
                               -s=<secretKey>
                               <accessId>
```

Arguments

Parameter	Description
omServiceID	Ozone Manager service ID. Required when a cluster is in the HA mode
accessId	Access ID
secretKey	Secret key to set
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

volume addacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Adds an access control list for a volume.

Usage

```
$ ozone sh volume addacl [-hV] [-s=<store>]
                              -a=<list>
                              <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`

list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where:
	- `<type>` is a system entity type like `user` or `group`.
	- `<entity>` is a user or a group the permissions for which you want to control.
	- `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available:
	  - `r` — read the resource data.
	  - `w` — write data to the resource.
	  - `c` — create objects at the resource.
	  - `d` — delete objects at the resource.
	  - `l` — list objects stored at the resource.
	  - `x` — read the access control list of the resource.
	  - `y` — edit the access control list of the resource.
	  - `a` — all of the above.
	  - `n` — no permissions.

-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Adding an ACL:

```
$ ozone sh volume addacl vol1 -s=OZONE -a=user:sergei:all
```

The confirmation output:

```
ACL user:sergei:la[ACCESS] added successfully.
```

volume clrquota

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Clears quota for a volume.

Usage

```
$ ozone sh volume clrquota [-hV] [--namespace-quota]
                              [--space-quota]
                              <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
--namespace-quota	Defines whether to clear the namespace quota (maximum number of buckets in a volume). The new value will be `-1` by default
--space-quota	Defines whether to clear the space quota (maximum number of space that can be used). The new value will be `-1` by default
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Clearing a quota for a volume:

```
$ ozone sh volume clrquota --namespace-quota --space-quota vol1
```

volume create

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Creates a new volume.

Usage

```
$ ozone sh volume create [-hV] [--namespace-quota=<quotaInNamespace>]
                         [--quota=<quotaInBytes>]
                         [-u=<ownerName>]
                         <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
quotaInNamespace	Maximum number of buckets in a volume
quotaInBytes	Maximum space quota that can be used (in bytes)
ownerName	Owner of the volume. Defaults to the current user if not specified
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Create a volume with a specified quota:

```
$ ozone sh volume create --quota=1073741824 vol1
```

volume delete

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Deletes a volume.

Usage

```
$ ozone sh volume delete [-hrVy] [-t=<threadNo>] <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
-r	Determines wether to delete a volume recursively
threadNo	Number of threads to perform recursive deletion
-y, --yes	Determines wether to delete without an interactive user confirmation
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Deleting a volume:

```
$ ozone sh volume delete vol1
```

volume getacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Lists all ACLs for a given volume.

Usage

```
$ ozone sh volume getacl [-hV] [-s=<store>] <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Listing all ACLs for a volume:

```
$ ozone sh volume getacl vol1
```

The output depends on the present ACLs, but should be similar to this:

```
[ {
  "type" : "USER",
  "name" : "s_tikhomirov_krb1@AD.RANGER-TEST",
  "aclScope" : "ACCESS",
  "aclList" : [ "ALL" ]
}, {
  "type" : "USER",
  "name" : "sergei",
  "aclScope" : "ACCESS",
  "aclList" : [ "LIST", "ALL" ]
} ]
```

volume info

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Prints information about a volume.

Usage

```
$ ozone sh volume info [-hV] <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files

Examples

Consider a volume `vol1` with five buckets created withing. To get information about the volume, call the following command:

```
$ ozone sh volume info vol1
```

The output depends on volume configuration, but overall should be similar to the following:

```
{
  "metadata" : { },
  "name" : "vol1",
  "admin" : "s_tikhomirov_krb1",
  "owner" : "s_tikhomirov_krb1",
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "usedNamespace" : 5,
  "creationTime" : "2025-01-22T21:23:28.796Z",
  "modificationTime" : "2025-02-12T23:08:16.179Z",
  "acls" : [ {
    "type" : "USER",
    "name" : "s_tikhomirov_krb1",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  }, {
    "type" : "USER",
    "name" : "sergei",
    "aclScope" : "ACCESS",
    "aclList" : [ "READ", "WRITE", "CREATE", "LIST", "DELETE", "READ_ACL", "WRITE_ACL", "ALL" ]
  } ],
  "refCount" : 0
}
```

The `usedNamespace` parameter here displays the number of buckets within a volume.

volume list

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Lists information about all volumes.

Usage

```
$ ozone sh volume list [-hV] [-p=<prefix>]
                             [-s=<startItem>]
                             [-u=<userName>]
                             [[-l=<limit>] | [-a]]
                             <value>
```

NOTE

In this command, `list` can be shortened to just `ls`.

Arguments

Parameter	Description
prefix	A name prefix to filter the items
startItem	The item to start listing from. The chosen item will not appear in the results
userName	A user, whose volumes should be displayed
limit	Maximum number of the results to list. The default value is `100`
-a, --all	If specified, the command will display all the results. By default, it is specified. To counter it, use the `-l` parameter
value	A URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/`) or a short URI (empty in this case). Any missing information is obtained from the configuration files

Examples

Consider there’s a volume `vol1`. To display information about it, call the command:

```
$ ozone sh volume list
```

The output depends on the volume configuration, but overall should be similar to the following:

```
[ {
  "metadata" : { },
  "name" : "vol1",
  "admin" : "s_tikhomirov_krb1",
  "owner" : "s_tikhomirov_krb1",
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "usedNamespace" : 5,
  "creationTime" : "2025-01-22T21:23:28.796Z",
  "modificationTime" : "2025-02-12T23:08:16.179Z",
  "acls" : [ {
    "type" : "USER",
    "name" : "s_tikhomirov_krb1",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  }, {
    "type" : "USER",
    "name" : "sergei",
    "aclScope" : "ACCESS",
    "aclList" : [ "READ", "WRITE", "CREATE", "LIST", "DELETE", "READ_ACL", "WRITE_ACL", "ALL" ]
  } ],
  "refCount" : 0
} ]
```

volume removeacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Removes an access control list from a volume.

Usage

```
$ ozone sh volume removeacl [-hV] [-s=<store>]
                                  -a=<list>
                                  <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`
list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where: ▪ `<type>` is a system entity type like `user` or `group`. ▪ `<entity>` is a user or a group the permissions for which you want to control. ▪ `<permissions>` is a sequence of letters that defines the entity’s permissions. The following values are available: ▪ `r` — read the resource data. ▪ `w` — write data to the resource. ▪ `c` — create objects at the resource. ▪ `d` — delete objects at the resource. ▪ `l` — list objects stored at the resource. ▪ `x` — read the access control list of the resource. ▪ `y` — edit the access control list of the resource. ▪ `a` — all of the above. ▪ `n` — no permissions.
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Removing an ACL from a volume:

```
$ ozone sh volume removeacl vol1 -a=user:sergei:all
```

Confirmation of removal:

```
ACL user:sergei:la[ACCESS] removed successfully.
```

volume setacl

Sergei Tikhomirov

Contents

To Table of Contents

Definition

Usage

Examples

Definition

Sets one or more ACLs, replacing the existing ones.

IMPORTANT

All the existing ACL will be deleted if they are not present in the new ACL list.

Usage

```
$ ozone sh volume setacl [-hV] [-s=<store>]
                          -a=<list>
                          <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
store	Store type: `OZONE` or `S3`

list	A comma-separated list of ACLs. An ACL should match the following pattern: `<type>:<entity>:<permissions>`, where:
	- `<type>` is a system entity type like `user` or `group`.
	- `<entity>` is a user or a group the permissions for which you want to control.
	- `<permissions>` is a sequence of letters that defines the entity's permissions. The following values are available:
	  - `r` — read the resource data.
	  - `w` — write data to the resource.
	  - `c` — create objects at the resource.
	  - `d` — delete objects at the resource.
	  - `l` — list objects stored at the resource.
	  - `x` — read the access control list of the resource.
	  - `y` — edit the access control list of the resource.
	  - `a` — all of the above.
	  - `n` — no permissions.

-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting an ACL to a volume:

```
$ ozone sh volume setacl -s=OZONE -a=user:sergei:all vol1
```

The confirmation output:

```
ACLs set successfully.
```

volume setquota

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Manages storage space quota for a volume.

Usage

```
$ ozone sh volume setquota [-hV] [--namespace-quota=<quotaInNamespace>]
                           [--quota=<quotaInBytes>]
                           <value>
```

Arguments

Parameter	Description
quotaInNamespace	Maximum number of buckets in a volume
quotaInBytes	Maximum space quota that can be used (in bytes)
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Specifying quota for a volume:

```
$ ozone sh volume setquota --namespace-quota=5 --quota=1073741824 vol1
```

volume update

Sergei Tikhomirov

Contents

To Table of Contents

Definition
Usage
Examples

Definition

Updates the owner of the volume.

Usage

```
$ ozone sh volume update [-hV] [--user=<ownerName>] <value>
```

Arguments

Parameter	Description
value	A volume URI. It can be a full URI (starts with `o3://`, e.g. `o3://hostname:9862/vol1/`) or a short URI (starts with a volume name, e.g. `vol1`). Any missing information is obtained from the configuration files
ownerName	Name of the new volume owner
-h, --help	Prints out a help manual for this command
-V, --version	Prints out version information and exits

Examples

Setting a new volume owner:

```
$ ozone sh volume update --user=sergei vol1
```

As a confirmation, the command outputs the new volume information:

```
{
  "metadata" : { },
  "name" : "vol2",
  "admin" : "s_tikhomirov_krb1",
  "owner" : "sergei",
  "quotaInBytes" : -1,
  "quotaInNamespace" : -1,
  "usedNamespace" : 0,
  "creationTime" : "2025-02-23T20:59:54.218Z",
  "modificationTime" : "2025-02-23T21:00:38.275Z",
  "acls" : [ {
    "type" : "USER",
    "name" : "s_tikhomirov_krb1",
    "aclScope" : "ACCESS",
    "aclList" : [ "ALL" ]
  } ],
  "refCount" : 0
}
```

spark3-shell

Konstantin Alpashkin

Contents

To Table of Contents

- Transformation functions
- Action functions
- Persistence storage levels
- Persistence functions
- Examples

Spark shell provides an environment for users to use the Spark functions. It has a lot of different commands that can be used to process data in the interactive shell.

The core concept in Apache Spark is **RDD** (Resilient Distributed Datasheet). It is an immutable distributed collection of data that is partitioned across machines in a cluster.

Transformation functions

A **transformation function** is an operation on an RDD such as `filter()`, `map()` or `union()` that yields another RDD.

Function	Description
map(function)	Returns a new RDD by applying a function on element
filter(function)	Returns a new dataset formed by selecting those source elements for which the function returns `true`
filterByRange(lower, upper)	Returns an RDD with elements in the specified range
flatMap(function)	Similar to the `map()` function but returns a sequence instead of a value
reduceByKey(function,[num Tasks])	It is used to aggregate values of a key using a function
groupByKey([num Tasks])	Used to convert(K,V) to (K, <iterable V>)
distinct([num Tasks])	Used to eliminate duplicates from RDD
mapPartitions(function)	Runs separately on each partition of RDD
mapPartitionsWithIndex(function)	Provides function with an integer value representing the index of the partition
sample(withReplacement, fraction, seed)	Samples a fraction of data using a given random number generating seeds
union(anotherRDD)	Returns a new RDD containing all elements from the source and `anotherRDD` RDDs
intersection(anotherRDD)	Returns a new RDD that contains an intersection of elements in the source and `anotherRDD` datasets
cartesian()	Returns the Cartesian product of all pair of elements
subtract(anotherRDD)	Creates new RDD by removing the content of one RDD using another RDD
join(RDD,[numTasks])	Joins two elements of the dataset with common arguments. When invoked on (A,B) and (A,C), it creates a new RDD (A,(B,C))
cogroup(RDD,[numTasks])	Converts (A,B) to (A, <iterable B>)

Action functions

An **action function** is an operation that triggers a computation such as `count()`, `first()`, `take(n)` or `collect()`.

Function	Description
count()	Gets the number of data elements in the RDD
collect()	Gets all the data elements in the RDD as an array
reduce(function)	Aggregates data elements into the RDD by taking two arguments and returning one
take(n)	Fetches the first `n` elements of the RDD
foreach(function)	Executes function for each data element in the RDD
first()	Retrieves the first data element of the RDD
saveastextfile(path)	Writes the content of RDD to a text file or set of text files to the local system
takeordered(n, [ordering])	Returns the first `n` elements of RDD using either the natural ordering or a custom comparator

Persistence storage levels

A **storage level** specifies how and where to persist or cache an RDD, DataFrame, or Dataset.

Level	Description
MEMORY_ONLY(default level)	Stores the RDD in available cluster memory as a deserialized Java object
MEMORY_AND_DISK	Stores the RDD as a deserialized Java object. If the RDD does not fit in the cluster memory, it stores the partitions on the disk and reads them
MEMORY_ONLY_SER	Stores the RDD as a serialized Java object. This operation is more CPU intensive
MEMORY_ONLY_DISK_SER	This option is the same as `MEMORY_ONLY_SER` but stores the data on disk when the memory is not sufficient
DISC_ONLY	Stores RDD only on the disk
MEMORY_ONLY_2, MEMORY_AND_DISK_2, etc.	This option is the same as other levels except that the partitions are replicated on 2 slave nodes

Persistence functions

Spark's **persistence functions** allow to persist (or cache) a dataset in memory across operations. When you persist an RDD, each node stores RDD partitions in memory and reuses them in other actions on that dataset (or datasets derived from it). This allows future operations on the dataset to be much faster. Caching is a key tool for iterative algorithms and fast interactive use.

Function	Description
cache()	Used to avoid unnecessary re-computation. This is same as `persist(MEMORY_ONLY)`
persist(<storage_level>)	Persists the RDD with the given storage level
unpersist()	Marks the RDD as non-persistent and removes the block from memory and disk
checkpoint()	Saves a file inside the checkpoint directory. All the references of its parent RDD will be removed

Examples

To start the Spark shell, run the following:

```
$ ./bin/spark3-shell
```

Below you can find sample RDD operations in Scala.

Reading a file from the local file system:

```
val data = sc.textFile("dataset1.txt")
```

Creating an RDD through parallelizing:

```
val num = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
val Dataset = sc.parallelize(num)
```

Counting items in RDD:

```
Dataset.count()
```

Saving output/processed data into the text file:

```
counts.saveAsTextFile("output")
```

Filtering an RDD:

```
val DFData = data.filter(line => line.contains("one"))
```

spark3-submit

Konstantin Alpashkin

Contents

To Table of Contents

Master URL

The `spark3-submit` script is used to launch applications on a cluster. It is located in the Spark's *bin* directory.

An example of invoking `spark3-submit` is below:

```
./bin/spark3-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

Where:

- `--class` — the entry point for your Spark application (e.g. `org.apache.spark.examples.SparkPi`).
- `--master` — the master URL for the cluster (e.g. *spark://23.195.26.187:7077*).
- `--deploy-mode` — indicates whether to deploy your driver on the worker nodes (`cluster`) or locally as an external client (`client`, default value).
- `--conf` — an arbitrary Spark configuration property in the `key=value` format. For values that contain spaces, you should wrap the properties in quotes (`"key1"="value 1"`). Multiple configurations should be passed as separate arguments (for example, `--conf <key>=<value> --conf <key2>=<value2>`).
- `<application-jar>` — the path to a JAR including your application and all the dependencies. The URL must be globally accessible inside of your cluster, for instance, an *hdfs://** path or a *file://** path that is present on all nodes.
- `[application-arguments]` — arguments to be passed to the main method of your main class, if any.

Master URL

The master URL passed to Spark can be in one of the following formats:

URL	Description
local	Runs Spark locally with one worker thread (no parallelism)
local[K]	Runs Spark locally with K worker threads (ideally, set this to the number of cores on your machine)
local[K,F]	Runs Spark locally with K worker threads and F `maxFailures` (for more information, see spark.task.maxFailures)
local[*]	Runs Spark locally with as many worker threads as the number of logical cores on your machine
local[*,F]	Runs Spark locally with as many worker threads as the number of logical cores on your machine and F `maxFailures`
local-cluster[N,C,M]	Local-cluster mode is used only for unit tests. It emulates a distributed cluster in a single JVM with N number of workers, C cores per worker and M MiB of memory per worker
spark://HOST:PORT	Connects to the given Spark cluster in the Spark standalone mode. The port must be whichever one your master is configured to use, which is 7077 by default
spark://HOST1:PORT1,HOST2:PORT2	Connects to the given Spark cluster in the Spark standalone mode with standby masters with Zookeeper. The list must have all the master hosts in the high availability cluster set up with Zookeeper. The port must be whichever each master is configured to use, which is 7077 by default
mesos://HOST:PORT	Connects to the given Mesos cluster. The port must be whichever one your is configured to use, which is 5050 by default. Or, for a Mesos cluster using ZooKeeper, use `mesos://zk://…`. To submit with `--deploy-mode` cluster, the HOST:PORT should be configured to connect to the MesosClusterDispatcher
yarn	Connects to a YARN cluster in client or cluster mode depending on the value of `--deploy-mode`. The cluster location will be found based on the `HADOOP_CONF_DIR` or `YARN_CONF_DIR` variable
k8s://HOST:PORT	Connects to a Kubernetes cluster in client or cluster mode depending on the value of `--deploy-mode`. The HOST and PORT refer to the Kubernetes API Server. It connects using TLS by default. In order to force it to use an unsecured connection, you can use `k8s://http://HOST:PORT`

SSM actions

Elena Kostyuchenko

To Table of Contents

Actions in SSM are commands that are applied to data if a certain condition in a rule is met.

The table below lists all possible actions in SSM.

SSM actions

Action	Arguments	Description
addec	[-policyName $policyName] [-codec $codecName -dataNum $dataNum -parityNum $parityNum -cellSize $cellSize]	Sets the requested data policy or an EC policy with the given parameters. For example, `RS-6-3-1024k`. The `cellSize` in bytes should be divisible by 1024. SSM supports human-readable values for `cellSize`, e.g. `1KB`, `1MB`
allssd	-file $file	Moves a file to an SSD storage
alldisk	-file $file	Moves a file to a disk storage
append	-file $file [-length $length -bufSize $bufSize]	Appends a file with the `$length` length and `$bufSize` buffer size
archive	-file $file	Archives a file
cache	-file $file [-replica $replica_num]	Caches a file in the HDFS cache. `$replica_num` — the number of replicas to be cached. If not specified, SSM will get the replica number for `$file` from HDFS or use `1` if it's a directory
checkcompress	-file $file	Shows the compression status of a file. Not applicable to a directory
checkec	-file $file	Shows the EC policy of a file or a directory
checkstorage	-file $file	Shows the storage type of a file
checksum	-file $file	Shows the checksum of a file
compact	-file ['$file1','$file2',..] -containerFile $containerFile	Compacts the requested files to a container file. The compacted files must not have whitespace characters. The containers can't be compacted
compress	-file $file [-bufSize $bufSize] [-codec $codec]	Compresses a file (not applicable to a directory). If the `-bufSize` option is not specified, SSM will use the value obtained by dividing the file size by the max number of splits, configured in *smartdefault.xml*. The least buffer size is 1 MB. SSM supports configuring human-readable values for `bufSize`, e.g. `1MB`. SSM also supports different compression implementations, including Lz4, Bzip2, Zlib and snappy. If the `-codec` option is not specified, SSM will use Zlib by default
concat	-file $src -dest $dest	Concatenates the requested files in the target file. `$src` accepts a list of at least two files separated by comma
copy	-file $src -dest $dest [-offset $offset] [-length $length] [-bufSize $size]	Copies the requested file with additional parameters: offset, length, and buffer size
copy2s3	copy2s3 -file $file -dest $s3_dest	Copies a file to an S3 storage. The `$s3_dest` value must be a URL starting with `s3a://`
decompress	-file $file [-bufSize $size]	Decompresses a file (not applicable to a directory)
delete	-file $file	Deletes a file
disableec	-policy $policyName	Disables the given EC policy
ec	-file $file [-policy $policyName -bufSize $bufSize]	Changes the EC policy of a file to the given one. If the `-policy` option is not provided, SSM will use the default one configured in HDFS. If `-bufSize` is not provided, SSM will use the default value in bytes equivalent to 1 MB. SSM supports human-readable values for `bufSize`, e.g. `1KB`, `1MB`
echo	-msg $message	Prints a message
enableec	-policy $policyName	Enables the given EC policy
list	-file $src	Lists all files in a directory
listec	—	Lists all EC policies
merge	-file $src -dest $dest [-bufSize $size]	Merges files into the target file. The `$src` values must contain at least two files separated by comma
onedisk	-file $file	Moves one replica of a file to disk. Not applicable to EC blocks
onessd	-file $file	Moves one replica of a file to SSD. Not applicable to EC blocks
ramdisk	-file $file	Sets the `LAZY_PERSIST` storage policy. The `$file` parameter must be a directory with the `RAM_DISK` parameter configured beforehand
read	-file $file [-bufSize $size]	Reads a file with the given buffer size
removeec	-policy $policyName	Removes the given EC policy
rename	-file $src -dest $dest	Renames a file or a directory
truncate	-file $src -length $length	Truncates a file to the given size
truncate0	-file $src	Truncates a file to 0 size
uncache	-file $file	Uncaches a file from HDFS cache
uncompact	-containerFile $containerFile	Uncompacts a container to the original files
unec	-file $file [-bufSize $bufSize]	Changes the EC policy for a file to replication. If the `-bufSize` value is not given, SSM will use the default value in bytes equivalent to 1 MB. SSM supports human-readable values for `bufSize`, e.g. `1KB`, `1MB`
write	-file $file -length $length [-bufSize $size]	Writes random data to a file with the given buffer size and length
sleep	-m $millisecond	Sleeps for the given number of milliseconds
sync	-dest $dest -preserve $attributes	Syncs a file in `$dest` and preserves the attributes provided as a comma-separated list: owner, group, permissions, replication, modification-time
distcp	-file $file -target $target [DistCp additional options]	Uses the Hadoop distcp command on given files
user defined actions	—	Allows to implement user-defined actions

SSM rules

Elena Kostyuchenko

To Table of Contents

The rules in SSM consist of four parts:

```
<objects>: <trigger> | <conditions> | <commands>
```

The tables below describe possible values for objects, triggers, and conditions. The commands are described in the actions reference.

For more information on rules syntax, see the Define rules in SSM article; and more complex rules usage examples can be found in SSM rule usage examples.

File object properties

Property	Abbreviation	Description
accessCount(interval)	ac	The access count during the given time interval
accessCountTop(interval,N)	acTop	The topmost N for access count during the given time interval
accessCountBottom(interval,N)	acBot	The bottommost N for access count during the given time interval
accessCountTopOnStoragePolicy(interval, N, "$StoragePolicy")	acTopSp	The topmost N for access count with regard to a storage policy. The supported HDFS storage policies are: - COLD - WARM - HOT - ONE_SSD - ALL_SSD - LAZY_PERSIST
accessCountBottomOnStoragePolicy(interval,N, "$StoragePolicy")	acBotSp	The bottommost N for access count with regard to a storage policy during the last time interval
age	—	The time span from last modification moment to now
atime	—	The last access time
blocksize	—	The block size of the file
ecPolicy	—	The erasure coding policy of the file
inCache	—	The file is in the cache storage
isDir	—	The file is a directory
length	—	The length of the file in bytes. Currently, only numeric values are supported, which indicates the number of bytes
mtime	—	The last modification time of the file
path	—	The file path in HDFS
storagePolicy	—	The storage policy of the file
unsynced	—	The file is not synced

A condition can include one of two possible time values:

- `<time>` — the exact time in the `"yyyy-MM-dd HH:mm:ss:ms"` format, `now`, or `now+<TimeInterval>`. For example, `"2017-07-29 23:00:00"`, `now+7day`.
- `<TimeInterval>` — a time interval described as a number and a unit (`5sec`, `5min`, `5day`) or in the format `Time - Time`, for example, `now - "2016-03-19 23:00:00"`.

Conditions

Value	Description	Example
at <time>	Execute the rule at the given time	at "2017-07-29 23:00:00" at now
every <TimeInterval>	Execute the rule at the given frequency	every 1min
from <Time> [To <Time>]	Along with `every` expression to specify the time scope	every 1day from now every 1min from now to now+7day

Conditional operators

Operator	Description	Example
Logical operator	and, or, not	accessCount(1min) > 0 and length = 5KB
Digital operator	+, -, *, /, %	acTop - acBot > 0
Compare	>, >=, <, <=, ==, !=	accessCount(1min) > 0

YARN CLI

Mikhail Serov

Contents

To Table of Contents

User commands
Administration commands

This section includes the reference documentation for the YARN shell.

All YARN commands are invoked by the *bin/yarn* script. Running the *bin/yarn* script without arguments prints the description for all commands.

The script usage is as follows:

```
$ yarn [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [SUB_COMMAND] [COMMAND_OPTIONS]
```

YARN has an option parsing framework that employs parsing generic options as well as running classes.

YARN shell basic structure

Field	Description
SHELL_OPTIONS	The common set of shell options
GENERIC_OPTIONS	The common set of options supported by multiple commands
COMMAND_OPTIONS	Various commands with their options are described in the following sections

User commands

Command	Description
application	Used to print application reports, kill application, and manage long running application
applicationattempt	Prints applicationattempt reports
classpath	Prints the classpath needed to get the Hadoop JAR and the required libraries
container	Prints container(s) report
jar	Runs a JAR file. Users can bundle their YARN code in a JAR file and execute it using this command
logs	Dumps the container logs
node	Prints node report(s)
queue	Prints queue information
top	Displays YARN applications and the resources they consume
version	Prints the Hadoop version
envvars	Displays computed Hadoop environment variables

Administration commands

Command	Description
daemonlog	Gets/sets the log level for a log identified by a qualified class name in the daemon dynamically
nodemanager	Starts the NodeManager
proxyserver	Starts the web proxy server
resourcemanager	Starts the ResourceManager
rmadmin	Runs ResourceManager admin client
schedulerconf	Updates scheduler configuration. Note, this feature is in alpha phase and is subject to change
scmadmin	Runs Shared Cache Manager admin client
sharedcachemanager	Starts the Shared Cache Manager
timelineserver	Starts the TimeLineServer
registrydns	Starts the RegistryDNS server

application or app

Mikhail Serov

To Table of Contents

The command is used to print application reports, kill applications, and manage long-running applications.

The script usage is as follows:

```
$ yarn application [options]
$ yarn app [options]
```

Arguments

-appId <applicationId>	Specifies the ID of the application to operate on
-appStates <states>	Works with `-list` to filter applications based on a comma-separated list of application states. Valid application states are: `ALL`, `NEW`, `NEW_SAVING`, `SUBMITTED`, `ACCEPTED`, `RUNNING`, `FINISHED`, `FAILED`, `KILLED`
-appTags <tags>	Works with `-list` to filter applications based on a comma-separated list of application tags
-appTags <types>	Works with `-list` to filter applications based on a comma-separated list of application types
-changeQueue <queue_name>	Moves application to a new queue. ApplicationId can be passed using the `-appId` flag. The `-movetoqueue` action is deprecated, this new flag `changeQueue` performs the same functionality
-component <component_name> <count>	Works with the `-flex` flag to change the number of components/containers running for an application or a long-running service. Supports absolute or relative changes, such as `+1`, `2`, or `-3`
-components <components>	Works with the `-upgrade` flag to trigger the upgrade of specified components of the application. Multiple components should be separated by commas
-decomission <application_name>	Decommissions component instances for an application or a long-running service. Requires the `-instances` option. Supports the `-appTypes` option to specify the client implementation to use
-destroy <application_name>	Destroys a saved application specification and removes all application data permanently. Supports the `-appTypes` option to specify which client implementation to use
-enableFastLaunch	Uploads AM dependencies to HDFS to make future launches faster. Supports the `-appTypes` option to specify which client implementation to use
-flex <application_name/ID>	Changes the number of running containers for a component of an application or a long-running service. Requires the `-component` option. If a name is provided, `appType` must be provided unless it is the default `yarn-service`. If an ID is provided, the `appType` will be looked up. Supports the `-appTypes` option to specify which client implementation to use
-help	Displays help for all commands
-instances <component_instances>	Works with the `-upgrade` option to trigger the upgrade of the specified component instances of the application. Also works with the `-decomission` option to decommission specified component instances. Multiple instances should be separated by commas
-kill <application_ID>	Kills the application. Multiple applications can be provided separated with a space
-launch <application_name> <file_name>	Launches the application from specification file (saves specification and starts application). Options `-updateLifetime` and `-changeQueue` can be specified to alter the values provided in the file. Supports the `-appTypes` option to specify which client implementation to use
-list	Lists applications. Supports optional use of `-appTypes` to filter applications based on applications type, `-appStates` to filter applications based on application state and `-appTags` to filter applications based on application tag
-movetoqueue <application_ID>	Deprecated. Moves the application to a different queue. Use the `-changeQueue` option instead
-queue <queue_name>	Works with the `-movetoqueue` flag to specify the queue to move an application to
-save <application_name> <file_name>	Saves specification file for an application. Options `-updateLifetime` and `-changeQueue` can be specified to alter the values provided in the file. Supports the `-appTypes` option to specify which client implementation to use
-start <application_name>	Starts a previously saved application. Supports the `-appTypes` option to specify which client implementation to use
-status <application_name/ID>	Prints the application status. If an ID is provided, prints the generic YARN application status. If a name is provided, prints the application specific status based on app's own implementation, and the `-appTypes` option must be specified unless it is the default `yarn-service` type
-stop <application_name/ID>	Stops application gracefully (may be started again later). If a name is provided, `-appType` must be provided unless it is the default `yarn-service`. If an ID is provided, the `-appType` will be looked up. Supports the `-appTypes` option to specify which client implementation to use
-updateLifetime <timeout>	Updates the timeout of an application from now. An application ID can be passed using the `-appId` option. The timeout value is in seconds
-updatePriority <priority>	Updates the priority of an application. An application ID can be passed using the `-appId` option

applicationattempt

Mikhail Serov

To Table of Contents

Prints the applicationattempt reports.

The usage is as follows:

```
$ yarn applicationattempt [options]
```

Arguments

-help	Displays help
-list <application_ID>	Lists application attempts for the given application
-status <application_attempt_ID>	Prints the status of the application attempt

classpath

Mikhail Serov

To Table of Contents

Prints the classpath needed to get the Hadoop JAR and the required libraries. If called without arguments, prints the classpath set up by the command scripts, which is likely to contain wildcards in the classpath entries. Additional options print the classpath after wildcard expansion or write the classpath into the manifest of a JAR file. The latter is useful in environments where wildcards cannot be used and the expanded classpath exceeds the maximum supported command line length.

The usage is as follows:

```
$ yarn classpath [--glob |--jar <path> |-h |--help]
```

Arguments

--glob	Expands wildcards
--jar <path/to/.jar>	Writes the classpath string into the manifest of the JAR named `<path/to/.jar>`
--help	Prints usage options

container

Mikhail Serov

To Table of Contents

Prints container(s) report.

The usage is as follows:

```
$ yarn container [options]
```

Arguments

-help	Prints help
-list <application_attempt_ID>	Lists containers for the application attempt
-status <container_ID>	Prints the container status

envvars

Mikhail Serov

To Table of Contents

The command prints computed Hadoop environment variables.

The usage is as follows:

```
$ yarn envvars
```

jar

Mikhail Serov

To Table of Contents

Runs a JAR file. You can bundle your YARN code in a JAR file and run it using this command.

The usage is as follows:

```
$ yarn jar <jar> [mainClass]
```

Arguments

<jar>	The path to the JAR to run
mainClass	The name of the main class in the Java. If not specified, the main class is retrieved from the JAR manifest

logs

Mikhail Serov

To Table of Contents

Dumps the container logs.

The usage is as follows:

```
$ yarn logs -applicationId <application_ID> [options]
```

Arguments

-applicationId <application_ID>	Specifies an application ID
-appOwner <appOwner>	Specifies the app owner (assumed to be current user if not specified)
-containerId <container_ID>	Specifies the container ID (must be specified if a `-nodeAddress` is set)
-help	Prints help
-nodeAddress <nodeAddress>	The node address in the format `nodeName:port` (must be specified if `-containerId` is set)

node

Mikhail Serov

To Table of Contents

Prints node reports.

The usage is as follows:

```
$ yarn node [options]
```

Arguments

-all	Works with `-list` to list all nodes
-list	Lists all running nodes. Supports optional use of `-states` to filter nodes based on a node state, and `-all` to list all nodes
-states <states>	Works with `-list` to filter nodes based on input comma-separated list of node states
-status <nodeId>	Prints the status report of the node

queue

Mikhail Serov

Prints queue information.

The usage is as follows:

```
$ yarn queue [options]
```

Arguments

-help	Prints help
-status <queueName>	Prints the queue status

top

Mikhail Serov

To Table of Contents

Displays the status of applications, memory and core usage, and the overall completion percentage of an application in the terminal-emulated screen. This command is similar to the Unix `top` command that displays Linux processes.

The usage is as follows:

```
$ yarn top
```

version

Mikhail Serov

To Table of Contents

The command prints the current Hadoop version.

The usage is as follows:

```
$ yarn version
```

daemonlog

Mikhail Serov

Gets/sets the log level for a log identified by a qualified class name in the daemon dynamically. By default, the command sends an HTTP request, but this can be overridden by using the `-protocol https` argument to send HTTPS requests.

Note that the setting is not permanent and will be reset when the daemon is restarted. This command works by sending a HTTP/HTTPS request to the daemon’s internal Jetty servlet, so it supports the following daemons:

- Common
  - key management server
- HDFS
  - name node
  - secondary name node
  - data node
  - journal node
  - HttpFS server
- YARN
  - resource manager
  - node manager
  - Timeline server

The usage is as follows:

```
$ hadoop daemonlog -getlevel <host:port> <classname> [-protocol (http|https)]
$ hadoop daemonlog -setlevel <host:port> <classname> <level> [-protocol (http|https)]
```

Arguments

-getlevel <host:port> <class_name> [-protocol (httphttps)]	For the daemon running at `<host:port>`, prints the log level of the log identified by a qualified `<class_name>`. The `-protocol` flag specifies the protocol for connection
-setlevel <host:port> <class_name> level [-protocol (httphttps)]	For the daemon running on `<host:port>`, sets the log level of the log identified by a qualified `<class_name>`. The `-protocol` flag specifies the protocol for connection

Example:

```
$ bin/hadoop daemonlog -setlevel 127.0.0.1:9870 org.apache.hadoop.hdfs.server.namenode.NameNode DEBUG
$ bin/hadoop daemonlog -getlevel 127.0.0.1:9871 org.apache.hadoop.hdfs.server.namenode.NameNode DEBUG -protocol https
```

nodemanager

Mikhail Serov

To Table of Contents

The command starts the NodeManager.

The usage is as follows:

```
$ yarn nodemanager
```

proxyserver

Mikhail Serov

To Table of Contents

Starts the proxy web server.

The usage is as follows:

```
$ yarn proxyserver
```

registrydns

Mikhail Serov

To Table of Contents

Starts the RegistryDNS server.

The usage is as follows:

```
$ yarn registrydns
```

resourcemanager

Mikhail Serov

To Table of Contents

Starts the ResourceManager.

The usage is as follows:

```
$ yarn resourcemanager [options]
```

Arguments

-format-state-store	Formats the RMStateStore. This will clear the RMStateStore and is useful if past applications are no longer needed. This should be run only when the ResourceManager is not running
-remove-application-from-state-store <appId>	Removes the application from RMStateStore. This should be run only when the ResourceManager is not running
-format-conf-store	Formats the YarnConfigurationStore. This should be run only when the ResourceManager is not running
-convert-fs-configuration [-y|yarnsiteconfig] [-f|fsconfig] [-r|rulesconfig] [-o|output-directory] [-p|print] [-c|cluster-resource]	This feature is experimental and not intended for production use. Development is still in progress so the converter should not be considered complete. Converts the specified Fair Scheduler configuration to the Capacity Scheduler configuration. Requires two mandatory input files. ▪ The *yarn-site.xml* with the following format: `[-y|yarnsiteconfig <path/to/yarn-site.xml>]`. ▪ The *fair-scheduler.xml* with the following format: `[-f|fsconfig <path/to/fair-scheduler.xml>]`. This config is not mandatory if there is a reference in *yarn-site.xml* to *fair-scheduler.xml* via the property `yarn.scheduler.fair.allocation.file`. If both files are specified, the `-f` option has precedence. The output directory of the config files should be specified as well, with `[-o|output-directory <directory>]`. An optional rules config file can be also specified with the following format: `[-r|rulesconfig <path/to/the/conversion_rules_file>]`. The rule config file format is a *.properties* file. There is an additional optional parameter `[-p|print]`. If defined, the configuration will be printed to the console. In its normal operation, the output files (*yarn-site.xml* and *capacity-scheduler.xml*) are stored to the specified output directory. The cluster resource parameter `[-c|cluster-resource <resource>]` needs to be specified if any queue has a `maxResources` setting with value as percentage. The format of the resource string is the same as in *fair-scheduler.xml*

rmadmin

Mikhail Serov

To Table of Contents

The command Runs the ResourceManager admin client.

The usage is as follows:

```
$ yarn rmadmin [options]
```

Arguments

-refreshQueues	Reloads the queues' acls, states, and scheduler-specific properties. ResourceManager will reload the mapred-queues configuration file
-refreshNodes [-g|graceful <timeout in seconds> -client|server]	Refreshes the hosts information at the ResourceManager. Here `[-g|graceful <timeout in seconds> -client|server]` is optional, if you specify the timeout, then ResourceManager will wait for the timeout before marking the NodeManager as decommissioned. The `[-client|server]` indicates if the timeout tracking should be handled by the client or the ResourceManager. The client-side tracking is blocking, while the server-side tracking is not. Omitting the timeout, or setting the timeout to `-1` indicates an infinite timeout. Known Issue: the server-side tracking will immediately decommission if an RM HA failover occurs
-refreshNodesResources	Refreshes resources of NodeManagers at the ResourceManager
-refreshSuperUsrGroupsConfiguration	Refreshes superuser proxy groups mappings
-refreshUserToGroupsMappings	Refreshes user-to-groups mappings
-refreshAdminAcls	Refreshes ACLs (Accecc Control Lists) for administration of ResourceManager
-refreshServiceAcl	Reloads the service-level authorization policy file
-getGroups <username>	Returns groups the specified user belongs to
-addToCLusterNodeLabels <"label1(exclusive=true),label2(exclusive=false),label3">	Adds node labels to cluster. Default exclusivity is `true`
-removeFromClusterNodeLabels <label1,label2,label3> (label splitted by ",")	Removes from cluster node labels
-replaceLabelsOnNode <"node1[:port]=label1,label2 node2[:port]=label1,label2"> [-failOnUnknownNodes]	Replaces labels on nodes (specifying multiple labels on a single host is not supported for now). `-failOnUnknownNodes` is optional. When set, it will cause a failure if specified nodes are unknown
-directlyAccessNodeLabelStore	Deprecated, will be removed in future releases. Directly access node label store, with this option, all node label related operations will not connect ResourceManager. Instead, they will access/modify stored node labels directly. By default, it is set to `false` (access via Resource Manager). If you configure `yarn.node-labels.fs-store.root-dir` to point to a local directory (instead of NFS or HDFS), this option will only work when the command runs on the machine where Resource Manager is running
-refreshClusterMaxPriority	Refreshes cluster maximum priority
-updateNodeResource [NodeID] [MemSize] [vCores] ([OvercommitTimeout])	Updates resource types on specific nodes. Resource types are comma-delimited key/value pairs of any resources available at Resource Manager. For example, `memory-mb=1024Mi,vcores=1,resource1=2G,resource2=4m`
-transitionToActive [-forceactive] [-forcemanual] <serviceId>	Transitions the service into active state. Try to make the target active without checking that there is no active node if the `-forcemanual` option is used. This command can not be used if automatic failover is enabled. Though you can override this by using `-forcemanual` option, use it with caution
-transitionToStandby [-forcemanual] <serviceId>	Transitions the service into the Standby state. This command cannot be used if automatic failover is enabled. Though you can override this by using `-forcemanual` option, use it with caution
-getServiceState <serviceId>	Returns the state of the specified service
-getAllServiceState	Returns the state of all services
-checkHealth <serviceId>	Requests the service to perform a health check. The RMAdmin tool will exit with a non-zero exit code if the check fails
-help [cmd]	Displays help for the given command or all commands if none is specified

schedulerconf

Mikhail Serov

To Table of Contents

Updates the scheduler configuration. This feature is in the alpha phase and is subject to change.

The usage is as follows:

```
$ yarn shedulerconf [options]
```

Arguments

--add <"queuePath1:key1=val1,key2=val2; queuePath2:key3=val3">	Semicolon-separated values of queues to add and their queue configurations. This example adds the queue `queuePath1` (a full path name) with the queue configurations `key1=val1` and `key2=val2`. It also adds the queue `queuePath2` with the queue configuration `key3=val3`
-remove <"queuePath1;queuePath2">	Semicolon-separated queues to remove. This example removes `queuePath1` and `queuePath2` queues (full path names). Queues must be put into STOPPED state before they are deleted
-update <"queuePath1:key1=val1,key2=val2; queuePath2:key3=val3">	Semicolon-separated values of queues whose configurations should be updated. This example sets `key1=val1` and `key2=val2` for queue configuration of `queuePath1` (full path name), and sets `key3=val3` for queue configuration of `queuePath2`
-global <key1=val1,key2=val2>	Updates global scheduler configurations. This example sets `key1=val1` and `key2=val2` for scheduler's global configuration

scmadmin

Mikhail Serov

To Table of Contents

Runs Shared Cache Manager admin client.

The usage is as follows:

```
$ yarn scmadmin [options]
```

Arguments

-help	Prints help
-runCleanerTask	Runs the cleaner task

sharedcachemanager

Mikhail Serov

To Table of Contents

Starts the Shared Cache Manager.

The usage is as follows:

```
$ yarn sharedcachemanager
```

timelineserver

Mikhail Serov

To Table of Contents

Starts the TimeLineServer.

The usage is as follows:

```
$ yarn timelineserver
```

Configuration parameters

Konstantin Alpashkin, Daria Barysheva

Contents

To Table of Contents

- Airflow
- Core configuration
- Flink
- Flink2
- HBase
- HDFS
- Hive
- HUE
- Impala
- Kyuubi
- MySQL
- Ozone
- Solr
- Spark
- Spark3
- SSM
- Trino
- YARN
- Zeppelin
- ZooKeeper
- Monitoring

This topic describes the parameters that can be configured for ADH services via ADCM. To read about the configuring process, refer to the relevant articles: Online installation, Offline installation.

NOTE

- Some of the parameters become visible in the ADCM UI after the ***Advanced*** flag has been set.
- The parameters that are set in the ***Custom*** group will overwrite the existing parameters even if they are read-only.

Airflow

Redis configuration

Parameter	Description	Default value
redis.conf	Redis configuration file	—
sentinel.conf	Sentinel configuration file	—
redis_port	Redis broker listen port	6379
sentinel_port	Sentinel port	26379

airflow.cfg

Parameter	Description	Default value
db_user	The user to connect to Metadata DB	airflow
db_password	The password to connect to Metadata DB	—
db_port	The port to connect to Metadata DB	3307
admin_password	The password for the web server’s admin user	—
server_port	The port to run the web server	8080
flower_port	The port that Celery Flower runs on	5555
worker_port	When you start an Airflow Worker, Airflow starts a tiny web server subprocess to serve the Workers local log files to the Airflow main web server, which then builds pages and sends them to users. This defines the port, on which the logs are served. The port must be free and accessible from the main web server to connect to the Workers	8793
fernet_key	The secret key to save connection passwords in the database	—
security	Defines which security module to use. For example, `kerberos`	—
keytab	The path to the keytab file	—
reinit_frequency	Sets the ticket renewal frequency	3600
principal	The Kerberos principal	
ssl_active	Defines if SSL is active for Airflow	false
web_server_ssl_cert	The path to SSL certificate	/etc/ssl/certs/host_cert.cert
web_server_ssl_key	The path to SSL certificate key	/etc/ssl/host_cert.key
Logging level	Specifies the logging level for Airflow activity	INFO
Logging level for Flask-appbuilder UI	Specifies the logging level for Flask-appbuilder UI	WARNING
cfg_properties_template	The Jinja template to initialize environment variables for Airflow	cfg_properties_template

External database

Parameter	Description	Default value
Database type	The external database type. Possible values: `PostgreSQL`, `MySQL/MariaDB`	MySQL/MariaDB
Hostname	The external database host	—
Custom port	The external database port	—
Airflow database name	The external database name	airflow

External Broker

Parameter	Description	Default value
Broker URL	The URL of an external broker	—

LDAP Security manager

Parameter	Description	Default value
AUTH_LDAP_SERVER	The LDAP server URI	—
AUTH_LDAP_BIND_USER	The path of the LDAP proxy user to bind on to the top level. Example: `cn=airflow`,`ou=users`,`dc=example`,`dc=com`	—
AUTH_LDAP_BIND_PASSWORD	The password of the bind user	—
AUTH_LDAP_SEARCH	Update with the LDAP path under which you'd like the users to have access to Airflow. Example: `dc=example`, `dc=com`	—
AUTH_LDAP_UID_FIELD	The UID (unique identifier) field in LDAP	—
AUTH_ROLES_MAPPING	The parameter for mapping the internal roles to the LDAP Active Directory groups	—
AUTH_LDAP_GROUP_FIELD	The LDAP user attribute which has their role DNs	—
AUTH_ROLES_SYNC_AT_LOGIN	A flag that indicates if all the user's roles should be replaced on each login, or only on registration	true
PERMANENT_SESSION_LIFETIME	Sets an inactivity timeout after which users have to re-authenticate (to keep roles in sync)	1800
AUTH_LDAP_USE_TLS	Boolean whether TLS is being used	false
AUTH_LDAP_ALLOW_SELF_SIGNED	Boolean to allow self-signed certificates	true
AUTH_LDAP_TLS_CACERTFILE	Location of the certificate	—

Core configuration

core-site.xml

Parameter	Description	Default value
fs.defaultFS	The name of the default file system. A URI whose scheme and authority determine the file system implementation	hdfs://hdfs
fs.trash.checkpoint.interval	Number of minutes between trash checkpoints. Should be smaller or equal to `fs.trash.interval`. If zero, the value is set to the value of `fs.trash.interval`. Every time the checkpointer runs, it creates a new checkpoint and removes checkpoints created more than `fs.trash.interval` minutes ago	60
fs.trash.interval	Number of minutes after which the checkpoint gets deleted. If zero, the trash feature is disabled. This option may be configured both on the server and the client. If trash is disabled on the server side, the client side	1440

	configuration is checked. If trash is enabled on the server side, the value configured on the server is used and the client configuration value is ignored	
hadoop.tmp.dir	A base for temporary directories	/tmp/hadoop-${user.name}
hadoop.zk.address	ZooKeeper server host and port	—
io.file.buffer.size	The size of the buffer to use in sequence files. The size value should be a multiple of hardware page size (e.g. 4096 on Intel x86) and it determines how much data is buffered during the read and write operations	131072
net.topology.script.file.name	The script name that should be invoked to resolve the DNS to `NetworkTopology` name mapping	—
hadoop.proxyuser.hbase.groups	Comma-separated list of groups users from which are allowed to be impersonated by HBase	*
hadoop.proxyuser.hbase.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by HBase	*
hadoop.proxyuser.hue.groups	Comma-separated list of groups users from which are allowed to be impersonated by HUE	*
hadoop.proxyuser.hue.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by HUE	*
hadoop.proxyuser.hbase-phoenix_queryserver.groups	Comma-separated list of groups users from which are allowed to be impersonated by Phoenix Query Server	*
hadoop.proxyuser.hbase-phoenix_queryserver.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Phoenix Query Server	*
hadoop.proxyuser.hive.groups	Comma-separated list of groups users from which are allowed to be impersonated by Hive	*
hadoop.proxyuser.hive.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Hive	*
hadoop.proxyuser.httpfs.groups	Comma-separated list of groups users from which are allowed to be impersonated by HttpFS	*
hadoop.proxyuser.httpfs.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by HttpFS	*
hadoop.proxyuser.HTTP.groups	Comma-separated list of groups users from which are allowed to be impersonated by HTTP keytab services	*
hadoop.proxyuser.HTTP.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by HTTP keytab services	*
hadoop.proxyuser.knox.groups	Comma-separated list of groups users from which are allowed to be impersonated by Knox	*
hadoop.proxyuser.knox.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Knox	*
hadoop.proxyuser.kyuubi.groups	Comma-separated list of groups users from which are allowed to be impersonated by Kyuubi	*
hadoop.proxyuser.kyuubi.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Kyuubi	*
hadoop.proxyuser.livy.groups	Comma-separated list of groups users from which are allowed to be impersonated by Livy	*

hadoop.proxyuser.livy.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Livy	*
hadoop.proxyuser.yarn.groups	Comma-separated list of groups users from which are allowed to be impersonated by YARN	*
hadoop.proxyuser.yarn.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by YARN	*
hadoop.proxyuser.zeppelin.groups	Comma-separated list of groups users from which are allowed to be impersonated by Zeppelin	*
hadoop.proxyuser.zeppelin.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Zeppelin	*
hadoop.proxyuser.trino.groups	Comma-separated list of groups users from which are allowed to be impersonated by Trino	*
hadoop.proxyuser.trino.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Trino	*
fs.s3a.endpoint	AWS S3 endpoint URL	—
fs.s3a.access.key	AWS S3 access key	—
fs.s3a.secret.key	AWS S3 secret key	—
fs.s3a.impl	AWS S3 filesystem class	org.apache.hadoop.fs.s3a.S3AFileSystem
fs.s3a.fast.upload	Defines whether the Fast Upload feature is enabled	true
fs.s3a.connection.ssl.enabled	Defines whether SSL for connections to AWS services is enabled	false
fs.s3a.path.style.access	Defines whether S3 path-style access is enabled	true
hadoop.proxyuser.om.groups	Comma-separated list of groups users from which are allowed to be impersonated by Ozone Manager	*
hadoop.proxyuser.om.hosts	Comma-separated list of hosts. Users connecting from these hosts are allowed to be impersonated by Ozone Manager	*
ha.zookeeper.quorum	A comma-separated list of ZooKeeper server addresses that are to be used by `ZKFailoverController` in automatic failover	—
ipc.client.fallback-to-simple-auth-allowed	Controls whether the client will accept the instruction to switch to the SASL SIMPLE (unsecure) authentication. When set to `false`, the client will not allow the fallback to SIMPLE authentication and will abort the connection	true
hadoop.security.authentication	The authentication type. Possible values: ▪ `simple` — no authentication; ▪ `kerberos` — Kerberos authentication.	simple
hadoop.security.authorization	Controls whether to allow the RPC service-level authorization	false
hadoop.rpc.protection	List of the active protection features. Possible values: ▪ `authentication` — authentication only; ▪ `integrity` — integrity check in addition to authentication; ▪ `privacy` — data encryption in addition to integrity.	authentication

hadoop.security.auth_to_local	General rule for mapping principal names to local user names	—
User managed hadoop.security.auth_to_local	Controls whether to enable automatic generation of `hadoop.security.auth_to_local`	false
hadoop.http.authentication.type	Defines the authentication used for the HTTP web-consoles. Possible values: - `simple` — simple authentication; - `kerberos` — Kerberos authentication; - `[AUTHENTICATION_HANDLER_CLASSNAME]` — custom authentication.	simple
hadoop.http.authentication.kerberos.principal	Kerberos principal to be used with the `kerberos` authentication. The principal short name must be HTTP per Kerberos HTTP SPNEGO specification. `_HOST` (if present) is replaced with the bind address of the HTTP server	HTTP/_HOST@$LOCALHOST
hadoop.http.authentication.kerberos.keytab	Location of the keytab file with the credentials for the Kerberos principal	/etc/security/keytabs/HTTP.service.keytab
ha.zookeeper.acl	ACLs for all znodes	—
hadoop.http.filter.initializers	Add to this property the `org.apache.hadoop.security.AuthenticationFilterInitializer` initializer class	org.apache.hadoop.security.AuthenticationFilterInitializer
hadoop.http.authentication.signature.secret.file	The signature secret file for signing the authentication tokens. If not set, a random secret is generated during the startup. The same secret should be used for all nodes in the cluster, JobTracker, NameNode, DataNode, and TastTracker. This file should be readable only by the Unix user running the daemons	/etc/security/http_secret
hadoop.http.authentication.cookie.domain	The domain to use for the HTTP cookie that stores the authentication token. In order for authentication to work properly across all nodes in the cluster, the domain must be correctly set. There is no default value, the HTTP cookie will not have a domain working only with the hostname issuing the HTTP cookie	—
hadoop.ssl.require.client.cert	Controls whether the client certificates are required	false
hadoop.ssl.hostname.verifier	The hostname verifier to provide for `HttpsURLConnections`. Possible values: - `DEFAULT` - `STIRCT` - `STRICT_IE6` - `DEFAULT_AND_LOCALHOST` - `ALLOW_ALL`	DEFAULT
hadoop.ssl.keystores.factory.class	The `KeyStoresFactory` implementation to use	org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory
hadoop.ssl.server.conf	Resource file from which the SSL server keystore information will be extracted. This file is looked up in the CLASSPATH. Typically, it should be in Hadoop's *conf/* directory	ssl-server.xml
hadoop.ssl.client.conf	Resource file from which the SSL client keystore information will be extracted. This file is looked up in the CLASSPATH. Typically, it should be in Hadoop's *conf/* directory	ssl-client.xml
hadoop.ssl.enabled.protocols	The supported SSL protocols	TLSv1.2

fs.AbstractFileSystem.ofs.impl	AbstractFileSystem for Rooted Ozone file system (ofs) URI	org.apache.hadoop.fs.ozone.RootedOzFs
fs.ofs.impl	The implementation class of the ofs file system	org.apache.hadoop.fs.ozone.RootedOzoneFileSystem

ssl-server.xml

Parameter	Description	Default value
ssl.server.truststore.location	Location of the truststore file to be used by NameNodes and DataNodes	—
ssl.server.truststore.password	Truststore password	—
ssl.server.truststore.type	Truststore file format	jks
ssl.server.truststore.reload.interval	Truststore reload check interval in milliseconds	10000
ssl.server.keystore.location	Location of the keystore file to be used by NameNodes and DataNodes	—
ssl.server.keystore.password	Keystore password	—
ssl.server.keystore.keypassword	Password to the key in the keystore	—
ssl.server.keystore.type	Keystore file format	jks

ssl-client.xml

Parameter	Description	Default value
ssl.client.truststore.location	Location of the truststore file to be used by NameNodes and DataNodes	—
ssl.client.truststore.password	Truststore password	—
ssl.client.truststore.type	Truststore file format	jks
ssl.client.truststore.reload.interval	Truststore reload check interval in milliseconds	10000
ssl.client.keystore.location	Location of the keystore file to be used by NameNodes and DataNodes	—
ssl.client.keystore.password	Keystore password	—
ssl.client.keystore.keypassword	Password to the key in the keystore	—
ssl.client.keystore.type	Keystore file format	jks

Other

Parameter	Description	Default value
Custom core-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *core-site.xml* configuration file	—
Custom hadoop-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *hadoop-env.sh* configuration file	—
Custom ssl-server.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *ssl-server.xml* configuration file	—
Custom ssl-client.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *ssl-client.xml* configuration file	—
Custom log4j.properties	Custom logging properties	log.conf

The Configuration server component

nginx.conf

Parameter	Description	Default value
root_config_dir	Root location for storing configurations	/srv/config
nginx_http_port	HTTP port for Nginx	9998
nginx_https_port	HTTPS port for Nginx	9998
ssl_certificate	Path to SSL certificate for Nginx	/etc/ssl/certs/host_cert.cert
ssl_certificate_key	Path to SSL certificate key for Nginx	/etc/ssl/host_cert.key
ssl_protocols	SSL protocol version to be supported for the SSL transport	TLSv1.2

Flink

flink-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *flink-env.sh*	/usr/lib/bigtop-utils/bigtop-detect-javahome
HADOOP_CLASSPATH	A list of files/directories to be added to the classpath	$(hadoop classpath)
HADOOP_HOME	Hadoop home directory	/usr/lib/hadoop
HADOOP_CONF_DIR	Directory that stores Hadoop configurations	/etc/hadoop/conf
FLINK_LOG_DIR	Flink server log directory	/var/log/flink
FLINK_CONF_DIR	Directory that stores Flink configurations	/etc/flink/conf
FLINK_HOME	The Flink home directory	/usr/lib/flink

$HADOOP_OPTS	Additional Flink startup parameters	$HADOOP_OPTS -Dsecurity.kerberos.token.provider.hadoopfs.renewer=yarn

flink-conf.yaml

Parameter	Description	Default value
historyserver.archive.fs.dir	A comma-separated list of directories with job archives scanned by Flink History Server	hdfs:///apps/flink/completed-jobs
jobmanager.archive.fs.dir	The directory where JobManager stores completed job archives	hdfs:///apps/flink/completed-jobs
archive.fs.refresh-interval	The polling interval in milliseconds for scanning archive directories	10000
historyserver.web.port	The port number of the Flink History Server web UI	8082
historyserver.web.ssl.enabled	Enables/disables SSL for the History Server web UI	false
rest.port	The port that the client connects to. This option is respected only if the high-availability configuration is `NONE`	8081
jobmanager.rpc.port	The RPC port through which the JobManager is reachable. In the high availability mode, this value is ignored and the port number to connect to JobManager is generated by ZooKeeper	6123
sql-gateway.endpoint.rest.port	A port to connect to the SQL Gateway service	8083
taskmanager.network.bind-policy	The automatic address binding policy used by the TaskManager	name
parallelism.default	The system-wide default parallelism level for all execution environments	1
taskmanager.numberOfTaskSlots	The number of task slots that each TaskManager offers. Each slot runs one parallel pipeline	1
taskmanager.cpu.cores	The number of CPU cores used by TaskManager. By default, the value is set to the number of slots per TaskManager	1
taskmanager.memory.flink.size	The total memory size for the TaskExecutors	—
taskmanager.memory.process.size	The total process memory size for the TaskExecutors. This includes all the memory that a TaskExecutor consumes, including the total Flink memory, JVM Metaspace, and JVM Overhead. In containerized setups, this parameter should be equal to the container memory	2048m
jobmanager.memory.flink.size	The total memory size for the JobManager	—
jobmanager.memory.process.size	The total process memory size for the JobManager. This includes all the memory that a JobManager JVM consumes, including the total Flink memory, JVM Metaspace, and JVM Overhead. In containerized setups, this parameter should be equal to the container memory	2048m
taskmanager.heap.size	The Java heap size for the TaskManager JVM	1024m
jobmanager.memory.heap.size	The heap size for the JobManager JVM	—
flink.yarn.appmaster.vcores	The number of virtual cores (vcores) used by YARN application master	1

taskmanager.host	The external address of the network interface where TaskManager runs	—
taskmanager.memory.task.heap.size	The size of the JVM heap memory reserved for tasks	256m
taskmanager.memory.task.off-heap.size	The size of the off-heap memory reserved for tasks	256m
taskmanager.memory.managed.size	The size of the managed memory for TaskExecutors. This is the size of off-heap memory managed by the memory manager, reserved for sorting, hash tables, caching of intermediate results, and the RocksDB state backend	256m
taskmanager.memory.framework.heap.size	The size of the JVM heap memory reserved for TaskExecutor framework that will not be allocated to task slots	256m
taskmanager.memory.framework.off-heap.size	The size of the off-heap memory reserved for TaskExecutor framework that will not be allocated to task slots	256m
taskmanager.memory.network.min	The minimum network memory size for TaskExecutors. The network memory is the off-heap memory reserved for ShuffleEnvironment (e.g. network buffers)	256m
taskmanager.memory.network.max	The maximum network memory size for TaskExecutors. The network memory is the off-heap memory reserved for ShuffleEnvironment (e.g. network buffers)	256m
taskmanager.memory.jvm-overhead.max	The maximum JVM overhead size for the TaskExecutors. This is the off-heap memory reserved for JVM overhead, such as thread stack space, compile cache, etc.	256m
taskmanager.memory.jvm-metaspace.size	The JVM metaspace size for TaskExecutors	256m
yarn.provided.lib.dirs	A semicolon-separated list of directories with provided libraries. Flink uses these libraries to exclude the local Flink JARs uploading to accelerate the job submission process	hdfs:///apps/flink/
flink.yarn.resourcemanager.scheduler.address	The address of the scheduler interface	—
flink.yarn.containers.vcores	Sets the number of vcores for Flink YARN containers	1
flink.yarn.application.classpath	A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ /etc/hadoop/conf/* ▪ /usr/lib/hadoop/* ▪ /usr/lib/hadoop/lib/* ▪ /usr/lib/hadoop-hdfs/* ▪ /usr/lib/hadoop-hdfs/lib/* ▪ /usr/lib/hadoop-yarn/* ▪ /usr/lib/hadoop-yarn/lib/* ▪ /usr/lib/hadoop-mapreduce/* ▪ /usr/lib/hadoop-mapreduce/lib/* ▪ /usr/lib/ssm/lib/smart*.jar
high-availability.cluster-id	The ID of the Flink cluster used to separate multiple Flink clusters from each other	default
high-availability.storageDir	A file system path (URI) where Flink persists metadata in the HA mode	—
high-availability	Defines the High Availability (HA) mode used for cluster execution	ZOOKEEPER

high-availability.zookeeper.quorum	The ZooKeeper quorum to use when running Flink in the HA mode with ZooKeeper	—
high-availability.zookeeper.path.root	The root path for Flink ZNode in Zookeeper	/flink
sql-gateway.session.check-interval	The check interval to detect idle sessions. A value <= `0` disables the checks	1 min
sql-gateway.session.idle-timeout	The timeout to close a session if no successful connection was made during this interval. A value <= `0` never closes the sessions	10 min
sql-gateway.session.max-num	The maximum number of sessions to run simultaneously	1000000
sql-gateway.worker.keepalive-time	The time to keep an idle worker thread alive. When the worker thread count exceeds `sql-gateway.worker.threads.min`, excessive threads are killed after this time interval	5 min
sql-gateway.worker.threads.max	The maximum number of worker threads on the SQL Gateway server	500
sql-gateway.worker.threads.min	The minimum number of worker threads. If the current number of worker threads is less than this value, the worker threads are not deleted automatically	5
env.java.opts.all	Java options used to start all Flink JVMs	-Djava.io.tmpdir={{ cluster.config.java_tmpdir | d('/tmp') }}
security.kerberos.login.use-ticket-cache	Indicates whether to read from the Kerberos ticket cache	false
security.delegation.token.provider.HiveServer2.enabled	Defines whether to obtain credentials from HiveServer2 when security is enabled	false
security.kerberos.login.keytab	The absolute path to the Kerberos keytab file that stores user credentials	—
security.kerberos.login.principal	Flink Kerberos principal	—
security.delegation.tokens.hive.renewer	Flink Kerberos principal for Hive	—
security.kerberos.login.contexts	A comma-separated list of login contexts to provide the Kerberos credentials to	—
security.ssl.internal.enabled	Enables SSL for internal communication channels between Flink components. This includes the communication between TaskManagers, transporting of blobs from JobManager to TaskManager, RPC-connections, etc.	false
security.ssl.internal.keystore	The path to the keystore file to be used by Flink's internal endpoints	—
security.ssl.internal.truststore	The path to the truststore file used by internal Flink's endpoints	—
security.ssl.internal.keystore-password	The password to the keystore file used by internal Flink's endpoints	—
security.ssl.internal.truststore-password	The password to the truststore file used by internal Flink's endpoints	—

security.ssl.internal.key-password	The password to decrypt the key in the keystore	—
security.ssl.rest.enabled	Turns on SSL for external communication via REST endpoints	false
security.ssl.rest.keystore	The Java keystore file with SSL keys and certificates to be used by Flink's external REST endpoints	—
security.ssl.rest.truststore	The truststore file containing public CA certificates to verify the peer for Flink's external REST endpoints	—
security.ssl.rest.keystore-password	The secret to decrypt the keystore file for Flink external REST endpoints	—
security.ssl.rest.truststore-password	The password to decrypt the truststore for Flink's external REST endpoints	—
security.ssl.rest.key-password	The secret to decrypt the key in the keystore for Flink's external REST endpoints	—
security.ssl.protocol	The TLS protocol version to be used for SSL. Accepts a single value, not a list	TLSv1.2
zookeeper.sasl.disable	Defines the SASL authentication in Zookeeper	false
Logging level	Defines the logging level for Flink activity	INFO
yarn.classpath.include-user-jar	Defines whether to add user JARs to the classpath and specifies their position in the classpath. The supported values: ▪ `DISABLED` — user JARs are not added to the classpath. ▪ `FIRST` — user JARs are added before system libraries. This allows you to override certain system classes/libraries. ▪ `LAST` — user JARs are added to the end of the classpath string. ▪ `ORDER` — user JARs are added to the classpath string in alphabetical order.	ORDER

Other

Parameter	Description	Default value
Custom flink_env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *flink_env.sh*	—
Custom flink-conf.yaml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *flink-conf.yaml*	—
log4j.properties	The contents of the *log4j.properties* configuration file	log4j.properties
log4j-cli.properties	The contents of the *log4j-cli.properties* configuration file	log4j-cli.properties

Flink2

flink-env.sh

Parameter	Description	Default value

Sources	A list of sources that will be written into *flink-env.sh*	/usr/lib/bigtop-utils/bigtop-detect-javahome
HADOOP_CLASSPATH	A list of files/directories to be added to the classpath	$(hadoop classpath)
HADOOP_HOME	Hadoop home directory	/usr/lib/hadoop
HADOOP_CONF_DIR	Directory that stores Hadoop configurations	/etc/hadoop/conf
FLINK_LOG_DIR	Flink server log directory	/var/log/flink2
FLINK_CONF_DIR	Directory that stores Flink configurations	/etc/flink2/conf
FLINK_HOME	The Flink home directory	/usr/lib/flink2
$HADOOP_OPTS	Additional Flink startup parameters	$HADOOP_OPTS -Dsecurity.kerberos.token.provider.hadoopfs.renewer=yarn

config.yaml

Parameter	Description	Default value
historyserver.archive.fs.dir	A comma-separated list of directories with job archives scanned by Flink History Server	hdfs:///apps/flink2/completed-jobs
jobmanager.archive.fs.dir	The directory where JobManager stores completed job archives	hdfs:///apps/flink2/completed-jobs
archive.fs.refresh-interval	The polling interval in milliseconds for scanning archive directories	10000
historyserver.web.port	The port number of the Flink History Server web UI	8092
historyserver.web.ssl.enabled	Enables/disables SSL for the History Server web UI	false
jobmanager.rpc.port	The RPC port through which the JobManager is reachable. In the high availability mode, this value is ignored and the port number to connect to JobManager is generated by ZooKeeper	6123
sql-gateway.endpoint.rest.port	A port to connect to the SQL Gateway service	8093
taskmanager.network.bind-policy	The automatic address binding policy used by the TaskManager	name
parallelism.default	The system-wide default parallelism level for all execution environments	1
taskmanager.numberOfTaskSlots	The number of task slots that each TaskManager offers. Each slot runs one parallel pipeline	1
taskmanager.cpu.cores	The number of CPU cores used by TaskManager. By default, the value is set to the number of slots per TaskManager	1
taskmanager.memory.process.size	The total process memory size for the TaskExecutors. This includes all the memory that a TaskExecutor consumes, including the total Flink memory, JVM Metaspace, and JVM Overhead. In containerized setups, this parameter should be equal to the container memory	2048m
jobmanager.memory.process.size	The total process memory size for the JobManager. This includes all the memory that a JobManager JVM consumes, including the total Flink memory, JVM Metaspace, and JVM Overhead. In containerized setups, this parameter should be equal to the container memory	2048m

flink.yarn.appmaster.vcores	The number of virtual cores (vcores) used by YARN application master	1
taskmanager.host	The external address of the network interface where TaskManager runs	—
taskmanager.memory.task.heap.size	The size of the JVM heap memory reserved for tasks	256m
taskmanager.memory.task.off-heap.size	The size of the off-heap memory reserved for tasks	256m
taskmanager.memory.managed.size	The size of the managed memory for TaskExecutors. This is the size of off-heap memory managed by the memory manager, reserved for sorting, hash tables, caching of intermediate results, and the RocksDB state backend	256m
taskmanager.memory.framework.heap.size	The size of the JVM heap memory reserved for TaskExecutor framework that will not be allocated to task slots	256m
taskmanager.memory.framework.off-heap.size	The size of the off-heap memory reserved for TaskExecutor framework that will not be allocated to task slots	256m
taskmanager.memory.network.min	The minimum network memory size for TaskExecutors. The network memory is the off-heap memory reserved for ShuffleEnvironment (e.g. network buffers)	256m
taskmanager.memory.network.max	The maximum network memory size for TaskExecutors. The network memory is the off-heap memory reserved for ShuffleEnvironment (e.g. network buffers)	256m
taskmanager.memory.jvm-overhead.max	The maximum JVM overhead size for the TaskExecutors. This is the off-heap memory reserved for JVM overhead, such as thread stack space, compile cache, etc.	256m
taskmanager.memory.jvm-metaspace.size	The JVM metaspace size for TaskExecutors	256m
yarn.provided.lib.dirs	A semicolon-separated list of directories with provided libraries. Flink uses these libraries to exclude the local Flink JARs uploading to accelerate the job submission process	hdfs:///apps/flink2/
flink.yarn.resourcemanager.scheduler.address	The address of the scheduler interface	—
flink.yarn.containers.vcores	Sets the number of vcores for Flink YARN containers	1
flink.yarn.application.classpath	A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ /etc/hadoop/conf/* ▪ /usr/lib/hadoop/* ▪ /usr/lib/hadoop/lib/* ▪ /usr/lib/hadoop-hdfs/* ▪ /usr/lib/hadoop-hdfs/lib/* ▪ /usr/lib/hadoop-yarn/* ▪ /usr/lib/hadoop-yarn/lib/* ▪ /usr/lib/hadoop-mapreduce/* ▪ /usr/lib/hadoop-mapreduce/lib/* ▪ /usr/lib/ssm/lib/smart*.jar
sql-gateway.session.check-interval	The check interval to detect idle sessions. A value <= `0` disables the checks	1 min
sql-gateway.session.idle-timeout	The timeout to close a session if no successful connection was made during this interval. A value <= `0` never closes the sessions	10 min

sql-gateway.session.max-num	The maximum number of sessions to run simultaneously	1000000
sql-gateway.worker.keepalive-time	The time to keep an idle worker thread alive. When the worker thread count exceeds `sql-gateway.worker.threads.min`, excessive threads are killed after this time interval	5 min
sql-gateway.worker.threads.max	The maximum number of worker threads on the SQL Gateway server	500
sql-gateway.worker.threads.min	The minimum number of worker threads. If the current number of worker threads is less than this value, the worker threads are not deleted automatically	5
env.java.opts.all	Java options used to start all Flink JVMs	-Djava.io.tmpdir={{ cluster.config.java_tmpdir | d('/tmp') }}
security.kerberos.login.use-ticket-cache	Indicates whether to read from the Kerberos ticket cache	false
security.delegation.token.provider.HiveServer2.enabled	Defines whether to obtain credentials from HiveServer2 when security is enabled	false
security.kerberos.login.keytab	The absolute path to the Kerberos keytab file that stores user credentials	—
security.kerberos.login.principal	Flink Kerberos principal	—
security.delegation.tokens.hive.renewer	Flink Kerberos principal for Hive	—
security.kerberos.login.contexts	A comma-separated list of login contexts to provide the Kerberos credentials to	—
security.ssl.internal.enabled	Enables SSL for internal communication channels between Flink components. This includes the communication between TaskManagers, transporting of blobs from JobManager to TaskManager, RPC-connections, etc.	false
security.ssl.internal.keystore	The path to the keystore file to be used by Flink's internal endpoints	—
security.ssl.internal.truststore	The path to the truststore file used by internal Flink's endpoints	—
security.ssl.internal.keystore-password	The password to the keystore file used by internal Flink's endpoints	—
security.ssl.internal.truststore-password	The password to the truststore file used by internal Flink's endpoints	—
security.ssl.internal.key-password	The password to decrypt the key in the keystore	—
security.ssl.rest.enabled	Turns on SSL for external communication via REST endpoints	false
security.ssl.rest.keystore	The Java keystore file with SSL keys and certificates to be used by Flink's external REST endpoints	—
security.ssl.rest.truststore	The truststore file containing public CA certificates to verify the peer for Flink's external REST endpoints	—
security.ssl.rest.keystore-password	The secret to decrypt the keystore file for Flink external REST endpoints	—

security.ssl.rest.truststore-password	The password to decrypt the truststore for Flink's external REST endpoints	—
security.ssl.rest.key-password	The secret to decrypt the key in the keystore for Flink's external REST endpoints	—
security.ssl.protocol	The TLS protocol version to be used for SSL. Accepts a single value, not a list	TLSv1.2
Logging level	Defines the logging level for Flink activity	INFO
yarn.classpath.include-user-jar	Defines whether to add user JARs to the classpath and specifies their position in the classpath. The supported values: ▪ `DISABLED` — user JARs are not added to the classpath. ▪ `FIRST` — user JARs are added before system libraries. This allows you to override certain system classes/libraries. ▪ `LAST` — user JARs are added to the end of the classpath string. ▪ `ORDER` — user JARs are added to the classpath string in alphabetical order.	ORDER

Other

Parameter	Description	Default value
Custom flink_env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *flink_env.sh*	—
Custom config.yaml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *config.yaml*	—
Custom sql-gateway-runner flink-conf.yaml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file sql-gateway-runner *flink-conf.yaml*	—
log4j.properties	The contents of the *log4j.properties* configuration file	log4j.properties
log4j-cli.properties	The contents of the *log4j-cli.properties* configuration file	log4j-cli.properties
log4j-console.properties	The contents of the *log4j-console.properties* configuration file	log4j-console.properties
log4j-session.properties	The contents of the *log4j-session.properties* configuration file	log4j-session.properties

HBase

hbase-site.xml

Parameter	Description	Default value
hbase.balancer.period	The time period to run the Region balancer in Master	300000
hbase.client.pause	General client pause value. Used mostly as value to wait before running a retry of a failed get, region lookup, etc. See `hbase.client.retries.number` for description of how this pause works with retries	100

hbase.client.max.perregion.tasks	The maximum number of concurrent mutation tasks the Client will maintain to a single Region. That is, if there is already `hbase.client.max.perregion.tasks` writes in progress for this Region, new puts won't be sent to this Region, until some writes finishes	1
hbase.client.max.perserver.tasks	The maximum number of concurrent mutation tasks a single HTable instance will send to a single Region Server	2
hbase.client.max.total.tasks	The maximum number of concurrent mutation tasks, a single HTable instance will send to the cluster	100
hbase.client.retries.number	The maximum number of retries. It is used as maximum for all retryable operations, such as: getting a cell value, starting a row update, etc. Retry interval is a rough function based on `hbase.client.pause`. See the constant `RETRY_BACKOFF` for how the backup ramps up. Change this setting and `hbase.client.pause` to suit your workload	15
hbase.client.scanner.timeout.period	The Client scanner lease period in milliseconds	60000
hbase.cluster.distributed	The cluster mode. Possible values are: `false` — for standalone mode and pseudo-distributed setups with managed ZooKeeper; `true` — for fully-distributed mode with unmanaged ZooKeeper Quorum. If `false`, the startup will run all HBase and ZooKeeper daemons together in the one JVM, if `true` — one JVM instance per daemon	true
hbase.hregion.majorcompaction	The time interval between Major compactions in milliseconds. Set to `0` to disable time-based automatic Major compactions. User-requested and size-based Major compactions will still run. This value is multiplied by `hbase.hregion.majorcompaction.jitter` to cause compaction to start at a somewhat-random time during a given time frame	604800000
hbase.hregion.max.filesize	The maximum file size. If the total size of some Region HFiles has grown to exceed this value, the Region is split in two. There are two options of how this option works: the first is when any store size exceeds the threshold — then split, and the other is if overall Region size exceeds the threshold — then split. It can be configured by `hbase.hregion.split.overallfiles`	10737418240
hbase.hstore.blockingStoreFiles	If more than this number of StoreFiles exists in any Store (one StoreFile is written per flush of MemStore), updates are blocked for this Region, until a compaction is completed, or until `hbase.hstore.blockingWaitTime` is exceeded	16
hbase.hstore.blockingWaitTime	The time for which a Region will block updates after reaching the StoreFile limit, defined by `hbase.hstore.blockingStoreFiles`. After this time is elapsed, the Region will stop blocking updates, even if a compaction has not been completed	90000
hbase.hstore.compaction.max	The maximum number of StoreFiles that will be selected for a single Minor compaction, regardless of the number of eligible StoreFiles. Effectively, the value of `hbase.hstore.compaction.max` controls the time it takes for a single compaction to complete. Setting it larger means that more StoreFiles are included in a compaction. For most cases, the default value is appropriate	10
hbase.hstore.compaction.min	The minimum number of StoreFiles that must be eligible for compaction before compaction can run. The goal of tuning `hbase.hstore.compaction.min` is to avoid a situation with too many tiny StoreFiles to compact. Setting this value to `2` would cause a Minor compaction each time you have two StoreFiles in a Store, and this is probably not appropriate. If you set this value too high, all the other values will need to be adjusted accordingly. For most cases, the default value is appropriate. In the previous versions of HBase, the parameter `hbase.hstore.compaction.min` was called `hbase.hstore.compactionThreshold`	3

hbase.hstore.compaction.min.size	A StoreFile, smaller than this size, will always be eligible for Minor compaction. StoreFiles this size or larger are evaluated by `hbase.hstore.compaction.ratio` to determine, if they are eligible. Because this limit represents the "automatic include" limit for all StoreFiles smaller than this value, this value may need to be reduced in write-heavy environments, where many files in the 1-2 MB range are being flushed, because every StoreFile will be targeted for compaction and the resulting StoreFiles may still be under the minimum size and require further compaction. If this parameter is lowered, the ratio check is triggered more quickly. This addressed some issues seen in earlier versions of HBase, but changing this parameter is no longer necessary in most situations	134217728
hbase.hstore.compaction.ratio	For Minor compaction, this ratio is used to determine, whether a given StoreFile that is larger than `hbase.hstore.compaction.min.size`, is eligible for compaction. Its effect is to limit compaction of large StoreFile. The value of `hbase.hstore.compaction.ratio` is expressed as a floating-point decimal	1.2F
hbase.hstore.compaction.ratio.offpeak	The compaction ratio used during off-peak compactions if the off-peak hours are also configured. Expressed as a floating-point decimal. This allows for more aggressive (or less aggressive, if you set it lower than `hbase.hstore.compaction.ratio`) compaction during a given time period. The value is ignored if off-peak is disabled (default). This works the same as `hbase.hstore.compaction.ratio`	5.0F
hbase.hstore.compactionThreshold	If more than this number of StoreFiles exists in any Store (one StoreFile is written per flush of MemStore), a compaction is run to rewrite all StoreFiles into a single StoreFile. Larger values delay the compaction, but when compaction does occur, it takes longer to complete	3
hbase.hstore.flusher.count	The number of flush threads. With fewer threads, the MemStore flushes will be queued. With more threads, the flushes will be executed in parallel, increasing the load on HDFS, and potentially causing more compactions	2
hbase.hstore.time.to.purge.deletes	The amount of time to delay purging of delete markers with future timestamps. If unset or set to `0`, all the delete markers, including those with future timestamps, are purged during the next Major compaction. Otherwise, a delete marker is kept until the Major compaction that occurs after the marker timestamp plus the value of this setting (in milliseconds)	0
hbase.master.ipc.address	HMaster RPC	0.0.0.0
hbase.normalizer.period	The period at which the Region normalizer runs on Master (in milliseconds)	300000
hbase.regionserver.compaction.enabled	Enables/disables compactions by setting `true`/`false`. You can further switch compactions dynamically with the `compaction_switch` shell command	true
hbase.regionserver.ipc.address	Region Server RPC	0.0.0.0
hbase.regionserver.regionSplitLimit	The limit for the number of Regions, after which no more Region splitting should take place. This is not hard limit for the number of Regions, but acts as a guideline for the Region Server to stop splitting after a certain limit	1000
hbase.rootdir	The directory shared by Region Servers and into which HBase persists. The URL should be fully-qualified to include the filesystem scheme. For example, to specify the HDFS directory */hbase* where the HDFS instance NameNode is running at *namenode.example.org* on port *9000*, set this value to: `hdfs://namenode.example.org:9000/hbase`	—
hbase.zookeeper.quorum	A comma-separated list of servers in the ZooKeeper ensemble. For example, `host1.mydomain.com,host2.mydomain.com,host3.mydomain.com`. By default, this is set to `localhost` for local and pseudo-distributed modes of operation. For a fully-distributed setup, this should be set to a full list of ZooKeeper ensemble servers. If `HBASE_MANAGES_ZK` is set in *hbase-env.sh*, this is the list of servers, which HBase will start/stop ZooKeeper on, as part of cluster start/stop. Client-side, the list of ensemble members is put together with the `hbase.zookeeper.property.clientPort` config and is passed to the ZooKeeper constructor as the connection string parameter	—

zookeeper.session.timeout	The ZooKeeper session timeout in milliseconds. It is used in two different ways. First, this value is processed by the ZooKeeper Client that HBase uses to connect to the ensemble. It is also used by HBase, when it starts a ZooKeeper Server (in that case the timeout is passed as the `maxSessionTimeout`). See more details in the ZooKeeper documentation. For example, if an HBase Region Server connects to a ZooKeeper ensemble that is also managed by HBase, then the session timeout will be the one specified by this configuration. But a Region Server that connects to an ensemble managed with a different configuration will be subjected to the `maxSessionTimeout` of that ensemble. So, even though HBase might propose using `90` seconds, the ensemble can have a max timeout, lower than this, and it will take precedence. The current default `maxSessionTimeout` that ZooKeeper ships with is `40` seconds, which is lower than HBase	90000
zookeeper.znode.parent	The root znode for HBase in ZooKeeper. All of the HBase ZooKeeper files configured with a relative path will go under this node. By default, all of the HBase ZooKeeper file paths are configured with a relative path, so they will all go under this directory unless changed	/hbase
hbase.rest.port	The port used by HBase Rest Servers	60080
hbase.zookeeper.property.authProvider.1	Specifies the ZooKeeper authentication method	
hbase.security.authentication	Set the value to `true` to run HBase RPC with strong authentication	false
hbase.security.authorization	Set the value to `true` to run HBase RPC with strong authorization	false
hbase.master.kerberos.principal	The Kerberos principal used to run the HMaster process	—
hbase.master.keytab.file	Full path to the Kerberos keytab file to use for logging in the configured HMaster server principal	—
hbase.regionserver.kerberos.principal	The Kerberos principal name that should be used to run the HRegionServer process	—
hbase.regionserver.keytab.file	Full path to the Kerberos keytab file to use for logging in the configured HRegionServer server principal	—
hbase.rest.authentication.type	REST Gateway Kerberos authentication type	—
hbase.rest.authentication.kerberos.principal	REST Gateway Kerberos principal	—
hbase.rest.authentication.kerberos.keytab	REST Gateway Kerberos principal	—
hbase.rest.support.proxyuser	Enables running the REST server to support the proxy user mode	false
hbase.thrift.keytab.file	Thrift Kerberos keytab	—
hbase.rest.keytab.file	HBase REST gateway Kerberos keytab	—
hbase.rest.kerberos.principal	HBase REST gateway Kerberos principal	—
hbase.thrift.kerberos.principal	Thrift Kerberos principal	—
hbase.thrift.security.qop	Defines authentication, integrity, and confidentiality checking. Supported values: `auth-conf` — authentication, integrity, and confidentiality checking; `auth-int` — authentication and integrity checking; `auth` — authentication checking only.	—
phoenix.queryserver.keytab.file	The path to the Kerberos keytab file	—

phoenix.queryserver.kerberos.principal	The Kerberos principal to use when authenticating. If `phoenix.queryserver.kerberos.http.principal` is not defined, this principal specified will be also used to both authenticate SPNEGO connections and to connect to HBase	—
phoenix.queryserver.kerberos.keytab	The full path to the Kerberos keytab file to use for logging in the configured HMaster server principal	—
phoenix.queryserver.http.keytab.file	The keytab file to use for authenticating SPNEGO connections. This configuration must be specified if `phoenix.queryserver.kerberos.http.principal` is configured. `phoenix.queryserver.keytab.file` will be used if this property is undefined	—
phoenix.queryserver.http.kerberos.principal	The Kerberos principal to use when authenticating SPNEGO connections. `phoenix.queryserver.kerberos.principal` will be used if this property is undefined	—
phoenix.queryserver.kerberos.http.principal	Deprecated, use `phoenix.queryserver.http.kerberos.principal` instead	—
hbase.security.authentication.ui	Enables Kerberos authentication to HBase web UI with SPNEGO	—
hbase.security.authentication.spnego.kerberos.principal	The Kerberos principal for SPNEGO authentication	—
hbase.security.authentication.spnego.kerberos.keytab	The path to the Kerberos keytab file with principals to be used for SPNEGO authentication	—
hbase.ssl.enabled	Defines whether SSL is enabled for web UIs	false
hadoop.ssl.enabled	Defines whether SSL is enabled for Hadoop RPC	false
ssl.server.keystore.location	The path to the keystore file	—
ssl.server.keystore.password	The password to the keystore	—
ssl.server.truststore.location	The path to the truststore to be used	—
ssl.server.truststore.password	The password to the truststore	—
ssl.server.keystore.keypassword	The password to the key in the keystore	—
hbase.rest.ssl.enabled	Defines whether SSL is enabled for HBase REST server	false
hbase.rest.ssl.keystore.store	The path to the keystore used by HBase REST server	—
hbase.rest.ssl.keystore.password	The password to the keystore	—
hbase.rest.ssl.keystore.keypassword	The password to the key in the keystore	—
hadoop.security.credential.provider.path	Path to the credential provider (jceks) containing the passwords to all services	—

Credential encryption

Parameter	Description	Default value
Encryption enable	Defines whether the credentials are encrypted	false
Credential provider path	Path to the credential provider for creating the *.jceks* files containing secret keys	jceks://file/etc/hbase/conf/hbase.jceks
Ranger plugin credential provider path	Path to the Ranger plugin credential provider	jceks://file/etc/hbase/conf/ranger-hbase.jceks

Custom jceks	Defines whether custom *.jceks* files located at the credential provider path are used (`true`) or auto-generated ones (`false`)	false
Password file name	Name of the password file in the classpath of the service if the password file is selected in the credstore options	hbase_credstore_pass

hbase-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *hbase-env.sh*	—
HBASE_MASTER_OPTS	Specifies additional parameters for `HBASE_MASTER_OPTS`	-Xms700m -Xmx9G
PHOENIX_QUERYSERVER_OPTS	Specifies additional parameters for `PHOENIX_QUERYSERVER_OPTS`	-Xms700m -Xmx8G
HBASE_THRIFT_OPTS	Specifies additional parameters for `HBASE_THRIFT_OPTS`	-Xms700m -Xmx8G
HBASE_REST_OPTS	Specifies additional parameters for `HBASE_REST_OPTS`	-Xms200m -Xmx8G
HBASE_OPTS	Specifies additional parameters for `HBASE_OPTS`	-XX:+UseConcMarkSweepGC
HBASE_CLASSPATH	The classpath for HBase. A list of files/directories to be added to the classpath. To add more items to the classpath, click +	/usr/lib/phoenix/phoenix-server-hbase.jar
Final HBASE_OPTS	Final value of the `HBASE_OPTS` parameter in *hbase-env.sh*	—

hbase-regionserver-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *hbase-regionserver-env.sh*	—
HBASE_REGIONSERVER_OPTS	Sets initial (-Xms) and maximum (-Xmx) Java heap size for HBase Region server	-Xms700m -Xmx9G
Final HBASE_REGIONSERVER_OPTS	Final value of the `HBASE_REGIONSERVER_OPTS` parameter in *hbase-regionserver-env.sh*	—

ranger-hbase-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/hdfs_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Uses in-memory JAAS configuration file to connect to Solr	—

xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal’s secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—
xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service that wants to log in	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal’s key to be stored in the subject’s private credentials	false
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal’s key from the keytab	false

ranger-hbase-security.xml

Parameter	Description	Default value
ranger.plugin.hbase.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.hbase.service.name	The name of the Ranger service containing policies for this instance	—
ranger.plugin.hbase.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	/srv/ranger/hbase/policycache
ranger.plugin.hbase.policy.pollIntervalMs	Defines how often to poll for changes in policies	30000
ranger.plugin.hbase.policy.rest.client.connection.timeoutMs	The HBase Plugin RangerRestClient connection timeout (in milliseconds)	120000
ranger.plugin.hbase.policy.rest.client.read.timeoutMs	The HBase Plugin RangerRestClient read timeout (in milliseconds)	30000
ranger.plugin.hbase.policy.rest.ssl.config.file	The path to the RangerRestClient SSL config file for HBase plugin	/etc/hbase/conf/ranger-hbase-policymgr-ssl.xml

ranger-hbase-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/hbase/conf/ranger-hbase.jceks

xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/hbase/conf/ranger-hbase.jceks
xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

Other

Parameter	Description	Default value
Custom hbase-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hbase-site.xml*	—
Custom hbase-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hbase-env.sh*	—
Custom hbase-regionserver-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hbase-regionserver-env.sh*	—
Ranger plugin enabled	Whether or not Ranger plugin is enabled	false
Custom ranger-hbase-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hbase-audit.xml*	—
Custom ranger-hbase-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hbase-security.xml*	—
Custom ranger-hbase-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hbase-policymgr-ssl.xml*	—
Custom log4j.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *log4j.properties*	log4j.properties
Custom hadoop-metrics2-hbase.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hadoop-metrics2-hbase.properties*	hadoop-metrics2-hbase.properties

HDFS

Credential Encryption

Parameter	Description	Default value
Encryption enable	Enables or disables the credential encryption feature. When enabled, HDFS stores configuration passwords and credentials required for interacting with other services in the encrypted form	false

Credential provider path	The path to a keystore file with secrets	jceks://file/etc/hadoop/conf/hadoop.jceks
Ranger plugin credential provider path	The path to a Ranger keystore file with secrets	jceks://file/etc/hadoop/conf/ranger-hdfs.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the default auto-generated JCEKS file	false
Password file name	The name of the file in the service's classpath that stores passwords	hadoop_credstore_pass

Enable CORS

Parameter	Description	Default value
hadoop.http.cross-origin.enabled	Enables cross-origin support for all web services	true
hadoop.http.cross-origin.allowed-origins	Comma-separated list of origins that are allowed. Values prefixed with `regex` are interpreted as regular expressions. Values containing wildcards (`*`) are possible as well, here a regular expression is generated, the use is discouraged and support is only available for backward compatibility	*
hadoop.http.cross-origin.allowed-headers	Comma-separated list of allowed headers	X-Requested-With,Content-Type,Accept,Origin,WWW-Authenticate,Accept-Encoding,Transfer-Encoding
hadoop.http.cross-origin.allowed-methods	Comma-separated list of methods that are allowed	GET,PUT,POST,OPTIONS,HEAD,DELETE
hadoop.http.cross-origin.max-age	Number of seconds a pre-flighted request can be cached	1800
core_site.enable_cors.active	Enables CORS (Cross-Origin Resource Sharing)	true

hdfs-site.xml

Parameter	Description	Default value
dfs.client.block.write.replace-datanode-on-failure.enable	If there is a DataNode/network failure in the write pipeline, DFSClient will try to remove the failed DataNode from the pipeline and then continue writing with the remaining DataNodes. As a result, the number of DataNodes in the pipeline is decreased. The feature is to add new DataNodes to the pipeline. This is a site-wide property to enable/disable the feature. When the cluster size is extremely small, e.g. 3 nodes or less, cluster administrators may want to set the policy to `NEVER` in the default configuration file or disable this feature. Otherwise, users may experience an unusually high rate of pipeline failures since it is impossible to find new DataNodes for replacement. See also `dfs.client.block.write.replace-datanode-on-failure.policy`	true

dfs.client.block.write.replace-datanode-on-failure.policy	This property is used only if the value of `dfs.client.block.write.replace-datanode-on-failure.enable` is `true`. Possible values:
	- `ALWAYS`. Always adds a new DataNode, when an existing DataNode is removed.
	- `NEVER`. Never adds a new DataNode.
	- `DEFAULT`. Let `r` be the replication number. Let `n` be the number of existing DataNodes. Add a new DataNode only, if `r` is greater than or equal to 3 and either:
	  - `floor(r/2)` is greater than or equal to `n`;
	  - `r` is greater than `n` and the block is hflushed/appended.
		DEFAULT

dfs.client.block.write.replace-datanode-on-failure.best-effort	This property is used only if the value of `dfs.client.block.write.replace-datanode-on-failure.enable` is `true`. Best effort means, that the client will try to replace a failed DataNode in write pipeline (provided that the policy is satisfied), however, it continues the write operation in case that the DataNode replacement also fails. Suppose, the DataNode replacement fails: `false` — an exception should be thrown so that the write will fail; `true` — the write should be resumed with the remaining DataNodes. Note, that setting this property to `true` allows writing to a pipeline with a smaller number of DataNodes. As a result, it increases the probability of data loss	false
dfs.client.block.write.replace-datanode-on-failure.min-replication	The minimum number of replications needed not to fail the write pipeline if new DataNodes can not be found to replace failed DataNodes (could be due to network failure) in the write pipeline. If the number of the remaining DataNodes in the write pipeline is greater than or equal to this property value, continue writing to the remaining nodes. Otherwise throw exception. If this is set to `0`, an exception will be thrown, when a replacement can not be found. See also `dfs.client.block.write.replace-datanode-on-failure.policy`	0
dfs.balancer.dispatcherThreads	The size of the thread pool for the HDFS balancer block mover — **dispatchExecutor**	200
dfs.balancer.movedWinWidth	The time window in milliseconds for the HDFS balancer tracking blocks and its locations	5400000
dfs.balancer.moverThreads	The thread pool size for executing block moves — **moverThreadAllocator**	1000
dfs.balancer.max-size-to-move	The maximum number of bytes that can be moved by the balancer in a single thread	10737418240
dfs.balancer.getBlocks.min-block-size	The minimum block threshold size in bytes to ignore, when fetching a source block list	10485760
dfs.balancer.getBlocks.size	The total size in bytes of DataNode blocks to get, when fetching a source block list	2147483648
dfs.balancer.block-move.timeout	The maximum amount of time for a block to move (in milliseconds). If set greater than `0`, the balancer will stop waiting for a block move completion after this time. In typical clusters, a 3-5 minute timeout is reasonable. If the timeout is set for a large proportion of block moves, this needs to be increased. It could also be that too much work is dispatched and many nodes are constantly exceeding the bandwidth limit as a result. In that case, other balancer parameters might need to be adjusted. It is disabled (`0`) by default	0
dfs.balancer.max-no-move-interval	If this specified amount of time has elapsed and no blocks have been moved out of a source DataNode, one more attempt will be made to move blocks out of this DataNode in the current Balancer iteration	60000
dfs.balancer.max-iteration-time	The maximum amount of time an iteration can be run by the Balancer. After this time the Balancer will stop the iteration, and re-evaluate the work needed to be done to balance the cluster. The default value is `20` minutes	1200000
dfs.blocksize	The default block size for new files (in bytes). You can use the following suffixes to define size units (case insensitive): `k` (kilo), `m` (mega), `g` (giga), `t` (tera), `p` (peta), `e` (exa). For example, 128k, 512m, 1g, etc. You can also specify the block size in bytes (such as 134217728 for 128 MB)	134217728
dfs.client.read.shortcircuit	Turns on short-circuit local reads	true
dfs.datanode.balance.max.concurrent.moves	The maximum number of threads for DataNode balancer pending moves. This value is reconfigurable via the `dfsadmin -reconfig` command	50
dfs.datanode.data.dir	Determines, where on the local filesystem a DFS data node should store its blocks. If multiple directories are specified, then data will be stored in all named directories, typically on different devices. The directories should be tagged with corresponding storage types (`SSD`/`DISK`/`ARCHIVE`/`RAM_DISK`) for HDFS storage policies. The default storage type will be `DISK` if the directory does not have a storage type tagged explicitly. Directories, that do not exist, will be created, if the local filesystem permission allows	/srv/hadoop-hdfs/data:DISK

dfs.disk.balancer.max.disk.throughputInMBperSec	The maximum disk bandwidth, used by the disk balancer during reads from a source disk. The unit is MB/sec	10
dfs.disk.balancer.block.tolerance.percent	The parameter specifies when a good enough value is reached for any copy step (in percents). For example, if set to to `10` then getting close to 10% of the target value is considered as good enough. In other words, if the move operation is 20GB in size, if 18GB (20 * (1-10%)) can be moved, the entire operation is considered successful	10
dfs.disk.balancer.max.disk.errors	During a block move from a source to destination disk, there might be various errors. This parameter defines how many errors to tolerate before declaring a move between 2 disks (or a step) has failed	5
dfs.disk.balancer.plan.valid.interval	The maximum amount of time a disk balancer plan (a set of configurations that define the data volume to be redistributed between two disks) remains valid. This setting supports multiple time unit suffixes as described in `dfs.heartbeat.interval`. If no suffix is specified, then milliseconds are assumed	1d
dfs.disk.balancer.plan.threshold.percent	Defines a data storage threshold in percents at which disks start participating in data redistribution or balancing activities	10
dfs.domain.socket.path	The path to a UNIX domain socket that will be used for communication between the DataNode and local HDFS clients. If the string `_PORT` is present in this path, it will be replaced by the TCP port of the DataNode. The parameter is optional	/var/lib/hadoop-hdfs/dn_socket
dfs.hosts	Names a file that contains a list of hosts allowed to connect to the NameNode. The full pathname of the file must be specified. If the value is empty, all hosts are permitted	/etc/hadoop/conf/dfs.hosts
dfs.mover.movedWinWidth	The minimum time interval for a block to be moved to another location again (in milliseconds)	5400000
dfs.mover.moverThreads	Sets the balancer mover thread pool size	1000
dfs.mover.retry.max.attempts	The maximum number of retries before the mover considers the move as failed	10
dfs.mover.max-no-move-interval	If this specified amount of time has elapsed and no block has been moved out of a source DataNode, one more attempt will be made to move blocks out of this DataNode in the current mover iteration	60000
dfs.namenode.name.dir	Determines where on the local filesystem the DFS name node should store the name table (fsimage). If multiple directories are specified, then the name table is replicated in all of the directories, for redundancy	/srv/hadoop-hdfs/name
dfs.namenode.checkpoint.dir	Determines where on the local filesystem the DFS secondary name node should store the temporary images to merge. If multiple directories are specified, then the image is replicated in all of the directories for redundancy	/srv/hadoop-hdfs/checkpoint
dfs.namenode.hosts.provider.classname	The class that provides access for host files. `org.apache.hadoop.hdfs.server.blockmanagement.HostFileManager` is used by default that loads files specified by `dfs.hosts` and `dfs.hosts.exclude`. If `org.apache.hadoop.hdfs.server.blockmanagement.CombinedHostFileManager` is used, it will load the JSON file defined in `dfs.hosts`. To change the class name, NameNode restart is required. `dfsadmin -refreshNodes` only refreshes the configuration files, used by the class	org.apache.hadoop.hdfs.server.blockmanagement.CombinedHostFileManager
dfs.namenode.rpc-bind-host	The actual address, the RPC Server will bind to. If this optional address is set, it overrides only the hostname portion of `dfs.namenode.rpc-address`. It can also be specified per NameNode or name service for HA/Federation. This is useful for making the NameNode listen on all interfaces by setting it to `0.0.0.0`	0.0.0.0
dfs.permissions.superusergroup	The name of the group of super-users. The value should be a single group name	hadoop
dfs.replication	The default block replication. The actual number of replications can be specified, when the file is created. The default is used, if replication is not specified in create time	3
dfs.journalnode.http-address	The HTTP address of the JournalNode web UI	0.0.0.0:8480

dfs.journalnode.https-address	The HTTPS address of the JournalNode web UI	0.0.0.0:8481
dfs.journalnode.rpc-address	The RPC address of the JournalNode web UI	0.0.0.0:8485
dfs.datanode.http.address	The address of the DataNode HTTP server	0.0.0.0:9864
dfs.datanode.https.address	The address of the DataNode HTTPS server	0.0.0.0:9865
dfs.datanode.address	The address of the DataNode for data transfer	0.0.0.0:9866
dfs.datanode.ipc.address	The IPC address of the DataNode	0.0.0.0:9867
dfs.namenode.http-address	The address and the base port to access the dfs NameNode web UI	0.0.0.0:9870
dfs.namenode.https-address	The secure HTTPS address of the NameNode	0.0.0.0:9871
dfs.ha.automatic-failover.enabled	Defines whether automatic failover is enabled	true
dfs.ha.fencing.methods	A list of scripts or Java classes that will be used to fence the Active NameNode during a failover	shell(/bin/true)
dfs.journalnode.edits.dir	The directory where to store journal edit files	/srv/hadoop-hdfs/journalnode
dfs.namenode.shared.edits.dir	The directory on shared storage between the multiple NameNodes in an HA cluster. This directory will be written by the active and read by the standby in order to keep the namespaces synchronized. This directory does not need to be listed in `dfs.namenode.edits.dir`. It should be left empty in a non-HA cluster	---
dfs.internal.nameservices	A unique nameservices identifier for a cluster or federation. For a single cluster, specify the name that will be used as an alias. For HDFS federation, specify, separated by commas, all namespaces associated with this cluster. This option allows you to use an alias instead of an IP address or FQDN for some commands, for example: `hdfs dfs -ls hdfs://<dfs.internal.nameservices>`. The value must be alphanumeric without underscores	—
dfs.block.access.token.enable	If set to `true`, access tokens are used as capabilities for accessing DataNodes. If set to `false`, no access tokens are checked on accessing DataNodes	false
dfs.namenode.kerberos.principal	The NameNode service principal. This is typically set to `nn/_HOST@REALM.TLD`. Each NameNode will substitute `_HOST` with its own fully qualified hostname during the startup. The `_HOST` placeholder allows using the same configuration setting on both NameNodes in an HA setup	nn/_HOST@REALM
dfs.namenode.keytab.file	The keytab file used by each NameNode daemon to login as its service principal. The principal name is configured with `dfs.namenode.kerberos.principal`	/etc/security/keytabs/nn.service.keytab
dfs.namenode.kerberos.internal.spnego.principal	HTTP Kerberos principal name for the NameNode	HTTP/_HOST@REALM
dfs.web.authentication.kerberos.principal	Kerberos principal name for the WebHDFS	HTTP/_HOST@REALM
dfs.web.authentication.kerberos.keytab	Kerberos keytab file for WebHDFS	/etc/security/keytabs/HTTP.service.keytab
dfs.journalnode.kerberos.principal	The JournalNode service principal. This is typically set to `jn/_HOST@REALM.TLD`. Each JournalNode will substitute `_HOST` with its own fully qualified hostname at startup. The `_HOST` placeholder allows using the same configuration setting on all JournalNodes	jn/_HOST@REALM

dfs.journalnode.keytab.file	The keytab file used by each JournalNode daemon to login as its service principal. The principal name is configured with `dfs.journalnode.kerberos.principal`	/etc/security/keytabs/jn.service.keytab
dfs.journalnode.kerberos.internal.spnego.principal	The server principal used by the JournalNode HTTP Server for SPNEGO authentication when Kerberos security is enabled. This is typically set to `HTTP/_HOST@REALM.TLD`. The SPNEGO server principal begins with the prefix `HTTP/` by convention. If the value is `*`, the web server will attempt to login with every principal specified in the keytab file `dfs.web.authentication.kerberos.keytab`. For most deployments this can be set to `${dfs.web.authentication.kerberos.principal}` that is use the value of `dfs.web.authentication.kerberos.principal`	HTTP/_HOST@REALM
dfs.datanode.data.dir.perm	Permissions for the directories on the local filesystem where the DFS DataNode stores its blocks. The permissions can either be octal or symbolic	700
dfs.datanode.kerberos.principal	The DataNode service principal. This is typically set to `dn/_HOST@REALM.TLD`. Each DataNode will substitute `_HOST` with its own fully qualified host name at startup. The `_HOST` placeholder allows using the same configuration setting on all DataNodes	dn/_HOST@REALM.TLD
dfs.datanode.keytab.file	The keytab file used by each DataNode daemon to login as its service principal. The principal name is configured with `dfs.datanode.kerberos.principal`	/etc/security/keytabs/dn.service.keytab
dfs.http.policy	Defines if HTTPS (SSL) is supported on HDFS. This configures the HTTP endpoint for HDFS daemons. The following values are supported: `HTTP_ONLY` — the service is provided only via http; `HTTPS_ONLY` — the service is provided only via https; `HTTP_AND_HTTPS` — the service is provided both via http and https	HTTP_ONLY
dfs.data.transfer.protection	A comma-separated list of SASL protection values used for secured connections to the DataNode when reading or writing block data. The possible values are: ■ `authentication` — provides only authentication; no integrity or privacy; ■ `integrity` — authentication and integrity are enabled; ■ `privacy` — authentication, integrity and privacy are enabled. If `dfs.encrypt.data.transfer=true`, then it supersedes the setting for `dfs.data.transfer.protection` and enforces that all connections must use a specialized encrypted SASL handshake. This property is ignored for connections to a DataNode listening on a privileged port. In this case, it is assumed that the use of a privileged port establishes sufficient trust	—
dfs.encrypt.data.transfer	Defines whether or not actual block data that is read/written from/to HDFS should be encrypted on the wire. This only needs to be set on the NameNodes and DataNodes, clients will deduce this automatically. It is possible to override this setting per connection by specifying custom logic via `dfs.trustedchannel.resolver.class`	false
dfs.encrypt.data.transfer.algorithm	This value may be set to either `3des` or `rc4`. If nothing is set, then the configured JCE default on the system is used (usually 3DES). It is widely believed that 3DES is more secure, but RC4 is substantially faster. Note that if AES is supported by both the client and server, then this encryption algorithm will only be used to initially transfer keys for AES	3des
dfs.encrypt.data.transfer.cipher.suites	This value can be either undefined or `AES`/`CTR`/`NoPadding`. If defined, then `dfs.encrypt.data.transfer` uses the specified cipher suite for data encryption. If not defined, then only the algorithm specified in `dfs.encrypt.data.transfer.algorithm` is used	—
dfs.encrypt.data.transfer.cipher.key.bitlength	The key bitlength negotiated by dfsclient and datanode for encryption. This value may be set to either `128`, `192`, or `256`	128
ignore.secure.ports.for.testing	Allows skipping HTTPS requirements in the SASL mode	false
dfs.client.https.need-auth	Whether SSL client certificate authentication is required	false

httpfs-site.xml

Parameter	Description	Default value

httpfs.http.administrators	The ACL for the admins. This configuration is used to control who can access the default servlets for HttpFS server. The value should be a comma-separated list of users and groups. The user list comes first and is separated by a space, followed by the group list, for example: `user1,user2 group1,group2`. Both users and groups are optional, so you can define only users, or groups, or both of them. Notice that in all these cases you should always use the leading space in the groups list. Using the asterisk grants access to all users and groups	*
hadoop.http.temp.dir	The HttpFS temp directory	${hadoop.tmp.dir}/httpfs
httpfs.ssl.enabled	Defines whether SSL is enabled. Default is `false`, that is disabled	false
httpfs.hadoop.config.dir	The location of the Hadoop configuration directory	/etc/hadoop/conf
httpfs.hadoop.authentication.type	Defines the authentication mechanism used by httpfs for its HTTP clients. Valid values are `simple` and `kerberos`. If `simple` is used, clients must specify the username with the `user.name` query string parameter. If `kerberos` is used, HTTP clients must use HTTP SPNEGO or delegation tokens	simple
httpfs.hadoop.authentication.kerberos.keytab	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by httpfs in the HTTP endpoint. `httpfs.authentication.kerberos.keytab` is deprecated. Instead, use `hadoop.http.authentication.kerberos.keytab`	/etc/security/keytabs/httpfs.service.keytab
httpfs.hadoop.authentication.kerberos.principal	The HTTP Kerberos principal used by HttpFS in the HTTP endpoint. The HTTP Kerberos principal MUST start with `HTTP/` as per Kerberos HTTP SPNEGO specification. `httpfs.authentication.kerberos.principal` is deprecated. Instead, use `hadoop.http.authentication.kerberos.principal`	HTTP/${httpfs.hostname}@${kerberos.realm}

ranger-hdfs-audit.xml

Parameter	**Description**	**Default value**
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/hdfs_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Uses in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal's secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—

xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service that wants to log in	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal's key to be stored in the subject's private credentials	false
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal's key from the keytab	false

ranger-hdfs-security.xml

Parameter	Description	Default value
ranger.plugin.hdfs.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.hdfs.service.name	The name of the Ranger service containing policies for this instance	—
ranger.plugin.hdfs.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	/srv/ranger/hdfs/policycache
ranger.plugin.hdfs.policy.pollIntervalMs	Defines how often to poll for changes in policies	30000
ranger.plugin.hdfs.policy.rest.client.connection.timeoutMs	The HDFS Plugin RangerRestClient connection timeout (in milliseconds)	120000
ranger.plugin.hdfs.policy.rest.client.read.timeoutMs	The HDFS Plugin RangerRestClient read timeout (in milliseconds)	30000
ranger.plugin.hdfs.policy.rest.ssl.config.file	The path to the RangerRestClient SSL config file for the HDFS plugin	/etc/hadoop/conf/ranger-hdfs-policymgr-ssl.xml

httpfs-env.sh

Parameter	Description	Default value
Sources	A list of sources which will be written into *httpfs-env.sh*	—
HADOOP_CONF_DIR	Hadoop configuration directory	/etc/hadoop/conf
HADOOP_LOG_DIR	Location of the log directory	${HTTPFS_LOG}
HADOOP_PID_DIR	PID file directory location	${HTTPFS_TEMP}
HTTPFS_SSL_ENABLED	Defines if SSL is enabled for httpfs	false
HTTPFS_SSL_KEYSTORE_FILE	The path to the keystore file	admin
HTTPFS_SSL_KEYSTORE_PASS	The password to access the keystore	admin
Final HTTPFS_ENV_OPTS	Final value of the `HTTPFS_ENV_OPTS` parameter in *httpfs-env.sh*	—

hadoop-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *hadoop-env.sh*	—
HDFS_NAMENODE_OPTS	NameNode Heap Memory. Sets initial (-Xms) and maximum (-Xmx) Java heap memory size and environment options for the NameNode	-Xms1G -Xmx8G
HDFS_DATANODE_OPTS	DataNode Heap Memory. Sets initial (-Xms) and maximum (-Xmx) Java heap memory size and environment options for the DataNode	-Xms700m -Xmx8G
HDFS_HTTPFS_OPTS	HttpFS Heap Memory. Sets initial (-Xms) and maximum (-Xmx) Java heap memory size and environment options for the httpfs server	-Xms700m -Xmx8G
HDFS_JOURNALNODE_OPTS	JournalNode Heap Memory. Sets initial (-Xms) and maximum (-Xmx) Java heap memory size and environment options for the JournalNode	-Xms700m -Xmx8G
HDFS_ZKFC_OPTS	ZKFC Heap Memory. Sets initial (-Xms) and maximum (-Xmx) Java heap memory size and environment options for ZKFC	-Xms500m -Xmx8G
Final HADOOP_ENV_OPTS	Final value of the `HADOOP_ENV_OPTS` parameter in *hadoop-env.sh*	—

ssl-server.xml

Parameter	Description	Default value
ssl.server.truststore.location	The truststore to be used by NameNodes and DataNodes	—
ssl.server.truststore.password	The password to the truststore	—
ssl.server.truststore.type	The truststore file format	jks
ssl.server.truststore.reload.interval	The truststore reload check interval (in milliseconds)	10000
ssl.server.keystore.location	The path to the keystore file used by NameNodes and DataNodes	—
ssl.server.keystore.password	The password to the keystore	—
ssl.server.keystore.keypassword	The password to the key in the keystore	—
ssl.server.keystore.type	The keystore file format	—

Lists of decommissioned and in maintenance hosts

Parameter	Description	Default value
DECOMMISSIONED	When an administrator decommissions a DataNode, the DataNode will first be transitioned into `DECOMMISSION_INPROGRESS` state. After all blocks belonging to that DataNode are fully replicated elsewhere based on each block replication factor, the DataNode will be transitioned to `DECOMMISSIONED` state. After that, the administrator can shutdown the node to perform long-term repair and maintenance that could take days or weeks. After the machine has been repaired, the machine can be recommissioned back to the cluster	—

IN_MAINTENANCE	Sometimes administrators only need to take DataNodes down for minutes/hours to perform short-term repair/maintenance. For such scenarios, the HDFS block replication overhead, incurred by decommission, might not be necessary and a light-weight process is desirable. And that is what maintenance state is used for. When an administrator puts a DataNode in the maintenance state, the DataNode will first be transitioned to `ENTERING_MAINTENANCE` state. As long as all blocks belonging to that DataNode, are minimally replicated elsewhere, the DataNode will immediately be transitioned to `IN_MAINTENANCE` state. After the maintenance has completed, the administrator can take the DataNode out of the maintenance state. In addition, maintenance state supports the timeout that allows administrators to configure the maximum duration, in which a DataNode is allowed to stay in the maintenance state. After the timeout, the DataNode will be transitioned out of maintenance state automatically by HDFS without human intervention	—

Other

Parameter	Description	Default value
Additional nameservices	Additional (internal) names for an HDFS cluster that allows querying another HDFS cluster from the current one	—
Custom core-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *core-site.xml*	—
Custom hdfs-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hdfs-site.xml*	—
Custom httpfs-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *httpfs-site.xml*	—
Ranger plugin enabled	Whether or not Ranger plugin is enabled	—
Custom ranger-hdfs-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hdfs-audit.xml*	—
Custom ranger-hdfs-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hdfs-security.xml*	—
Custom ranger-hdfs-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hdfs-policymgr-ssl.xml*	—
Custom httpfs-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *httpfs-env.sh*	—
Custom hadoop-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hadoop-env.sh*	—
Custom ssl-server.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ssl-server.xml*	—
Custom ssl-client.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ssl-client.xml*	—

Topology script	The topology script used in HDFS	—
Topology data	An otional text file to map host names to the rack number for topology script. Stored to */etc/hadoop/conf/topology.data*	—
Custom log4j.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *log4j.properties*	log4j.properties
Custom httpfs-log4j.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *httpfs-log4j.properties*	httpfs-log4j.properties

HDFS DataNode component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display HDFS DataNode metrics in the Prometheus format	9202
Mapping config path	Path to the metrics mapping configuration file	/etc/hadoop/conf/jmx_hdfs_datanode_metric_config.yml
Mapping config	Metrics mapping configuration file	hdfs-mapping-config.yml

HDFS JournalNode component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display HDFS JournalNode metrics in the Prometheus format	9203
Mapping config path	Path to the metrics mapping configuration file	/etc/hadoop/conf/jmx_hdfs_journalnode_metric_config.yml
Mapping config	Metrics mapping configuration file	hdfs-mapping-config.yml

HDFS NameNode component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display HDFS NameNode metrics in the Prometheus format	9201

Mapping config path	Path to the metrics mapping configuration file	/etc/hadoop/conf/jmx_hdfs_namenode_metric_config.yml
Mapping config	Metrics mapping configuration file	hdfs-mapping-config.yml

Hive

hive-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *hive-env.sh*	—
HADOOP_CLASSPATH	A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ /etc/tez/conf/ ▪ /usr/lib/tez/* ▪ /usr/lib/tez/lib/*
HIVE_HOME	Hive home directory	/usr/lib/hive
METASTORE_PORT	Hive Metastore port	9083
HADOOP_CLIENT_OPTS	Hadoop client options. For example, JVM startup parameters	$HADOOP_CLIENT_OPTS -Djava.io.tmpdir={{ cluster.config.java_tmpdir | d('/tmp') }}

hive-server2-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *hive-server2-env.sh*	—
HADOOP_CLIENT_OPTS	Hadoop client options for HiveServer2	-Xms256m -Xmx256m
HIVE_AUX_JARS_PATH	Allows including custom JAR files to the Hive's classpath. A list of files/directories to be added to the classpath. To add more items to the classpath, click +	—
Final HIVE_SERVER2_ENV_OPTS	Final value of the `HIVE_SERVER2_ENV_OPTS` parameter in *hive-server2-env.sh*	—

hive-metastore-env.sh

Parameter	Description	Default value
Sources	A list of sources that will be written into *hive-metastore-env.sh*	—
HADOOP_CLIENT_OPTS	Hadoop client options for Hive Metastore	-Xms256m -Xmx256m
Final HIVE_METASTORE_ENV_OPTS	Final value of the `HIVE_METASTORE_ENV_OPTS` parameter in *hive-metastore-env.sh*	—

Credential Encryption

Parameter	Description	Default value
Encryption enable	Enables or disables the credential encryption feature. When enabled, Hive stores configuration passwords and credentials required for interacting with other services in the encrypted form	false
Credential provider path	Path to a keystore file with secrets	jceks://file/etc/hive/conf/hive.jceks
Ranger plugin credential provider path	Path to a Ranger keystore file with secrets	jceks://file/etc/hive/conf/ranger-hive.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the default auto-generated JCEKS file	false
Password file name	Name of the file in the service's classpath that stores passwords	hive_credstore_pass

hive-site.xml

Parameter	Description	Default value
hive.cbo.enable	When set to `true`, enables the cost-based optimizer that uses the Calcite framework	true
hive.compute.query.using.stats	When set to `true`, Hive will answer a few queries like `min`, `max`, and `count(1)` purely using statistics stored in the Metastore. For basic statistics collection, set the configuration property `hive.stats.autogather` to `true`. For more advanced statistics collection, run the `ANALYZE TABLE` queries	false
hive.execution.engine	Selects the execution engine. Supported values are: `mr` (Map Reduce, default), `tez` (Tez execution, for Hadoop 2 only), or `spark` (Spark execution, for Hive 1.1.0 onward)	Tez
hive.log.explain.output	When enabled, logs the `EXPLAIN EXTENDED` output for the query at log4j INFO level and in the HiveServer2 web UI (***Drilldown*** → ***Query Plan***). Starting Hive 3.1.0, this configuration property only logs as the log4j INFO. To log the `EXPLAIN EXTENDED` output in WebUI/Drilldown/Query Plan in Hive 3.1.0 and later, use `hive.server2.webui.explain.output`	true
hive.metastore.event.db.notification.api.auth	Defines whether the Metastore should perform the authorization against database notification related APIs such as `get_next_notification`. If set to `true`, then only the superusers in proxy settings have the permission	false
hive.metastore.uris	Metastore URI used to access metadata in a remote metastore setup. For a remote metastore, you should specify the Thrift metastore server URI: *thrift://<hostname>:<port>* where *<hostname>* is a name or IP address of the Thrift metastore server, *<port>* is the port, on which the Thrift server is listening	—
hive.metastore.warehouse.dir	Absolute HDFS file path of the default database for the warehouse, that is local to the cluster	/apps/hive/warehouse
hive.server2.enable.doAs	Impersonate the connected user	false
hive.stats.fetch.column.stats	Annotation of the operator tree with statistics information requires column statistics. Column statistics are fetched from the Metastore. Fetching column statistics for each needed column can be expensive, when the number of columns is high. This flag can be used to disable fetching of column statistics from the Metastore	false
hive.tez.container.size	By default, Tez will spawn containers of the size of a mapper. This parameter can be used to overwrite the default value	1024

hive.support.concurrency	Defines whether Hive should support concurrency or not. A ZooKeeper instance must be up and running for the default Hive Lock Manager to support read/write locks	false
hive.txn.manager	Set this to `org.apache.hadoop.hive.ql.lockmgr.DbTxnManager` as part of turning on Hive transactions. The default DummyTxnManager replicates pre-Hive-0.13 behavior and provides no transactions	org.apache.hadoop.hive.ql.lockmgr.DummyTxnManager
hive.users.in.admin.role	A comma-separated list of users to be assigned the administrator role when the Metastore starts	—
javax.jdo.option.ConnectionUserName	Metastore database user name	APP
javax.jdo.option.ConnectionPassword	Password for the metastore user name	—
javax.jdo.option.ConnectionURL	JDBC connection URI used to access the data stored in the local Metastore setup. Use the following connection URI: *jdbc:<datastore type>://<node name>:<port>/<database name>* where: - *<node name>* is the host name or IP address of the data store; - *<data store type>* is the type of the data store; - *<port>* is the port on which the data store listens for remote procedure calls (RPC); - *<database name>* is the name of the database. For example, the following URI specifies a local metastore that uses MySQL as a data store: `jdbc:mysql://hostname23:3306/metastore`	jdbc:postgresql://{{ groups['adpg.adpg'][0] | d(omit) }}:5432/hive
javax.jdo.option.ConnectionDriverName	JDBC driver class name used to access Hive Metastore	org.postgresql.Driver
hive.server2.transport.mode	Sets the transport mode	binary
hive.server2.thrift.port	Port number used for the binary connection with Thrift Server2	10000
hive.server2.thrift.http.port	Port number used for the HTTP connection with Thrift Server2	10001
hive.server2.thrift.http.path	HTTP endpoint of the Thrift Server2 service	cliservice
hive.metastore.transactional.event.listeners	Listener class that stores events in a database	org.apache.hive.hcatalog.listener.DbNotificationListener
hive.metastore.dml.events	Indicates whether Hive should track DML events	true
hive.server2.authentication.kerberos.principal	HiveServer2 Kerberos principal	—
hive.server2.authentication.kerberos.keytab	Path to the Kerberos keytab file containing the HiveServer2 principal	—
hive.server2.authentication.spnego.principal	SPNEGO Kerberos principal	—
hive.server2.webui.spnego.principal	SPNEGO Kerberos principal to access Web UI	—
hive.server2.webui.spnego.keytab	SPNEGO Kerberos keytab file to access Web UI	—
hive.server2.webui.use.spnego	Defines whether to use Kerberos SPNEGO for Web UI access	false

hive.server2.authentication.spnego.keytab	Path to SPNEGO principal	—
hive.server2.authentication	Sets the authentication mode	NONE
hive.metastore.sasl.enabled	If `true`, the Metastore Thrift interface will be secured with SASL. Clients must authenticate with Kerberos	false
hive.metastore.kerberos.principal	Service principal for the metastore Thrift server. The `_HOST` token will be automatically replaced with the appropriate host name	—
hive.metastore.kerberos.keytab.file	Path to the Kerberos keytab file containing the metastore Thrift server’s service principal	—
hive.server2.use.SSL	Defines whether to use SSL for HiveServer2	false
hive.server2.keystore.path	Keystore to be used by HiveServer2	—
hive.server2.keystore.password	Password to the HiveServer2 keystore	—
hive.server2.truststore.path	Truststore to be used by HiveServer2	—
hive.server2.truststore.password	Password to the HiveServer2 truststore	—
hive.server2.webui.use.ssl	Defines whether to use SSL for the Hive web UI	false
hive.server2.webui.keystore.path	Path to the keystore file used to access the Hive web UI	—
hive.server2.webui.keystore.password	Password to the keystore file used to access the Hive web UI	—
hive.ssl.protocol.blacklist	A comma-separated list of TLS versions that cannot be used by Hive	SSLv2Hello,SSLv3,TLSv1,TLSv1.1
metastore.keystore.path	Path to the Hive Metastore keystore	—
metastore.keystore.password	Password to the Hive Metastore keystore	—
metastore.truststore.path	Path to the Hive Metastore truststore	—
metastore.truststore.password	Password to the Hive Metastore truststore	—
metastore.use.SSL	Defines whether to use SSL for interaction with Hive Metastore	false
metastore.ssl.protocol.blacklist	A comma-separated list of TLS versions that cannot be used for communication with Hive Metastore	SSLv2Hello,SSLv2,SSLv3,TLSv1,TLSv1.1
iceberg.engine.hive.enabled	Enables Iceberg tables support	true
hive.security.authorization.sqlstd.confwhitelist.append	A regex to append configuration properties to the white list in addition to `hive.security.authorization.sqlstd.confwhitelist`	kyuubi\.operation\.handle|kyuubi\.client\.version|kyuubi\.client\.ipAddress|tez\.application\.tags
hive.server2.support.dynamic.service.discovery	Defines whether to support dynamic service discovery via ZooKeeper	true

hive.zookeeper.quorum	A comma-separated list of ZooKeeper servers (<host>:<port>) running in the cluster	—
hive.server2.zookeeper.namespace	Specifies the root namespace on ZooKeeper	hiveserver2
hive.cluster.delegation.token.store.class	Name of the class that implements the delegation token store system	org.apache.hadoop.hive.metastore.security.ZooKeeperTokenStore

ACID Housekeeping

Parameter	Description	Default value
Enable Housekeeping Thread	Defines whether background housekeeping tasks are performed by the Hive Metastore	true
Cleaner Interval	Frequency at which the background cleaning threads for ACID transactions are executed	10s
Leader Election Mode	Defines whether to use the host specified by the `Housekeeping Leader Hostname` parameter as the housekeeping leader (the `host` option). The other option is `lock` which means that Hive Metastore must use Hive locks to elect a leader	host
Housekeeping Leader Hostname	FQDN of the housekeeping leader	—
ZK Lock Namespace	Specifies the lock namespace on ZooKeeper	—
Compactor Initiator	Defines whether to start the `compaction-initiator` thread	true
Compactor Cleaner	Defines whether to start the `compaction-cleaner` thread	true

Custom log4j.properties

Parameter	Description	Default value
HiveServer2 hive-log4j.properties	Log4j configuration used for logging HiveServer2's activity	hive-log4j.properties
Hive Metastore hive-log4j2.properties	Log4j2 configuration used for logging Hive Metastore's activity	hive-log4j2.properties
Hive Beeline beeline-log4j2.properties	Log4j2 configuration used for logging Hive Beeline's activity	beeline-log4j2.properties

ranger-hive-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/hdfs_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—

xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Uses in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal's secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—
xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service that wants to log in	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal's key to be stored in the subject's private credentials	false
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal's key from the keytab	false

ranger-hive-security.xml

Parameter	Description	Default value
ranger.plugin.hive.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.hive.service.name	The name of the Ranger service containing policies for this instance	—
ranger.plugin.hive.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	/srv/ranger/hive/policycache
ranger.plugin.hive.policy.pollIntervalMs	Defines how often to poll for changes in policies	30000
ranger.plugin.hive.policy.rest.client.connection.timeoutMs	Hive Plugin RangerRestClient connection timeout (in milliseconds)	120000
ranger.plugin.hive.policy.rest.client.read.timeoutMs	Hive Plugin RangerRestClient read timeout (in milliseconds)	30000
xasecure.hive.update.xapolicies.on.grant.revoke	Controls Hive Ranger policy update from SQL Grant/Revoke commands	true
ranger.plugin.hive.policy.rest.ssl.config.file	Path to the RangerRestClient SSL config file for the Hive plugin	/etc/hive/conf/ranger-hive-policymgr-ssl.xml

ranger-hive-policymgr-ssl.xml

Parameter	Description	Default value

xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/hive/conf/ranger-hive.jceks
xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/hive/conf/ranger-hive.jceks
xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

tez-site.xml

Parameter	**Description**	**Default value**
tez.am.resource.memory.mb	Amount of memory in MB, that YARN will allocate to the Tez Application Master. The size increases with the size of the DAG	1024
tez.history.logging.service.class	Enables Tez to use the Timeline Server for History Logging	org.apache.tez.dag.history.logging.ats.ATSHistoryLoggingService
tez.lib.uris	HDFS paths containing the Tez JAR files	${fs.defaultFS}/apps/tez/tez-0.10.3.tar.gz
tez.task.resource.memory.mb	Amount of memory used by launched tasks in TEZ containers. Usually this value is set in the DAG	1024
tez.tez-ui.history-url.base	URL where the Tez UI is hosted	—
tez.use.cluster.hadoop-libs	Specifies, whether Tez will use the cluster Hadoop libraries	true

nginx.conf

Parameter	**Description**	**Default value**
ssl_certificate	Path to the SSL certificate for Nginx	/etc/ssl/certs/host_cert.cert
ssl_certificate_key	Path to the SSL certificate key for Nginx	/etc/ssl/host_cert.key
ssl_protocols	A list of allowed TLS protocols to set up SSL connection	TLSv1.2
nginx_http_port	Nginx HTTP port	8089
nginx_https_port	Nginx HTTPS port	9999

Other

Parameter	**Description**	**Default value**

ACID Transactions	Defines whether to enable ACID transactions	false
Database type	Type of the external database used for Hive Metastore	postgres
Custom hive-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hive-site.xml*	—
Custom hive-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hive-env.sh*	—
Ranger plugin enabled	Whether or not Ranger plugin is enabled	false
Custom ranger-hive-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hive-audit.xml*	—
Custom ranger-hive-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hive-security.xml*	—
Custom ranger-hive-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hive-policymgr-ssl.xml*	—
Custom tez-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *tez-site.xml*	—
beeline-hs2-connection.xml	An XML template to generate property-value pairs from the `hive_beeline_connection_conf` object	beeline-hs2-connection.xml

Hive HiveServer2 component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display Hive HiveServer2 metrics in the Prometheus format	9208
Mapping config path	Path to the metrics mapping configuration file	/etc/hive/conf/jmx_hive_server_metric_config.yml
Mapping config	Metrics mapping configuration file	hive-mapping-config.yml

Hive Metastore component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display Hive Metastore metrics in the Prometheus format	9207

Mapping config path	Path to the metrics mapping configuration file	/etc/hive/conf/jmx_hive_server_metric_config.yml
Mapping config	Metrics mapping configuration file	hive-mapping-config.yml

HUE

The HUE Server component

hue.ini syntax

The *hue.ini* configuration file displayed in ADCM has different syntax from its original syntax. In the original file, the nesting level is determined by placing the section names into the corresponding number of square brackets. Example:

```
[notebook]
show_notebooks=true
[[interpreters]]
[[[mysql]]]
name = MySQL
interface=sqlalchemy
options='{"url": "mysql://root:secret@database:3306/hue"}'
[[[hive]]]
name=Hive
interface=hiveserver2
```

In ADCM, the nesting level is determined by separating the section names with periods. The structure from the above example will look the following way:

```
notebook.show_notebooks: true
notebook.interpreters.mysql.name: MySQL
notebook.interpreters.mysql.interface: sqlalchemy
notebook.interpreters.mysql.options: '{"url": "mysql://root:secret@database:3306/hue"}'
notebook.interpreters.hive.name: Hive
notebook.interpreters.hive.interface: hiveserver2
```

hue.ini

Parameter	Description	Default value
desktop.enable_prometheus	Defines whether Prometheus metrics are enabled	false
desktop.http_host	HUE Server listening IP address	0.0.0.0
desktop.http_port	HUE Server listening port	8000
desktop.use_cherrypy_server	Defines whether CherryPy (`true`) or Gunicorn (`false`) is used as the webserver	false
desktop.gunicorn_work_class	Gunicorn work class: `gevent`, `eventlet`, `gthread`, or `sync`	gthread
desktop.secret_key	Random string used for secure hashing in the session store	jFE93j;2[290-eiw.KEiwN2s3['d;/.q[eIW^y#e=+Iei*@Mn<qW5o
desktop.enable_xff_for_hive_impala	Defines whether the `X-Forwarded-For` header is used if Hive or Impala require it	false
desktop.enable_x_csrf_token_for_hive_impala	Defines whether the `X-CSRF-Token` header is used if Hive or Impala require it	false
desktop.app_blacklist	Comma-separated list of apps to not load at server startup	security,pig,sqoop,oozie,hbase,search

desktop.auth.backend	Comma-separated list of authentication backend combinations in order of priority	desktop.auth.backend.AllowFirstUserDjangoBackend
desktop.database.host	HUE Server database network or IP address	{{ groups['adpg.adpg'][0] | d(omit) }}
desktop.database.port	HUE Server database network port	5432
desktop.database.engine	Engine used by the HUE Server database	postgresql_psycopg2
desktop.database.user	Admin username for the HUE Server database	hue
desktop.database.name	HUE Server database name	hue
desktop.database.password	Password for the `desktop.database.user` username	—
desktop.auth_username	Username for authentication in HUE UI	—
desktop.auth_password	Password for authentication in HUE UI	—

Interpreter Impala

Parameter	Description	Default value
notebook.interpreters.impala.name	Impala interpreter name	impala
notebook.interpreters.impala.interface	Interface for the Impala interpreter	hiveserver2
impala.server_host	Host of the Impala Server (one of the Impala Daemon components)	—
impala.server_port	Port of the Impala Server	21050
impala.impersonation_enabled	Enables the impersonation mechanism during interaction with Impala	true
impala.impala_conf_dir	Path to the Impala configuration directory that contains the *impalad_flags* file	/etc/hue/conf
impala.ssl.cacerts	Path to the CA certificates	/etc/pki/tls/certs/ca-bundle.crt
impala.ssl.validate	Defines whether HUE should validate certificates received from the server	false
impala.ssl.enabled	Enables SSL communication for this server	false
impala.impala_principal	Kerberos principal name for Impala	—
impala.auth_username	Username for authentication in Impala	—
impala.auth_password	Password for authentication in Impala	—

Interpreter HDFS

Parameter	Description	Default value
hadoop.hdfs_clusters.default.webhdfs_url	WebHDFS or HttpFS endpoint link for accessing HDFS data	—
hadoop.hdfs_clusters.default.hadoop_conf_dir	Path to the directory of the Hadoop configuration files	/etc/hadoop/conf
hadoop.hdfs_clusters.default.security_enabled	Defines whether the Hyperwave cluster is secured by Kerberos	false

hadoop.hdfs_clusters.default.ssl_cert_ca_verify	Defines whether to verify SSL certificates against the CA	false

Interpreter Hive

Parameter	Description	Default value
notebook.interpreters.hive.name	Hive interpreter name	hive
notebook.interpreters.hive.interface	Interface for the Hive interpreter	hiveserver2
beeswax.hive_discovery_hs2	Defines whether to use service discovery for HiveServer2	true
beeswax.hive_conf_dir	Path to the Hive configuration directory containing the *hive-site.xml* file	/etc/hive/conf
beeswax.use_sasl	Defines whether to use the SASL framework to establish connection to host	true
beeswax.hive_discovery_hiveserver2_znode	Hostname of the znode of the HiveServer2 if Hive is using ZooKeeper service discovery mode	hive.server2.zookeeper.namespace
libzookeeper.ensemble	List of ZooKeeper ensemble members hosts and ports	host1:2181,host2:2181,host3:2181
libzookeeper.principal_name	Kerberos principal name for ZooKeeper	—
beeswax.auth_username	Username for authentication in Hive	—
beeswax.auth_password	Password for authentication in Hive	—

Interpreter YARN

Parameter	Description	Default value
hadoop.yarn_clusters.default.resourcemanager_host	Network address of the host where the Resource Manager is running	—
hadoop.yarn_clusters.default.resourcemanager_port	Port listened by the Resource Manager IPC	8031
hadoop.yarn_clusters.default.submit_to	Defines whether the jobs are submitted to this cluster	true
hadoop.yarn_clusters.default.logical_name	Resource Manager logical name (required for High Availability mode)	—
hadoop.yarn_clusters.default.security_enabled	Defines whether the YARN cluster is secured by Kerberos	false
hadoop.yarn_clusters.default.ssl_cert_ca_verify	Defines whether to verify the SSL certificates from YARN Rest APIs against the CA when using the secure mode (HTTPS)	false
hadoop.yarn_clusters.default.resourcemanager_api_url	URL of the Resource Manager API	—
hadoop.yarn_clusters.default.proxy_api_url	URL of the first Resource Manager API	—
hadoop.yarn_clusters.default.history_server_api_url	URL of the History Server API	—

hadoop.yarn_clusters.default.spark_history_server_url	URL of the Spark History Server	—
hadoop.yarn_clusters.default.spark_history_server_security_enabled	Defines whether the Spark History Server is secured by Kerberos	false
hadoop.yarn_clusters.ha.resourcemanager_host	Network address of the host where the Resource Manager is running (High Availability mode)	—
hadoop.yarn_clusters.ha.resourcemanager_port	Port listened by the Resource Manager IPC (High Availability mode)	—
hadoop.yarn_clusters.ha.logical_name	Resource Manager logical name (required for High Availability mode)	—
hadoop.yarn_clusters.ha.security_enabled	Defines whether the YARN cluster is secured by Kerberos (High Availability mode)	false
hadoop.yarn_clusters.ha.submit_to	Defines whether the jobs are submitted to this cluster (High Availability mode)	true
hadoop.yarn_clusters.ha.ssl_cert_ca_verify	Defines whether to verify the SSL certificates from YARN Rest APIs against the CA when using the secure mode (HTTPS) (High Availability mode)	false
hadoop.yarn_clusters.ha.resourcemanager_api_url	URL of the Resource Manager API (High Availability mode)	—
hadoop.yarn_clusters.ha.history_server_api_url	URL of the History Server API (High Availability mode)	—
hadoop.yarn_clusters.ha.spark_history_server_url	URL of the Spark History Server (High Availability mode)	—
hadoop.yarn_clusters.ha.spark_history_server_security_enabled	Defines whether the Spark History Server is secured by Kerberos (High Availability mode)	false

Interpreter Kyuubi

Parameter	Description	Default value
notebook.dbproxy_extra_classpath	Classpath to be appended to the default DBProxy server classpath	/usr/share/java/kyuubi-hive-jdbc.jar
notebook.interpreters.kyuubi.name	Kyuubi interpreter name	Kyuubi[Spark3]
notebook.interpreters.kyuubi.options	Special parameters for connection to the Kyuubi server	—
notebook.interpreters.kyuubi.interface	Interface for the Kyuubi service	jdbc

Interpreter Trino

Parameter	Description	Default value
notebook.interpreters.trino.name	Trino interpreter name	Trino
notebook.interpreters.trino.interface	Interface for the Trino service	trino

notebook.interpreters.trino.options	Special parameters for connection to the Trino server	

```json
{
    "url": "",
    "auth_type": "basic",
    "kerberos_principal": "",
    "kerberos_service_name": "HTTP",
    "kerberos_force_preemptive": true,
    "kerberos_delegate": true,
    "ssl_cert_ca_verify": false
}
```

Interpreter Ozone

Parameter	Description	Default value
desktop.ozone.default.webhdfs_url	WebHDFS or HttpFS endpoint for accessing HDFS data	—
desktop.ozone.default.ozone_conf_dir	Path to the Ozone configuration directory	/etc/ozone/conf
desktop.ozone.default.security_enabled	Defines whether the Ozone cluster is secured by Kerberos	false
desktop.ozone.default.ssl_cert_ca_verify	Defines whether to verify SSL certificates against the CA	false
desktop.ozone.default.fs_defaultfs	Ozone service ID	—

hue.ini kerberos config

Parameter	Description	Default value
desktop.kerberos.hue_keytab	Path to HUE Kerberos keytab file	—
desktop.kerberos.hue_principal	Kerberos principal name for HUE	—
desktop.kerberos.kinit_path	Path to `kinit` utility	/usr/bin/kinit
desktop.kerberos.reinit_frequency	Time interval in seconds for HUE to renew its keytab	3600
desktop.kerberos.ccache_path	Path to cached Kerberos credentials	/tmp/hue_krb5_ccache
desktop.kerberos.krb5_renewlifetime_enabled	This must be set to `false` if the `renew_lifetime` parameter in *krb5.conf* file is set to `0m`	false
desktop.auth.auth	Authentication type	—

Authentication on WEB UIs

Parameter	Description	Default value
desktop.kerberos.kerberos_auth	Defines whether to use Kerberos authentication for HTTP clients based on the current ticket	false

desktop.kerberos.spnego_principal	Default Kerberos principal name for the HTTP client	—

hue.ini SSL config

Parameter	Description	Default value
desktop.ssl_certificate	Path to the SSL certificate file	/etc/ssl/certs/host_cert.cert
desktop.ssl_private_key	Path to the SSL RSA private key file	/etc/ssl/host_cert.key
desktop.ssl_password	SSL certificate password	—
desktop.ssl_no_renegotiation	Disables all renegotiation in TLSv1.2 and earlier	true
desktop.ssl_validate	Defines whether HUE should validate certificates received from the server	false
desktop.ssl_cacerts	This must be set to `false` if the `renew_lifetime` parameter in *krb5.conf* file is set to `0m`	/etc/pki/tls/certs/ca-bundle.crt
desktop.session.secure	Defines whether the cookie containing the user's session ID and csrf cookie will use the `secure` flag	true
desktop.session.http_only	Defines whether the cookie containing the user's session ID and csrf cookie will use the `HTTP only` flag	false

LDAP security

Parameter	Description	Default value
desktop.ldap.ldap_url	URL of the LDAP server	—
desktop.ldap.base_dn	The search base for finding users and groups	"DC=mycompany,DC=com"
desktop.ldap.nt_domain	The NT domain used for LDAP authentication	mycompany.com
desktop.ldap.ldap_cert	Certificate files in PEM format for the CA that HUE will trust for authentication over TLS	—
desktop.ldap.use_start_tls	Set this to `true` if you are not using Secure LDAP (LDAPS) but want to establish secure connections using TLS	true
desktop.ldap.bind_dn	Distinguished name of the user to bind as	"CN=ServiceAccount,DC=mycompany,DC=com"
desktop.ldap.bind_password	Password of the bind user	—
desktop.ldap.ldap_username_pattern	Pattern for username search. Specify the `<username>` placeholder for this parameter	"uid=<username>,ou=People,dc=mycompany,dc=com"
desktop.ldap.create_users_on_login	Defines whether to create users in HUE when they try to login with their LDAP credentials	true
desktop.ldap.sync_groups_on_login	Defines whether to synchronize users groups when they login	true
desktop.ldap.login_groups	A comma-separated list of LDAP groups containing users that are allowed to login	—
desktop.ldap.ignore_username_case	Defines whether to ignore the case of usernames when searching for existing users	true
desktop.ldap.force_username_lowercase	Defines whether to force use lowercase for usernames when creating new users from LDAP	true

desktop.ldap.force_username_uppercase	Defines whether to force use uppercase for usernames when creating new users from LDAP. This parameter cannot be combined with `desktop.ldap.force_username_lowercase`	false
desktop.ldap.search_bind_authentication	Enables search bind authentication	true
desktop.ldap.subgroups	Specifies the kind of subgrouping to use: nested or suboordinate (deprecated)	nested
desktop.ldap.nested_members_search_depth	The number of levels to search for nested members	10
desktop.ldap.follow_referrals	Defines whether to follow referrals	false
desktop.ldap.users.user_filter	Base filter for users search	"objectclass=*"
desktop.ldap.users.user_name_attr	The username attribute in the LDAP schema	sAMAccountName
desktop.ldap.groups.group_filter	Base filter for groups search	"objectclass=*"
desktop.ldap.groups.group_name_attr	The group name attribute in the LDAP schema	cn
desktop.ldap.groups.group_member_attr	The attribute of the group object that identifies the group members	member

Others

Parameter	**Description**	**Default value**

Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	`[Service]`
		`DefaultLimitCPU=`
		`DefaultLimitFSIZE=`
		`DefaultLimitDATA=`
		`DefaultLimitSTACK=`
		`DefaultLimitCORE=`
		`DefaultLimitRSS=`
		`DefaultLimitNOFILE=`
		`DefaultLimitAS=`
		`DefaultLimitNPROC=`
		`DefaultLimitMEMLOCK=`
		`DefaultLimitLOCKS=`
		`DefaultLimitSIGPENDING=`
		`DefaultLimitMSGQUEUE=`
		`DefaultLimitNICE=`
		`DefaultLimitRTPRIO=`
		`DefaultLimitRTTIME=`

Custom hue.ini	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *hue.ini*. List of available parameters can be found in the HUE documentation	—
log.conf	A configuration file with definitions for various logging objects	log.conf

Custom impalad_flags	Custom parameter values that overwrite the original ones	—

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process’s data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

Impala

Parameter	Description	Default value
impala-env.sh	The contents of the *impala-env.sh* file that contains Impala environment settings	impala-env.sh
Custom impala-env.sh	The contents of the custom *impala-env.sh* file that contains custom Impala environment settings	custom-impala-env.sh

Credential encryption

Parameter	Description	Default value
Encryption enable	Defines whether the credentials are encrypted	false

Credential provider path	Path to the credential provider for creating the *.jceks* files containing secret keys	jceks://hdfs/apps/impala/security/impala.jceks
Ranger plugin credential provider path	Path to the Ranger plugin credential provider	jceks://file/etc/impala/conf/ranger-impala.jceks
Custom jceks	Defines whether custom *.jceks* files located at the credential provider path are used (`true`) or auto-generated ones (`false`)	false
Password file name	Name of the password file in the classpath of the service if the password file is selected in the credstore options	impala_credstore_pass

ranger-hive-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/hdfs_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Uses in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal's secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—
xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service that wants to log in	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal's key to be stored in the subject's private credentials	false
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal's key from the keytab	false

ranger-hive-security.xml

Parameter	Description	Default value
ranger.plugin.hive.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.hive.service.name	Name of the Ranger service containing policies for this Impala instance	—

ranger.plugin.hive.policy.cache.dir	Directory, where Ranger policies are cached after a successful retrieval from the source	/srv/ranger/impala/policycache
ranger.plugin.hive.policy.pollIntervalMs	How often to poll for changes in policies in milliseconds	30000
ranger.plugin.hive.policy.rest.client.connection.timeoutMs	Impala plugin connection timeout in milliseconds	120000
ranger.plugin.hive.policy.rest.client.read.timeoutMs	Impala plugin read timeout in milliseconds	30000
xasecure.hive.update.xapolicies.on.grant.revoke	Specifies whether the Impala plugin should update the Ranger policies on the updates to permissions done using `GRANT` / `REVOKE`	true
ranger.plugin.hive.policy.rest.ssl.config.file	The path to the RangerRestClient SSL config file for HBase plugin	/etc/hbase/conf/ranger-hbase-policymgr-ssl.xml

ranger-hive-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/impala/conf/ranger-impala.jceks
xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/impala/conf/ranger-impala.jceks
xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

Enable LDAP

Parameter	Description	Default value
ldap_uri	URI of the LDAP server. Typically, the URI is prefixed with `ldap://` or `ldaps://` for SSL-based LDAP transport. The URI can optionally specify the port, for example: `ldap://ldap_server.example.com:389`	—
ldap_domain	Replaces the username with a string `<username>@ldap_domain`, where `<username>` is the name of the user trying to authenticate. Mutually exclusive with `ldap_baseDN` and `ldap_bind_pattern`	—
ldap_bind_dn	Distinguished name of the user to bind to for user/group searches. Required only if the user or group filters are being used and the LDAP server is not configured to allow anonymous searches	—
ldap_bind_password	Password of the user to bind to for user/group searches. Required only if the anonymous bind is not activated	—

ldap_bind_password_cmd	A Unix command the output of which returns the password to use with the `--ldap_bind_dn` option. The output of the command will be truncated to 1024 bytes and trimmed of trailing whitespace.	cat /etc/impala/conf/pass.pwd
ldap_user_search_basedn	The base DN for the LDAP subtree to search users	—
ldap_group_search_basedn	The base DN for the LDAP subtree to search groups	—
ldap_baseDN	Search base. Replaces the username with a DN of the form: `uid=<userid>,ldap_baseDN`, where `<userid>` is the username of the user trying to authenticate. Mutually exclusive with `ldap_domain` and `ldap_bind_pattern`	—
ldap_user_filter	A filter for both simple and search bind mechanisms. For a simple bind, it is a comma-separated list of user names. If specified, users must be on this list for authentication to succeed. For a search bind, it is an LDAP filter that will be used during an LDAP search, it can contain the `{0}` pattern which will be replaced with the user name	—
ldap_group_filter	Comma-separated list of groups. If specified, users must belong to one of these groups for authentication to succeed	—
ldap_allow_anonymous_binds	When `true`, LDAP authentication with a blank password (an anonymous bind) is allowed by Impala	false
ldap_search_bind_authentication	Allows switching between the search and simple bind user lookup methods when authenticating	true
ldap_ca_certificate	Specifies the location of the certificate in standard PEM format for SSL. Store this certificate on the local filesystem, in a location that only the `impala` user and other trusted users can read	—
ldap_passwords_in_clear_ok	Enables the webserver to start with the LDAP authentication even if SSL is not enabled. If set to `true`, the `auth_creds_ok_in_clear` parameter in the *impalarc* file gets set to `true` as well. A potentially unsecure configuration	false
ldap_bind_pattern	A string in which the `#UID` instance is replaced with the user id. For example, if this parameter is set to `user=#UID,OU=foo,CN=bar` and the user `henry` tries to authenticate, the constructed bind name will be `user=henry,OU=foo,CN=bar`. Mutually exclusive with `ldap_domain` and `ldap_baseDN`	—
allow_custom_ldap_filters_with_kerberos_auth	Specifies whether to allow custom LDAP user and group filters even if Kerberos is enabled	true

Other

Parameter	Description	Default value
Ranger plugin enabled	Whether or not Ranger plugin is enabled	false
Custom ranger-hive-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hive-audit.xml*	—
Custom ranger-hive-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hive-security.xml*	—
Custom ranger-hive-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-hive-policymgr-ssl.xml*	—

The Impala Catalog Service component

catalogstore.conf

Parameter	Description	Default value
hostname	The hostname to use for the Catalog Service daemon. If Kerberos is enabled, it is also used as a part of the Kerberos principal. If this option is not set, the system default is used	—
state_store_host	The host where the Impala Statestore component is running	—
state_store_port	The port on which the Impala Statestore component is running	24000
catalog_service_host	The host where the Impala Catalog Service component is running	—
catalog_service_port	The port on which the Impala Catalog Service component listens	26000
enable_webserver	Enables or disables the Catalog Service web server. Its Web UI includes information about the databases, tables, and other objects managed by Impala, in addition to the resource usage and configuration settings of the Catalog Service	True
webserver_require_spnego	Enables the Kerberos authentication for Hadoop HTTP web consoles for all roles of this service using the SPNEGO protocol. Use this option only if Kerberos is enabled for the HDFS service	False
webserver_port	The port on which the Catalog Service web server is running	25020
log_dir	The directory where the Catalog Service daemon places its log files	/var/log/impala/catalogd/
log_filename	The Prefix of the log filename — the full path is `<log_dir>/<log_filename>`	catalogd
max_log_files	The number of log files that are kept for each severity level (`INFO`, `WARNING`, `ERROR`, and `FATAL`) before older log files are removed. The number should be greater than 1 to keep at least the current log file to remain open. If set to `0`, all log files are retained and log rotation is disabled	10
minidump_path	The directory for storing the Catalog Service daemon Breakpad dumps	/var/log/impala-minidumps
max_minidumps	The maximum number of Breakpad dump files stored by Catalog Service. A negative value or `0` is interpreted as an unlimited number	9
hms_event_polling_interval_s	When this parameter is set to a positive integer, Catalog Service fetches new notifications from Hive Metastore at the specified interval in seconds. If `hms_event_polling_interval_s` is set to `0`, the automatic metadata invalidation and updates are disabled. See Metadata management	2
load_auth_to_local_rules	If checked (True) and Kerberos is enabled for Impala, Impala uses the `auth_to_local` option from `hadoop.security.auth_to_local` rules of the HDFS configuration	True
load_catalog_in_background	If it is set to `True`, the metadata is loaded in the background, even if that metadata is not required for any query. If `False`, the metadata is loaded when it is referenced for the first time	False
catalog_topic_mode	The granularity of on-demand metadata fetches between the Impala Daemon coordinator and Impala Catalog Service. See Metadata management	minimal
statestore_subscriber_timeout_seconds	The timeout in seconds for Impala Daemon and Catalog Server connections to Statestore	30
state_store_subscriber_port	The port where StateStoreSubscriberService is running. StateStoreSubscriberService listens on this port for updates from the Statestore daemon	23020

enable_statestored_ha	Indicates whether two Impala Statestore components are present in the cluster, which allows them to work in the high availability mode	false
state_store_2_host	The host where the second Impala Statestore component is running (high availability mode)	—
state_store_2_port	The port on which the second Impala Statestore component is running (high availability mode)	24000
enable_catalogd_ha	Indicates whether two Impala Catalog Service components are present in the cluster, which allows them to work in the high availability mode	false
kerberos_reinit_interval	The number of minutes between reestablishing the ticket with the Kerberos server	60
principal	The service Kerberos principal	—
keytab_file	The service Kerberos keytab file	—
ssl_server_certificate	The path to the TLS/SSL file with the server certificate key used for TLS/SSL. It is used when Impala operates as a TLS/SSL server. The certificate file must be in the PEM format	—
ssl_private_key	The path to the TLS/SSL file with the private key used for TLS/SSL. It is used when Impala operates as a TLS/SSL server. The file must be in the PEM format	—
ssl_client_ca_certificate	The path to the certificate, in the PEM format, used to confirm the authenticity of SSL/TLS servers that the Impala daemons can connect to. Since the Impala daemons connect to each other, it should also include the CA certificate used to sign all the SSL/TLS certificates. SSL/TLS between Impala daemons cannot be enabled without this parameter	—
webserver_certificate_file	The path to the TLS/SSL file with the server certificate key used for TLS/SSL. It is used when the Catalog Service web server operates as a TLS/SSL server. The certificate file must be in the PEM format	—
webserver_private_key_file	The path to the TLS/SSL file with the private key used for TLS/SSL. It is used when the Catalog Service web server operates as a TLS/SSL server. The certificate file must be in the PEM format	—
ssl_minimum_version	The minimum version of TLS	TLSv1.2

Others

Parameter	Description	Default value
Custom catalogstore.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *catalogstore.conf*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

Enable custom ulimits — Default value:

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process’s data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

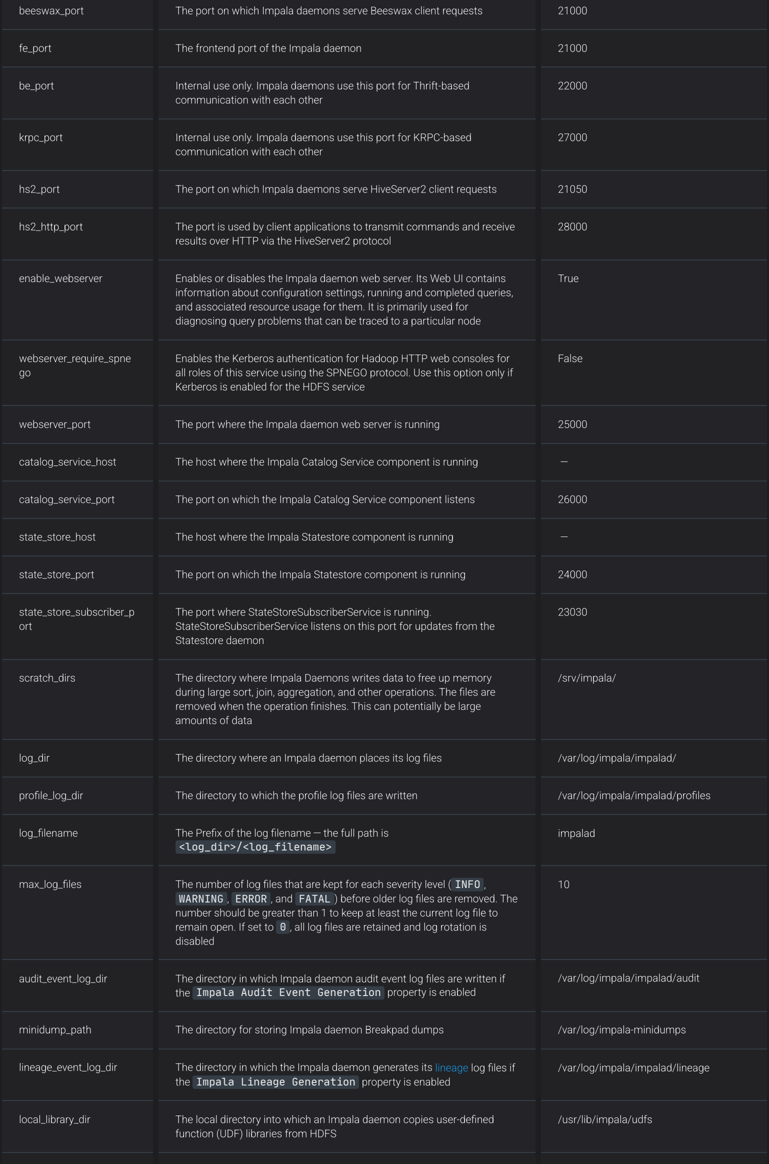
The Impala Client component

Parameter	Description	Default value
impala-shell-env.sh	The contents of the *impala-shell-env.sh* file that sets up necessary environment variables	impala-shell-env.sh
impalarc	The contents of the *impalarc* file with the LDAP settings for impala-shell	impalarc

The Impala Daemon component

impalastore.conf

Parameter	Description	Default value
hostname	The hostname to use for the Impala daemon. If Kerberos is enabled, it is also used as a part of the Kerberos principal. If this option is not set, the system default is used	—



beeswax_port	The port on which Impala daemons serve Beeswax client requests	21000
fe_port	The frontend port of the Impala daemon	21000
be_port	Internal use only. Impala daemons use this port for Thrift-based communication with each other	22000
krpc_port	Internal use only. Impala daemons use this port for KRPC-based communication with each other	27000
hs2_port	The port on which Impala daemons serve HiveServer2 client requests	21050
hs2_http_port	The port is used by client applications to transmit commands and receive results over HTTP via the HiveServer2 protocol	28000
enable_webserver	Enables or disables the Impala daemon web server. Its Web UI contains information about configuration settings, running and completed queries, and associated resource usage for them. It is primarily used for diagnosing query problems that can be traced to a particular node	True
webserver_require_spnego	Enables the Kerberos authentication for Hadoop HTTP web consoles for all roles of this service using the SPNEGO protocol. Use this option only if Kerberos is enabled for the HDFS service	False
webserver_port	The port where the Impala daemon web server is running	25000
catalog_service_host	The host where the Impala Catalog Service component is running	—
catalog_service_port	The port on which the Impala Catalog Service component listens	26000
state_store_host	The host where the Impala Statestore component is running	—
state_store_port	The port on which the Impala Statestore component is running	24000
state_store_subscriber_port	The port where StateStoreSubscriberService is running. StateStoreSubscriberService listens on this port for updates from the Statestore daemon	23030
scratch_dirs	The directory where Impala Daemons writes data to free up memory during large sort, join, aggregation, and other operations. The files are removed when the operation finishes. This can potentially be large amounts of data	/srv/impala/
log_dir	The directory where an Impala daemon places its log files	/var/log/impala/impalad/
profile_log_dir	The directory to which the profile log files are written	/var/log/impala/impalad/profiles
log_filename	The Prefix of the log filename — the full path is `<log_dir>/<log_filename>`	impalad
max_log_files	The number of log files that are kept for each severity level (`INFO`, `WARNING`, `ERROR`, and `FATAL`) before older log files are removed. The number should be greater than 1 to keep at least the current log file to remain open. If set to `0`, all log files are retained and log rotation is disabled	10
audit_event_log_dir	The directory in which Impala daemon audit event log files are written if the `Impala Audit Event Generation` property is enabled	/var/log/impala/impalad/audit
minidump_path	The directory for storing Impala daemon Breakpad dumps	/var/log/impala-minidumps
lineage_event_log_dir	The directory in which the Impala daemon generates its lineage log files if the `Impala Lineage Generation` property is enabled	/var/log/impala/impalad/lineage
local_library_dir	The local directory into which an Impala daemon copies user-defined function (UDF) libraries from HDFS	/usr/lib/impala/udfs

max_lineage_log_file_size	The maximum size (in entries) of the Impala daemon lineage log file. When the size is exceeded, a new file is created	5000
max_audit_event_log_file_size	The maximum size (in queries) of the Impala Daemon audit event log file. When the size is exceeded, a new file is created	5000
fe_service_threads	The maximum number of concurrent client connections allowed. The parameter determines how many queries can run simultaneously. When more clients try to connect to Impala, the later arriving clients have to wait until previous clients disconnect. Setting the `fe_service_threads` value too high could negatively impact query latency	64
mem_limit	The memory limit (in bytes) for an Impala daemon enforced by the daemon itself. This limit does not include memory consumed by the daemon’s embedded JVM. The Impala daemon uses up this amount of memory for query processing, cached data, network buffers, background operations, etc. If the limit is exceeded, queries will be killed until the used memory becomes under the limit	1473249280
idle_query_timeout	The time in seconds after which an idle query (no processing work is done and no updates are received from the client) is cancelled. If set to `0`, idle queries are never expired	0
idle_session_timeout	The time in seconds after which Impala closes an idle session and cancels all running queries. If set to `0`, idle sessions never expire	0
max_result_cache_size	The maximum number of query results a client can request to be cached on a per-query basis to support restarting fetches. This option guards against unreasonably large result caches. Requests exceeding this maximum are rejected	100000
max_cached_file_handles	The maximum number of cached HDFS file handles. Caching HDFS file handles reduces the number of new file handles opened and thus reduces the load on a HDFS NameNode. Each cached file handle consumes a small amount of memory. If set to `0`, the file handle caching is disabled	20000
unused_file_handle_timeout_sec	The maximum time in seconds during which an unused HDFS file handle remains in the HDFS file handle cache. When the underlying file for a cached file handle is deleted, the disk space may not be freed until the cached file handle is removed from the cache. This timeout allows the disk space occupied by deleted files to be freed in a predictable period of time. If set to `0`, unused cached HDFS file handles are not removed	21600
statestore_subscriber_timeout_seconds	The timeout in seconds for Impala Daemon and Catalog Server connections to Statestore	30
default_query_options	A list of key/value pairs representing additional query options to pass to the Impala Daemon command line, separated by commas	default_file_format=parquet,default_transactional_type=none
load_auth_to_local_rules	If checked (True) and Kerberos is enabled for Impala, Impala uses the `auth_to_local` option from `hadoop.security.auth_to_local` rules of the HDFS configuration	True
catalog_topic_mode	The granularity of on-demand metadata fetches between the Impala Daemon coordinator and Impala Catalog Service. See Metadata management	minimal
use_local_catalog	Allows coordinators to cache metadata from Impala Catalog Service. If this is set to `True`, coordinators pull metadata as needed from catalogd and cache it locally. The cached metadata is automatically removed under memory pressure or after an expiration time. See Metadata management	True
abort_on_failed_audit_event	Specifies whether shutdown Impala if there is a problem with recording an audit event	False
max_minidumps	The maximum number of Breakpad dump files stored by the Impala daemon. A negative value or `0` is interpreted as an unlimited number	9

authorized_proxy_user_config	Specifies the set of authorized proxy users (the users who can impersonate other users during authorization), and users who they are allowed to impersonate. The example of syntax for the option is: `authenticated_user1=delegated_user1,delegated_user2;authenticated_user2=*.` See Configuring Impala delegation for clients. The list can contain short usernames or `*` to indicate all users	knox=*;zeppelin=*
queue_wait_timeout_ms	The maximum amount of time (in milliseconds) that a request waits to be admitted before timing out. Must be a positive integer	60000
disk_spill_encryption	Specifies whether to encrypt and verify the integrity of all data spilled to the disk as part of a query	False
abort_on_config_error	Specifies whether to abort Impala startup if there are incorrect configs or Impala is running on unsupported hardware	True
llama_site_path	Path to the *llama-site.xml* configuration file	/etc/impala/conf/llama-site.xml
fair_scheduler_allocation_path	Path to the *fair-scheduler.xml* configuration file	/etc/impala/conf/fair-scheduler.xml
enable_statestored_ha	Indicates whether two Impala Statestore components are present in the cluster, which allows them to work in the high availability mode	false
state_store_2_host	The host where the second Impala Statestore component is running (high availability mode)	—
state_store_2_port	The port on which the second Impala Statestore component is running (high availability mode)	24000
kerberos_reinit_interval	The number of minutes between reestablishing the ticket with the Kerberos server	60
principal	The service Kerberos principal	—
keytab_file	The service Kerberos keytab file	—
ssl_server_certificate	The path to the TLS/SSL file with the server certificate key used for TLS/SSL. It is used when Impala operates as a TLS/SSL server. The certificate file must be in the PEM format	—
ssl_private_key	The path to the TLS/SSL file with the private key used for TLS/SSL. It is used when Impala operates as a TLS/SSL server. The file must be in the PEM format	—
ssl_client_ca_certificate	The path to the certificate, in the PEM format, used to confirm the authenticity of SSL/TLS servers that the Impala daemons can connect to. Since the Impala daemons connect to each other, it should also include the CA certificate used to sign all the SSL/TLS certificates. SSL/TLS between Impala daemons cannot be enabled without this parameter	—
webserver_certificate_file	The path to the TLS/SSL file with the server certificate key used for TLS/SSL. It is used when the Impala daemon web server operates as a TLS/SSL server. The certificate file must be in the PEM format	—
webserver_private_key_file	The path to the TLS/SSL file with the private key used for TLS/SSL. It is used when the Impala daemon web server operates as a TLS/SSL server. The certificate file must be in the PEM format	—
ssl_minimum_version	The minimum version of TLS	TLSv1.2

Others

Parameter	Description	Default value
Custom impalastore.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the	—

	configuration file *impalastore.conf*	

log4j.properties	Apache Log4j utility settings

```
log.threshold=INFO
main.logger=FA
impala.root.logger=DEBUG,FA
log4j.rootLogger=DEBUG,FA
log.dir=/var/log/impala/impalad
max.log.file.size=200MB
log4j.appender.FA=org.apache.log4j.FileAppender
log4j.appender.FA.File=/var/log/impalad/impalad.INFO
log4j.appender.FA.layout=org.apache.log4j.PatternLayout
log4j.appender.FA.layout.ConversionPattern=%p%d{MMdd HH:mm:ss.SSS'000'} %t %c] %m%n
log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.target=System.err
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{2}: %m%n
```

llama-site.xml	Resource pools configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
</configuration>
```

fair_scheduler.xml	Resource pools configuration

```
<allocations>

</allocations>
```

Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process’s data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)

DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Impala Statestore component

statestore.conf

Parameter	Description	Default value
hostname	The hostname to use for the Statestore daemon. If Kerberos is enabled, it is also used as a part of the Kerberos principal. If this option is not set, the system default is used	—
state_store_host	The host where the Impala Statestore component is running	—
state_store_port	The port on which the Impala Statestore component is running	24000
catalog_service_host	The host where the Impala Catalog Service component is running	—
catalog_service_port	The port on which the Impala Catalog Service component listens	26000
enable_webserver	Enables or disables the Statestore daemon web server. Its Web UI contains information about memory usage, configuration settings, and ongoing health checks performed by Statestore	True
webserver_require_spnego	Enables the Kerberos authentication for Hadoop HTTP web consoles for all roles of this service using the SPNEGO protocol. Use this option only if Kerberos is enabled for the HDFS service	False
webserver_port	The port on which the Statestore web server is running	25010
log_dir	The directory where the Statestore daemon places its log files	/var/log/impala/statestored/
log_filename	The Prefix of the log filename — the full path is `<log_dir>/<log_filename>`	statestored
max_log_files	The number of log files that are kept for each severity level (`INFO` , `WARNING` , `ERROR` , and `FATAL`) before older log files are removed. The number should be greater than 1 to keep at least the current log file to remain open. If set to `0` , all log files are retained and log rotation is disabled	10

minidump_path	The directory for storing Statestore daemon Breakpad dumps	/var/log/impala-minidumps
max_minidumps	The maximum number of Breakpad dump files stored by Statestore daemon. A negative value or `0` is interpreted as an unlimited number	9
state_store_num_server_worker_threads	The number of worker threads for the thread manager of the Statestore Thrift server	4
state_store_pending_task_count_max	The maximum number of tasks allowed to be pending by the thread manager of the Statestore Thrift server. The `0` value allows an infinite number of pending tasks	0
enable_statestored_ha	Indicates whether two Impala Statestore components are present in the cluster, which allows them to work in the high availability mode	false
state_store_ha_port	RPC port of the peer Statestore instance in the high availability mode	24020
kerberos_reinit_interval	The number of minutes between reestablishing the ticket with the Kerberos server	60
principal	The service Kerberos principal	—
keytab_file	The service Kerberos keytab file	—
ssl_server_certificate	The path to the TLS/SSL file with the server certificate key used for TLS/SSL. It is used when Impala operates as a TLS/SSL server. The certificate file must be in the PEM format	—
ssl_private_key	The path to the TLS/SSL file with the private key used for TLS/SSL. It is used when Impala operates as a TLS/SSL server. The file must be in the PEM format	—
ssl_client_ca_certificate	The path to the certificate, in the PEM format, used to confirm the authenticity of SSL/TLS servers that the Impala daemons can connect to. Since the Impala daemons connect to each other, it should also include the CA certificate used to sign all the SSL/TLS certificates. SSL/TLS between Impala daemons cannot be enabled without this parameter	—
webserver_certificate_file	The path to the TLS/SSL file with the server certificate key used for TLS/SSL. It is used when the Statestore web server operates as a TLS/SSL server. The certificate file must be in the PEM format	—
webserver_private_key_file	The path to the TLS/SSL file with the private key used for TLS/SSL. It is used when the Statestore web server operates as a TLS/SSL server. The certificate file must be in the PEM format	—
ssl_minimum_version	The minimum version of TLS	TLSv1.2

Others

Parameter	Description	Default value
Custom statestore.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *statestore.conf*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

Enable custom ulimits — Default value:

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
```

```
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

Kyuubi

The Kyuubi Server component

kyuubi-defaults.conf

Parameter	Description	Default value
kyuubi.frontend.rest.bind.port	Port on which the REST frontend service runs	10099
kyuubi.frontend.thrift.binary.bind.port	Port on which the Thrift frontend service runs via a binary protocol	10099
kyuubi.frontend.thrift.http.bind.port	Port on which the Thrift frontend service runs via HTTP	10010

kyuubi.frontend.thrift.http.path	The `path` component of the URL endpoint for the HTTP version of Thrift	cliservice
kyuubi.engine.share.level	An engine share level. Possible values: `CONNECTION` (one engine per connection), `USER` (one engine per user), `GROUP` (one engine per group), `SERVER` (one engine per server)	USER
kyuubi.engine.type	An engine type supported by Kyuubi. Possible values: `SPARK_SQL`, `FLINK_SQL`, `TRINO`, `HIVE_SQL`, `JDBC`	SPARK_SQL
kyuubi.operation.language	Programming language used to interpret inputs. Possible values: `SQL`, `SCALA`, `PYTHON`	SQL
kyuubi.frontend.protocols	A comma-separated list for supported frontend protocols. Possible values: `THRIFT_BINARY`, `THRIFT_HTTP`, `REST`	THRIFT_BINARY
kyuubi.frontend.thrift.binary.ssl.disallowed.protocols	Forbidden SSL versions for Thrift binary frontend	SSLv2,SSLv3,TLSv1.1
kyuubi.frontend.thrift.http.ssl.protocol.blacklist	Forbidden SSL versions for Thrift HTTP frontend	SSLv2,SSLv3,TLSv1.1
kyuubi.ha.addresses	External Kyuubi instance addresses	<hostname_1>:2181, ..., <hostname_N>:2181
kyuubi.ha.namespace	The root directory for the service to deploy its instance URI	kyuubi
kyuubi.metadata.store.jdbc.database.type	A database type for the server metadata store. Possible values: `SQLITE`, `MYSQL`, `POSTGRESQL`	POSTGRESQL
kyuubi.metadata.store.jdbc.url	A JDBC URL for the server metadata store	jdbc:postgresql://{{ groups['adpg.adpg'][0] | d(omit) }}:5432/kyuubi
kyuubi.metadata.store.jdbc.driver	A JDBC driver classname for the server metadata store	org.postgresql.Driver
kyuubi.metadata.store.jdbc.user	A username for the server metadata store	kyuubi
kyuubi.metadata.store.jdbc.password	A password for the server metadata store	—
kyuubi.metrics.enabled	Enables metrics collection from Kyuubi Server	false
kyuubi.metrics.reporters	Monitoring service for collecting metrics	PROMETHEUS
kyuubi.metrics.prometheus.port	Prometheus service port	10019
kyuubi.metrics.prometheus.path	Prometheus service metrics endpoint	/metrics
kyuubi.frontend.thrift.binary.ssl.enabled	Indicates whether to use the SSL encryption in the Thrift binary mode	false
kyuubi.frontend.thrift.http.use.SSL	Indicates whether to use the SSL encryption in the Thrift HTTP mode	false
kyuubi.frontend.ssl.keystore.type	Type of the SSL certificate keystore	—
kyuubi.frontend.ssl.keystore.path	Path to the SSL certificate keystore	—
kyuubi.frontend.ssl.keystore.password	Password for the SSL certificate keystore	—
kyuubi.frontend.thrift.http.ssl.keystore.path	Path to the SSL certificate keystore	—
kyuubi.frontend.thrift.http.ssl.keystore.password	Password for the SSL certificate keystore	—

kyuubi.authentication	Authentication type. Possible values: `NONE`, `LDAP`, or `LDAP,KERBEROS`	NONE
kyuubi.ha.zookeeper.acl.enabled	Indicates whether the ZooKeeper ensemble is kerberized	false
kyuubi.ha.zookeeper.auth.type	ZooKeeper authentication type. Possible values: `NONE`, `KERBEROS`	NONE
kyuubi.ha.zookeeper.auth.principal	Kerberos principal name used for ZooKeeper authentication	—
kyuubi.ha.zookeeper.auth.keytab	Path to Kyuubi Server's keytab used for ZooKeeper authentication	—
kyuubi.kinit.principal	Name of the Kerberos principal	—
kyuubi.kinit.keytab	Path to Kyuubi Server's keytab	—
kyuubi.spnego.principal	Name of the SPNego service principal. Set only if using SPNego in authentication	—
kyuubi.spnego.keytab	Path to the SPNego service keytab. Set only if using SPNego in authentication	—
kyuubi.engine.hive.java.options	Extra Java options for the Hive query engine	—

LDAP Security

Parameter	Description	Default value
kyuubi.authentication.ldap.url	A whitespace-separated list of LDAP connection URLs	—
kyuubi.authentication.ldap.domain	An LDAP domain	—
kyuubi.authentication.ldap.binddn	The distinguished name of the user to bind to for user/group searches. If not set, the name of the user trying to authenticate will be used. For example, `CN=bindUser,CN=Users,DC=subdomain,DC=domain,DC=com`	—
kyuubi.authentication.ldap.bindpw	The password for the bind user. Required only if `kyuubi.authentication.ldap.binddn` was specified	—
kyuubi.authentication.ldap.baseDN	An LDAP base DN	—
kyuubi.authentication.ldap.groupClassKey	An LDAP attribute name on the group entry to be used in the LDAP group searches. For example: `group`, `groupOfNames`, or `groupOfUniqueNames`	—
kyuubi.authentication.ldap.groupDNPattern	A colon-separated list of the patterns to use to find DNs for group entities in this directory. Use `%s` where the actual group name needs to be put. For example: `CN=%s,CN=Groups,DC=subdomain,DC=domain,DC=com`	—
kyuubi.authentication.ldap.groupFilter	A comma-separated list of the LDAP group names (short names, not full DNs). For example: `HiveAdmins,HadoopAdmins,Administrators`	—
kyuubi.authentication.ldap.groupMembershipKey	An LDAP attribute name on the group object that contains the list of distinguished names for the user, group, and contact objects that are members of the group. For example: `member`, `uniqueMember`, or `memberUid`	—
kyuubi.authentication.ldap.guidKey	An LDAP attribute name whose values are unique in this LDAP server. For example: `uid` or `CN`	—
kyuubi.authentication.ldap.userDNPattern	A colon-separated list of patterns to use to find DNs for users in this directory. Use `%s` where the actual user name needs to be put. For example: `CN=%s,CN=Users,DC=subdomain,DC=domain,DC=com`	—
kyuubi.authentication.ldap.userFilter	A comma-separated list of the LDAP usernames (short names, not full DNs). For example: `hiveuser,impalauser,hiveadmin,hadoopadmin`	—

kyuubi.authentication.ldap.userMembershipKey	An LDAP attribute name on the user object that contains groups of which the user is a direct member, except for the primary group, which is represented by the `primaryGroupId`. For example: `memberOf`	—

kyuubi-env.sh

Parameter	Description	Default value
KYUUBI_HOME	Kyuubi home directory	/usr/lib/kyuubi
KYUUBI_CONF_DIR	Directory that stores Kyuubi configurations	/etc/kyuubi/conf
KYUUBI_LOG_DIR	Kyuubi server log directory	/var/log/kyuubi
KYUUBI_PID_DIR	Directory that stores the Kyuubi instance *.pid*-file	/var/run/kyuubi
KYUUBI_ADDITIONAL_CLASSPATH	Additional classpath entries to be added to Kyuubi's classpath. For example, SSM libraries for statistics collection. A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ /usr/lib/ssm/lib/smart*
HADOOP_HOME	Hadoop home directory	/usr/lib/hadoop
HADOOP_LIB_DIR	Directory that stores Hadoop libraries	${HADOOP_HOME}/lib
KYUUBI_JAVA_OPTS	Java parameters for Kyuubi	-Djava.library.path=${HADOOP_LIB_DIR}/native/ -Djava.io.tmpdir={{ cluster.config.java_tmpdir | d('/tmp') }}
HADOOP_CLASSPATH	A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ $HADOOP_CLASSPATH ▪ /usr/lib/ssm/lib/smart*
HADOOP_CONF_DIR	Directory that stores Hadoop configurations	/etc/hadoop/conf
SPARK_HOME	Spark home directory	/usr/lib/spark3
SPARK_CONF_DIR	Directory that stores Spark configurations	/etc/spark3/conf
FLINK_HOME	Flink home directory	/usr/lib/flink
FLINK_CONF_DIR	Directory that stores Flink configurations	/etc/flink/conf
FLINK_HADOOP_CLASSPATH	Additional Hadoop JAR files required to use the Kyuubi Flink engine. A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ $(hadoop classpath) ▪ /usr/lib/ssm/lib/smart*
HIVE_HOME	Hive home directory	/usr/lib/hive
HIVE_CONF_DIR	Directory that stores Hive configurations	/etc/hive/conf
HIVE_HADOOP_CLASSPATH	Additional Hadoop JAR files required to use the Kyuubi Hive engine A list of files/directories to be added to the classpath. To add more items to the classpath, click +	▪ $(hadoop classpath) ▪ /etc/tez/conf/ ▪ /usr/lib/tez/* ▪ /usr/lib/tez/lib/* ▪ /usr/lib/ssm/lib/smart*

Others

Parameter	Description	Default value

Custom properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for the Kyuubi service	—
Custom kyuubi-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *kyuubi-env.sh*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Custom log4j2-repl.xml	The contents of the *log4j2-repl.xml* configuration file	log4j2-repl.xml
log4j2.xml	The contents of the *log4j2.xml* configuration file	log4j2.xml

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)

DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

MySQL

root user

Parameter	Description	Default value
Password	A password for the `root` user	—

Ozone

Parameter	Description	Default value
ad-runtime-utils	YAML configuration file that contains the Java installation settings: version, installation path, and the environment variable	ad-runtime-utils.yml
ozone-env.sh	The contents of the *ozone-env.sh* file that contains Ozone-specific environment variables	ozone-env.sh
Topology script	The script content that should be invoked to resolve DNS names to NetworkTopology names. Example: the script takes `host.foo.bar` as an argument and returns `/rack1` as the output	—
Topology data	Optional auxiliary text file to map hostnames to rack numbers for topology script. Will be placed at */etc/hadoop/conf/topology.data*	—
Ranger plugin enabled	Defines whether the Ranger plugin for Ozone is enabled	false

ozone-site.xml

Parameter	Description	Default value
hdds.prometheus.endpoint.enabled	Enables the Prometheus endpoint for HDDS	false
ozone.administrators	A comma-separated list of user principals who have administrator privileges in Ozone	om
ozone.administrators.groups	A comma-separated list of groups whose members have administrator privileges in Ozone	hadoop
ozone.replication	The default replication factor for data in Ozone. Higher values provide higher redundancy	3
ozone.service.id	Unique identifier for the Ozone service which is used for multi-cluster configurations	—
ozone.http.basedir	The base directory for the HTTP Jetty server to extract contents	/srv/ozone/meta/webserver
ozone.network.topology.aware.read	Defines whether the data is read from the closest pipeline	false
ozone.security.enabled	Defines whether secure connections are enabled for Ozone	false
hadoop.security.authentication	Authentication mechanism for Hadoop and Ozone	simple

ozone.security.http.kerberos.enabled	Defines whether Kerberos-based HTTP authentication is used for Ozone services	false
ozone.http.filter.initializers	HTTP filter initializer for Kerberos-based authentication	—
ozone.http.policy	Specifies the HTTP filter initializer for Kerberos-based authentication	HTTP_ONLY
hdds.grpc.tls.enabled	Defines whether TLS is enabled for HDDS GRPC server communication	false
ozone.https.client.need-auth	Specifies whether HTTPS clients need to authenticate using certificates	false

ssl-server.xml

Parameter	Description	Default value
ssl.server.keystore.location	Path to the keystore file	—
ssl.server.keystore.password	Password for the keystore file	—
ssl.server.truststore.location	Path to the truststore	—
ssl.server.truststore.password	Password for the truststore	—

ranger-ozone-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/ozone_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Whether to use in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal's secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—

xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service for login	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal's key to be stored in the subject's private credentials	—
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal's key from the keytab	—

ranger-ozone-security.xml

Parameter	Description	Default value
ranger.plugin.ozone.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.ozone.service.name	Name of the Ranger service containing policies for this Ozone instance	—
ranger.plugin.ozone.policy.cache.dir	Directory where Ranger policies are cached after a successful retrieval from the source	/srv/ranger/ozone/policycache
ranger.plugin.ozone.policy.pollIntervalMs	How often to poll for changes in policies in milliseconds	30000
ranger.plugin.ozone.policy.rest.client.connection.timeoutMs	Ozone plugin connection timeout in milliseconds	120000
ranger.plugin.ozone.policy.rest.client.read.timeoutMs	Ozone plugin read timeout in milliseconds	30000
ranger.plugin.ozone.policy.rest.ssl.config.file	The path to the RangerRestClient SSL config file for the Ozone plugin	/etc/ozone/conf/ranger-hbase-policymgr-ssl.xml

ranger-ozone-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/ozone/conf/ranger-ozone.jceks
xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/ozone/conf/ranger-ozone.jceks
xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

Credential encryption

Parameter	Description	Default value
Encryption enable	Defines whether the credentials are encrypted	false
Credential provider path	Path to the credential provider for creating the *.jceks* files containing secret keys	jceks://file/etc/ozone/conf/ozone.jceks
Ranger plugin credential provider path	Path to the Ranger plugin credential provider	jceks://file/etc/ozone/conf/ranger-ozone.jceks
Custom jceks	Defines whether custom *.jceks* files located at the credential provider path (`true`) or auto-generated ones (`false`) are used	false
Password file name	Name of the password file in the classpath of the service if the password file is selected in the credstore options	ozone_credstore_pass

Other

Parameter	Description	Default value
Custom ozone-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ozone-site.xml*	—
Custom ssl-server.xml.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ssl-server.xml.xml*	—
Custom ranger-ozone-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-ozone-audit.xml*	—
Custom ranger-ozone-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-ozone-security.xml*	—
Custom ranger-ozone-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-ozone-policymgr-ssl.xml*	—

The Ozone Datanode component

ozone-site.xml

Parameter	Description	Default value
hdds.datanode.http-address	HTTP address of the Datanode web interface	0.0.0.0:9882
hdds.datanode.https-address	HTTPS address of the Datanode web interface	0.0.0.0:9883
ozone.scm.datanode.id.dir	Path to the directory where the Datanode stores its unique identifier file	/srv/ozone/meta/node
hdds.datanode.dir	Determines, where on the local filesystem the Datanode should store its data. If multiple directories are specified, then data will be stored in all named directories, typically on different devices. The directories should be tagged with corresponding storage types (`SSD`/`DISK`/`ARCHIVE`/`RAM_DISK`). The default storage type will be `DISK` if the directory does not have a storage type tagged explicitly. Directories, that do not exist, will be created if the local filesystem permission allows	/srv/ozone/data:DISK

hdds.datanode.container.db.dir	Determines, where on the local filesystem the Datanode should store container database files. If multiple directories are specified, then data will be stored in all named directories, typically on different devices. The directories should be tagged with corresponding storage types (`SSD` / `DISK` / `ARCHIVE` / `RAM_DISK`). The default storage type will be `DISK` if the directory does not have a storage type tagged explicitly. Directories, that do not exist, will be created if the local filesystem permission allows	/srv/ozone/data/db:DISK
hdds.container.ratis.datanode.storage.dir	Path to the directory for storing Ratis logs for the Datanode	/srv/ozone/data/logs
hdds.container.ratis.enabled	Defines whether the Ratis for the Datanode is enabled to provide high availability and replication	true
hdds.container.ratis.datastream.enabled	Defines whether the DataStream is supported in Ratis for efficient data streaming between Datanodes	true
hdds.container.ratis.datastream.port	Port used for Ratis DataStream on the Datanode	9855
ozone.fs.datastream.enabled	Defines whether the data streaming over the filesystem for Ozone is enabled	true
ozone.container.cache.size	Size of the container cache for metadata and other frequently accessed data, in megabytes	8192
ozone.container.cache.lock.stripes	Number of lock stripes for container cache. It is used to manage concurrent access	8192
hdds.datanode.du.factory.classname	Defines the factory class for calculating disk usage on the Datanode	org.apache.hadoop.hdds.fs.DedicatedDiskSpaceUsageFactory
hdds.container.report.interval	Interval for the Datanode to send the container reports to Storage Container Manager	10m
hdds.container.ratis.leader.pending.bytes.limit	Limit for the pending bytes for the Ratis leader in a Datanode	2GB
ozone.recon.address	Address for connecting to the Recon server for the Datanode metrics and reports	{{ groups['ozone.ozone_recon'][0] | d('0.0.0.0') }}:9891
dfs.datanode.kerberos.principal	Datanode service Kerberos principal	—
dfs.datanode.kerberos.keytab.file	Path to the keytab file used by the Datanode daemon as its service principal to log in with	—
hdds.datanode.http.auth.type	Authentication mechanism for the Datanode HTTP server	—
hdds.datanode.http.auth.kerberos.principal	Datanode HTTP server service principal	—
hdds.datanode.http.auth.kerberos.keytab	Path to the keytab file used by the Datanode HTTP server as its service principal to log in with	—

Others

Parameter	Description	Default value
Custom ozone-site.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ozone-site.xml*	—

Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process’s data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Ozone HttpFS component

httpfs-env.sh

Parameter	Description	Default value

HTTPFS_CONFIG	Path to the directory with the HttpFS configuration files	${OZONE_CONF_DIR}
HTTPFS_LOG	Path to the directory with the HttpFS logs	${OZONE_LOG_DIR}
HTTPFS_TEMP	Path to the HttpFS temporary directory	${OZONE_LOG_DIR}
HADOOP_PID_DIR	Path to the directory where HttpFS stores process ID files	${OZONE_LOG_DIR}
HDFS_HTTPFS_OPTS	JVM options for the HttpFS service	-Xms700m -Xmx8G

httpfs-site.xml

Parameter	Description	Default value
httpfs.http.port	Port of the HttpFS service	14001
hadoop.http.temp.dir	Path to the temporary directory for the HttpFS service	${hadoop.tmp.dir}/httpfs
httpfs.hadoop.config.dir	Path to the Hadoop configuration directory for HttpFS	/etc/ozone/conf
httpfs.http.administrators	List of administrators for HttpFS service	*
httpfs.proxyuser.om.groups	List of user groups for Ozone Manager authentication in HttpFS	*
httpfs.proxyuser.om.hosts	List of hosts for Ozone Manager authentication in HttpFS	*
httpfs.proxyuser.hue.groups	List of user groups for HUE authentication in HttpFS	*
httpfs.proxyuser.hue.hosts	List of hosts for HUE authentication in HttpFS	*
httpfs.hadoop.authentication.kerberos.keytab	Kerberos keytab for Ozone Manager authentication in HttpFS	—
httpfs.hadoop.authentication.kerberos.principal	Kerberos principal for Ozone Manager authentication in HttpFS	—
httpfs.hadoop.authentication.type	Authentication mechanism for Ozone Manager	simple
hadoop.http.authentication.kerberos.keytab	Path to Kerberos keytab for HttpFS authentication	—
hadoop.http.authentication.kerberos.principal	Kerberos principal for HttpFS authentication	—
hadoop.http.authentication.type	Authentication mechanism for HttpFS	simple
httpfs.ssl.enabled	Defines whether SSL is enabled for HttpFS	false

Others

Parameter	Description	Default value
Custom httpfs-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *httpfs-env.sh*	—

Custom httpfs-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *httpfs-site.xml*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

Enable custom ulimits:

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)

DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Ozone Manager component

ozone-site.xml

Parameter	Description	Default value
ozone.om.address	Address of the Ozone Manager	0.0.0.0:9862
ozone.om.enable.filesystem.paths	Defines whether the filesystem path-style operations are enabled	true
ozone.om.http-address	HTTP address of the Ozone Manager web interface	0.0.0.0:9874
ozone.om.https-address	HTTPS address of the Ozone Manager web interface	0.0.0.0:9875
ozone.om.ratis.port	Port for Ratis communication used by Ozone Manager	9858
ozone.om.db.dirs	Path to the directory where the Ozone Manager stores its database files	/srv/ozone/meta/db
ozone.om.db.dirs.permissions	Permission mode for the directory specified by the `ozone.om.db.dirs` parameter	750
ozone.metadata.dirs	Path to the directory for general metadata storage in Ozone	/srv/ozone/meta
ozone.metadata.dirs.permissions	Permission mode for the directory specified by the `ozone.metadata.dirs` parameter	750
ozone.om.snapshot.diff.db.dir	Path to the directory for storing the snapshot diff database for Ozone Manager	/srv/ozone/meta/snapshot
ozone.om.ratis.enable	Defines whether the Ratis-based replication for Ozone Manager is enabled to ensure high availability	true
ozone.om.service.ids	Identifiers for the Ozone Manager services used in multi-instance setups for high availability	—
ozone.om.kerberos.principal	Ozone Manager service principal	—
ozone.om.kerberos.keytab.file	Path to the keytab file used by the Ozone Manager daemon as its service principal to log in with	—
ozone.om.http.auth.type	Authentication mechanism for the Ozone Manager HTTP server	simple
ozone.om.http.auth.kerberos.principal	Principal of the Ozone Manager HTTP server service if SPNEGO is enabled	—
ozone.om.http.auth.kerberos.keytab	Path to the keytab file used by the Ozone Manager HTTP server as its service principal to log in with if SPNEGO is enabled	—

Others

Parameter	Description	Default value
Custom ozone-site.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ozone-site.xml*	—

Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Ozone Recon component

ozone-site.xml

Parameter	Description	Default value

ozone.recon.db.dir	Path to the directory where Recon stores its database files for metrics and reports	/srv/ozone/recon/db
ozone.recon.task.pipelinesync.interval	Interval for the pipeline sync task, which synchronizes the pipeline information from Storage Container Manager	120s
ozone.recon.task.missingcontainer.interval	Interval for the missing container check, used to identify and report missing containers in the cluster	3600s
ozone.recon.http-address	HTTP address for the Recon web interface, used to view metrics and reports for the Ozone cluster	0.0.0.0:9888
ozone.recon.https-address	HTTPS address for the Recon web interface	0.0.0.0:9889
ozone.recon.kerberos.principal	Recon service principal	—
ozone.recon.kerberos.keytab.file	Path to the keytab file used by the Recon daemon as its service principal to log in with	—
ozone.recon.http.auth.type	Authentication mechanism for the Recon HTTP server	simple
ozone.recon.http.auth.kerberos.principal	Principal of the Recon HTTP server service for Kerberos authentication	—
ozone.recon.http.auth.kerberos.keytab	Path to the keytab file used by the Recon HTTP server as its service principal to log in with	—

Others

Parameter	Description	Default value
Custom ozone-site.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ozone-site.xml*	—

Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	`[Service]`
		`DefaultLimitCPU=`
		`DefaultLimitFSIZE=`
		`DefaultLimitDATA=`
		`DefaultLimitSTACK=`
		`DefaultLimitCORE=`
		`DefaultLimitRSS=`
		`DefaultLimitNOFILE=`
		`DefaultLimitAS=`
		`DefaultLimitNPROC=`
		`DefaultLimitMEMLOCK=`
		`DefaultLimitLOCKS=`
		`DefaultLimitSIGPENDING=`
		`DefaultLimitMSGQUEUE=`
		`DefaultLimitNICE=`
		`DefaultLimitRTPRIO=`
		`DefaultLimitRTTIME=`

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)

DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process’s data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Ozone S3G component

ozone-site.xml

Parameter	Description	Default value
ozone.s3g.http-address	HTTP address for the S3 Gateway, which provides the S3-compatible API for Ozone	0.0.0.0:9878
ozone.s3g.https-address	HTTPS address for the S3 Gateway	0.0.0.0:9879
ozone.s3g.volume.name	Specifies the volume name used by the S3 Gateway for storing S3-compatible buckets in Ozone	s3v
ozone.s3g.client.buffer.size	Size of the client buffer used by the S3 Gateway for data transfers	4KB
ozone.s3g.kerberos.principal	S3 Gateway service principal	—
ozone.s3g.kerberos.keytab.file	Path to the keytab file used by the S3 Gateway as its service principal to log in with	—
ozone.s3g.http.auth.type	Authentication mechanism for the S3G HTTP server	simple
ozone.s3g.http.auth.kerberos.principal	The S3 Gateway service principal if SPNEGO is enabled for the HTTP server	—
ozone.s3g.http.auth.kerberos.keytab	Path to the keytab file used by the S3 Gateway HTTP server as its service principal to log in with if SPNEGO is enabled	—

Others

Parameter	Description	Default value
Custom ozone-site.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ozone-site.xml*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	See the code block below

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process’s data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)

DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Ozone Storage Container Manager component

ozone-site.xml

Parameter	Description	Default value
ozone.scm.http-address	HTTP address for the Storage Container Manager web interface	0.0.0.0:9876
ozone.scm.https-address	HTTPS address for the Storage Container Manager web interface	0.0.0.0:9877
ozone.scm.ratis.port	Port for Ratis communication used by Storage Container Manager for high availability	9894
ozone.scm.db.dirs	Path to the directory where Storage Container Manager stores its database files	/srv/ozone/scm/db
ozone.scm.db.dirs.permissions	Permission mode for the directory specified by the `ozone.scm.db.dirs` parameter	750
ozone.scm.ha.ratis.storage.dir	Path to the directory for storing Ratis logs for high availability in Storage Container Manager	/srv/ozone/scm/ratis
ozone.scm.ha.ratis.snapshot.dir	Path to the directory for storing Ratis snapshots in Storage Container Manager	/srv/ozone/scm/ratis/snapshot
ozone.scm.service.ids	Identifiers for Storage Container Manager services used in multi-instance setups for high availability	—
ozone.scm.primordial.node.id	Specifies the ID of the first Storage Container Manager node in the cluster	{{ groups['ozone.ozone_scm'][0] }}
ozone.scm.names	Comma-separated list of fully qualified domain names (FQDNs) for the Storage Container Manager nodes in the cluster	{{ groups['ozone.ozone_scm'] | join(',') }}
ozone.scm.client.address	Comma-separated list of addresses for the Storage Container Manager clients, typically FQDNs of Storage Container Manager nodes	{{ groups['ozone.ozone_scm'] | join(',') }}
ozone.scm.ratis.enable	Defines whether the Ratis-based replication is enabled for Storage Container Manager to ensure high availability	true
ozone.scm.datanode.pipeline.limit	Maximum number of pipelines each Datanode can be part of in Storage Container Manager	10
ozone.scm.pipeline.owner.container.count	Maximum number of containers owned by each pipeline	10
ozone.scm.pipeline.creation.auto.factor.one	Defines whether Storage Container Manager should automatically create pipelines with a replication factor of one	false
ozone.scm.container.placement.impl	Specifies the container placement policy for Storage Container Manager, typically to optimize disk usage based on available capacity	org.apache.hadoop.hdds.scm.container.placement.algorithms.SCMContainerPlacementCapacity
ozone.scm.kerberos.principal	Storage Container Manager service principal	—
ozone.scm.kerberos.keytab.file	Path to the keytab file used by the Storage Container Manager daemon as its service principal to log in with	—
ozone.scm.http.auth.typ	Authentication mechanism for the Storage Container Manager HTTP	simple

e	server	
ozone.scm.http.auth.kerberos.principal	Principal of the Storage Container Manager service if SPNEGO is enabled for the HTTP server	—
ozone.scm.http.auth.kerberos.keytab	Path to the keytab file used by the Storage Container Manager HTTP server as its service principal to log in with if SPNEGO is enabled	—

Others

Parameter	Description	Default value
Custom ozone-site.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ozone-site.xml*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)

DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

Solr

solr-env.sh

Parameter	Description	Default value
SOLR_HOME	The location for index data and configs	/srv/solr/server
SOLR_AUTH_TYPE	Specifies the authentication type for Solr	—
SOLR_AUTHENTICATION_OPTS	Solr authentication options	—
SOLR_AUTHENTICATION_OPTS_CUSTOM	Custom authentication options for Solr	—
GC_TUNE	JVM parameters for Solr	-XX:-UseLargePages
SOLR_SSL_KEY_STORE	The path to the Solr keystore file	—
SOLR_SSL_KEY_STORE_TYPE	The type of the keystore	JKS
SOLR_SSL_KEY_STORE_PASSWORD	The password to the Solr keystore file	—
SOLR_SSL_TRUST_STORE	The path to the Solr truststore file	—
SOLR_SSL_TRUST_STORE_TYPE	The type of the truststore	JKS
SOLR_SSL_TRUST_STORE_PASSWORD	The password to the Solr truststore file	—
SOLR_SSL_NEED_CLIENT_AUTH	Defines if client authentication is enabled	false
SOLR_SSL_WANT_CLIENT_AUTH	Enables clients to authenticate (but not requires)	false
SOLR_SSL_CLIENT_HOSTNAME_VERIFICATION	Defines whether to enable hostname verification	false
SOLR_HOST	Specifies the host name of the Solr server	—
SOLR_PORT	The port number for Solr nodes to listen	8983
LOG4J_PROPS	A list of comma-separated paths to Log4j configuration files used by Solr	/etc/solr/conf/log4j2.xml,/etc/solr/conf/log4j2-console.xml

External zookeeper

Parameter	Description	Default value
ZK_HOST	Comma-separated locations of all servers in the ensemble and the ports on which they communicate. You can put ZooKeeper chroot at the end of your `ZK_HOST` connection string. For example, `host1.mydomain.com:2181,host2.mydomain.com:2181,host3.mydomain.com:2181/solr`	—
The external zookeeper is kerberized	If the external ZooKeeper is kerberized, the value must be set to `true`	false

Solr server heap memory settings

Parameter	Description	Default value
Solr Server Heap Memory	Sets initial (-Xms) and maximum (-Xmx) Java heap size for Solr Server	-Xms512m -Xmx512m

Solr collections ttl settings

Parameter	Description	Default value
collection_name	The name of the collection to configure auto-purging	—
ttl	The TTL (time-to-live) for documents in the collection	—
auto_delete_period	The time period to automatically delete documents from the collection	—

Credential Encryption

Parameter	Description	Default value
Credstore password	The encryption provider password	false
Credstore options	Defines a way to store the encryption provider password. The following options are available: ▪ `no password` — no password is used; ▪ `password in the environment` — the password is set in an environment variable.	no password
Credential provider path	The path to a keystore file with secrets	jceks://file/etc/solr/conf/solr.jceks
Ranger plugin credential provider path	The path to a Ranger keystore file with secrets	jceks://file/etc/solr/conf/ranger-solr.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the default auto-generated JCEKS file	false

ranger-solr-audit.xml

Parameter	Description	Default value

xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.solr.solr_url	A path to a Solr collection to store audit logs	—
xasecure.audit.solr.async.max.queue.size	The maximum size of internal queue used for storing audit logs	1
xasecure.audit.solr.async.max.flush.interval.ms	The maximum time interval between flushes to disk (in milliseconds)	100
xasecure.audit.solr.is.enabled	Enables Ranger audit for Solr	true
ranger.solr.plugin.audit.excluded.users	A list of users to exclude audit records	HTTP,rangeradmin,rangerkms

ranger-solr-security.xml

Parameter	Description	Default value
ranger.plugin.solr.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.solr.service.name	The name of the Ranger service containing policies for this instance	—
ranger.plugin.solr.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	/srv/ranger/solr/policycache
ranger.plugin.solr.policy.pollIntervalMs	Defines how often to poll for changes in policies	30000
ranger.plugin.solr.policy.rest.client.connection.timeoutMs	The Solr Plugin RangerRestClient connection timeout (in milliseconds)	120000
ranger.plugin.solr.policy.rest.client.read.timeoutMs	The Solr Plugin RangerRestClient read timeout (in milliseconds)	30000
ranger.plugin.solr.policy.rest.ssl.config.file	Path to the file containing SSL details to contact Ranger Admin	/usr/lib/solr/server/resources/ranger-solr-policymgr-ssl.xml
ranger.plugin.solr.policy.source.impl	Class to retrieve policies from the source	org.apache.ranger.admin.client.RangerAdminRESTClient

ranger-solr-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/solr/conf/ranger-solr.jceks
xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/solr/conf/ranger-solr.jceks

xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

Other

Parameter	Description	Default value
solr.xml	The content of *solr.xml*	solr.xml
Custom solr-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *solr-env.sh*	—
Ranger plugin enabled	Shows status of the Ranger plugin	false
Custom ranger-solr-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *ranger-solr-audit.xml* configuration file	—
Custom ranger-solr-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *ranger-solr-security.xml* configuration file	—
Custom ranger-solr-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the *ranger-solr-policymgr-ssl.xml* configuration file	—

Spark

Common

Parameter	Description	Default value
Dynamic allocation (spark.dynamicAllocation.enabled)	Defines whether to use dynamic resource allocation that scales the number of executors, registered with this application, up and down, based on the workload	false

Credential Encryption

Parameter	Description	Default value
Encryption enable	Enables or disables the credential encryption feature. When enabled, Spark stores configuration passwords and credentials required for interacting with other services in the encrypted form	false
Credential provider path	The path to a keystore file with secrets	jceks://hdfs/apps/spark/security/spark.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the default auto-generated JCEKS file	false

spark-defaults.conf

Parameter	Description	Default value
spark.yarn.archive	The archive containing needed Spark JARs for distribution to the YARN cache. If set, this configuration replaces `spark.yarn.jars` and the archive is used in all the application containers. The archive should contain JAR files in its root directory. The archive can also be hosted on HDFS to speed up file distribution	hdfs:///apps/spark/spark-yarn-archive.tgz
spark.yarn.historyServer.address	The address of Spark History Server	—
spark.master	The cluster manager to connect to	yarn
spark.dynamicAllocation.enabled	Defines whether to use dynamic resource allocation that scales the number of executors, registered with this application, up and down, based on the workload	false
spark.shuffle.service.enabled	Enables the external shuffle service. This service preserves the shuffle files written by executors so that executors can be safely removed, or so that shuffle fetches can continue in the event of executor failure. The external shuffle service must be set up in order to enable it	false
spark.eventLog.enabled	Defines whether to log Spark events, useful for reconstructing the Web UI after the application has finished	true
spark.eventLog.dir	The base directory where Spark events are logged, if `spark.eventLog.enabled=true`. Within this base directory, Spark creates a sub-directory for each application, and logs the events specific to the application in this directory. You may want to set this to a unified location like an HDFS directory so history files can be read by the History Server	hdfs:///var/log/spark/apps
spark.serializer	The class to use for serializing objects that will be sent over the network or need to be cached in serialized form. The default of Java serialization works with any `Serializable` Java object but is quite slow, so we recommend using `org.apache.spark.serializer.KryoSerializer` and configuring Kryo serialization when speed is necessary. Can be any subclass of `org.apache.spark.Serializer`	org.apache.spark.serializer.KryoSerializer
spark.dynamicAllocation.executorIdleTimeout	If dynamic allocation is enabled and an executor has been idle for more than this duration, the executor will be removed. For more details, see Spark documentation	120s
spark.dynamicAllocation.cachedExecutorIdleTimeout	If dynamic allocation is enabled and an executor which has cached data blocks has been idle for more than this duration, the executor will be removed. For more details, see Spark documentation	600s
spark.history.provider	The name of the class that implements the application history backend. Currently, there is only one implementation provided with Spark that looks for application logs stored in the file system	org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.cleaner.enabled	Specifies whether the History Server should periodically clean up event logs from storage	true
spark.history.store.path	A local directory where to cache application history data. If set, the History Server will store application data on disk instead of keeping it in memory. The data written to disk will be re-used in case of the History Server restart	/var/log/spark3/history
spark.driver.extraClassPath	Extra classpath entries to prepend to the classpath of the driver	▪ /usr/lib/hive/lib/hive-shims-scheduler.jar ▪ /usr/lib/hadoop-yarn/hadoop-yarn-server-resourcemanager.jar
spark.executor.extraClassPath	Extra classpath entries to prepend to the classpath of the executor	—
spark.history.ui.port	The port number of the History Server web UI	18082

spark.history.fs.logDirectory	The log directory of the History Server	hdfs:///var/log/spark/apps
spark.driver.extraLibraryPath:	The path to extra native libraries for driver	/usr/lib/hadoop/lib/native/
spark.yarn.am.extraLibraryPath:	The path to extra native libraries for Application Master	/usr/lib/hadoop/lib/native/
spark.executor.extraLibraryPath	The path to extra native libraries for Executor	/usr/lib/hadoop/lib/native/
spark.yarn.appMasterEnv.HIVE_CONF_DIR	A directory on the Application Master with Hive configs required for running Hive in the cluster mode	/etc/spark3/conf
spark.yarn.historyServer.allowTracking	Allows using Spark History Server for tracking UI even if web UI is disabled for a job	true
spark.ssl.enabled	Defines whether to use SSL for Spark	false
spark.ssl.protocol	TLS protocol to be used. The protocol must be supported by JVM	TLSv1.2
spark.ssl.ui.port	The port where the SSL service will listen on	4040
spark.ssl.historyServer.port	The port to access History Server web UI	18082
spark.ssl.keyPassword	The password to the private key in the key store	—
spark.ssl.keyStore	The path to the keystore file	—
spark.ssl.keyStoreType	The type of the keystore	JKS
spark.ssl.trustStorePassword	The password to the truststore used by Spark	—
spark.ssl.trustStore	The path to the truststore file	—
spark.ssl.trustStoreType	The type of the truststore	JKS
spark.history.kerberos.enabled	Indicates whether the History Server should use Kerberos to login. This is required if the History Server is accessing HDFS files on a secure Hyperwave cluster	false
spark.acls.enable	Enables Spark ACL	false
spark.modify.acls	Defines who has access to modify a running Spark application	spark,hdfs
spark.modify.acls.groups	A comma-separated list of user groups that have modify access to the Spark application	spark,hdfs
spark.history.ui.acls.enable	Specifies whether ACLs should be checked to authorize users viewing the applications in the History Server. If enabled, access control checks are performed regardless of what the individual applications had set for `spark.ui.acls.enable`. If disabled, no access control checks are made for any application UIs available through the History Server	false
spark.history.ui.admin.acls	A comma-separated list of users that have view access to all the Spark applications in History Server	spark,hdfs,dr.who
spark.history.ui.admin.acls.groups	A comma-separated list of groups that have view access to all the Spark applications in History Server	spark,hdfs,dr.who
spark.ui.view.acls	A comma-separated list of users that have view access to the Spark application. By default, only the user that started the Spark job has view access. Using `*` as a value means that any user can have view access to this Spark job	spark,hdfs,dr.who

spark.ui.view.acls.groups	A comma-separated list of groups that have view access to the Spark web UI to view the Spark Job details. This can be used if you have a set of administrators or developers or users who can monitor the Spark job submitted. Using `*` in the list means any user in any group can view the Spark job details on the Spark web UI. The user groups are obtained from the instance of the groups mapping provider specified by `spark.user.groups.mapping`	spark,hdfs,dr.who

Spark Heap Memory settings

Parameter	Description	Default value
Spark History Server Heap Memory	Sets initial (-Xms) and maximum (-Xmx) Java heap size for Spark History Server	1G
Livy Server Heap Memory	Sets initial (-Xms) and maximum (-Xmx) Java heap size for Livy Server	-Xms300m -Xmx4G

Custom log4j.properties

Parameter	Description	Default value
Spark spark-log4j.properties	Stores the Log4j configuration used for logging Spark's activity	spark-log4j.properties
Livy livy-log4.properties	Stores the Log4j configuration used for logging Livy's activity	livy-log4j.properties

livy.conf

Parameter	Description	Default value
livy.server.host	The host address to start the Livy server. By default, Livy will bind to all network interfaces	0.0.0.0
livy.server.port	The port to run the Livy server	8998
livy.spark.master	The Spark master to use for Livy sessions	yarn-cluster
livy.impersonation.enabled	Defines if Livy should impersonate users when creating a new session	false
livy.server.csrf-protection.enabled	Defines whether to enable the CSRF protection. If enabled, clients should add the `X-Requested-By` HTTP header for POST/DELETE/PUT/PATCH HTTP methods	true
livy.repl.enable-hive-context	Defines whether to enable HiveContext in the Livy interpreter. If set to `true`, *hive-site.xml* and the Livy server classpath will be detected on user request automatically	true
livy.server.recovery.mode	Sets the recovery mode for Livy	recovery
livy.server.recovery.state-store	Defines where Livy should store the state for recovery	filesystem
livy.server.recovery.state-store.url	For the `filesystem` state store, the path of the state store directory. Do not use a filesystem that does not support atomic rename like S3. For example: `file:///tmp/livy` or `hdfs:///`. For ZooKeeper, specify the address to the ZooKeeper servers. For example: `host1:port1,host2:port2`	/livy-recovery
livy.server.auth.type	Sets the Livy authentication type	—

livy.server.access_control.enabled	Defines whether to enable the access control for a Livy server. If set to `true`, then all the incoming requests will be checked if the requested user has permission	false
livy.server.access_control.users	Users allowed to access Livy. By default, any user is allowed to access Livy. If a user wants to limit the access, the user should list all the permitted users separated by a comma	livy,hdfs,spark
livy.superusers	A list of comma-separated users that have the permissions to change other user’s submitted session, like submitting statements, deleting session, and so on	livy,hdfs,spark
livy.keystore	A path to the keystore file. The path can be absolute or relative to the directory in which the process is started	—
livy.keystore.password	The password to access the keystore	—
livy.key-password	The password to access the key in the keystore	—
livy.server.thrift.ssl.protocol.blacklist	The list of banned TLS protocols	SSLv2,SSLv3,TLSv1,TLSv1.1

Other

Parameter	Description	Default value
Custom spark-defaults.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *spark-defaults.conf*	—
spark-env.sh	Enter the contents for the *spark-env.sh* file that is used to initialize environment variables on worker nodes	spark-env.sh
Custom livy.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *livy.conf*	—
livy-env.sh	Enter the contents for the *livy-env.sh* file that is used to prepare the environment for Livy startup	livy-env.sh
spark-history-env.sh	Enter the contents for the *spark-history-env.sh* file that is used to prepare the environment for History Server startup	spark-history-env.sh
Ranger plugin enabled	Enables or disables the Ranger plugin	false

Spark3

Common

Parameter	Description	Default value
Dynamic allocation (spark.dynamicAllocation.enabled)	Defines whether to use dynamic resource allocation that scales the number of executors, registered with this application, up and down, based on the workload	false

Credential Encryption

Parameter	Description	Default value

Encryption enable	Enables or disables the credential encryption feature. When enabled, Spark3 stores configuration passwords and credentials required for interacting with other services in the encrypted form	false
Credential provider path	The path to a keystore file with secrets	jceks://hdfs/apps/spark/security/spark.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the default auto-generated JCEKS file	false

spark3_iceberg_extensions

Parameter	Description	Default value
version	The version of the spark-iceberg extension package	1.5.2_arenadata1

spark-defaults.conf

Parameter	Description	Default value
spark.yarn.archive	The archive containing all the required Spark JARs for distribution to the YARN cache. If set, this configuration replaces `spark.yarn.jars` and the archive is used in all the application containers. The archive should contain JAR files in its root directory. The archive can also be hosted on HDFS to speed up file distribution	hdfs:///apps/spark/spark3-yarn-archive.tgz
spark.yarn.historyServer.address	Spark History server address	—
spark.master	The cluster manager to connect to	yarn
spark.dynamicAllocation.enabled	Defines whether to use dynamic resource allocation that scales the number of executors, registered with this application, up and down, based on the workload	false
spark.shuffle.service.enabled	Enables the external shuffle service. This service preserves the shuffle files written by executors so that executors can be safely removed, or so that shuffle fetches can continue in the event of executor failure. The external shuffle service must be set up in order to enable it	false
spark.eventLog.enabled	Defines whether to log Spark events, useful for reconstructing the Web UI after the application has finished	true
spark.eventLog.dir	The base directory where Spark events are logged, if `spark.eventLog.enabled=true`. Within this base directory, Spark creates a sub-directory for each application, and logs the events specific to the application in this directory. You may want to set this to a unified location like an HDFS directory so history files can be read by the History Server	hdfs:///var/log/spark/apps
spark.dynamicAllocation.executorIdleTimeout	If dynamic allocation is enabled and an executor has been idle for more than this duration, the executor will be removed. For more details, see Spark documentation	120s
spark.dynamicAllocation.cachedExecutorIdleTimeout	If dynamic allocation is enabled and an executor which has cached data blocks has been idle for more than this duration, the executor will be removed. For more details, see Spark documentation	600s
spark.history.provider	The name of the class that implements the application history backend. Currently there is only one implementation provided with Spark that looks for application logs stored in the file system	org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.cleaner.enabled	Specifies whether the History Server should periodically clean up event logs from storage	true

spark.history.store.path	A local directory where to cache application history data. If set, the History Server will store application data on disk instead of keeping it in memory. The data written to disk will be re-used in case of the History Server restart	/var/log/spark3/history
spark.serializer	The class used for serializing objects that will be sent over the network or need to be cached in the serialized form. By default, works with any Serializable Java object but it may be quite slow, so we recommend using `org.apache.spark.serializer.KryoSerializer` and configuring Kryo serialization when speed is necessary. Can be any subclass of `org.apache.spark.Serializer`	org.apache.spark.serializer.KryoSerializer
spark.driver.extraClassPath	Extra classpath entries to be added to the classpath of the driver	▪ /usr/lib/hive/lib/hive-shims-scheduler.jar ▪ /usr/lib/hadoop-yarn/hadoop-yarn-server-resourcemanager.jar ▪ /usr/lib/spark3/jars/adb-spark-connector-assembly-release-1.0.5-spark-3.5.2_arenadata1.jar ▪ /usr/lib/spark3/jars/adqm-spark-connector-assembly-release-1.0.0-spark-3.5.2_arenadata1.jar
spark.executor.extraClassPath	Extra classpath entries to add to the classpath of the executors	▪ /usr/lib/spark3/jars/adb-spark-connector-assembly-release-1.0.5-spark-3.5.2_arenadata1.jar ▪ /usr/lib/spark3/jars/adqm-spark-connector-assembly-release-1.0.0-spark-3.5.2_arenadata1.jar
spark.history.ui.port	The port number of the History Server web UI	18092
spark.ui.port	The port number of the Spark web UI	4140
spark.history.fs.logDirectory	The log directory of the History Server	hdfs:///var/log/spark/apps
spark.sql.extensions	A comma-separated list of Iceberg SQL extensions classes	org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
spark.sql.catalog.spark_catalog	The Iceberg catalog implementation class	org.apache.iceberg.spark.SparkSessionCatalog
spark.sql.hive.metastore.jars	The location of the JARs that should be used to instantiate HiveMetastoreClient	path
spark.sql.hive.metastore.jars.path	A list of comma-separated paths to JARs used to instantiate HiveMetastoreClient	file:///usr/lib/hive/lib/*.jar
spark.sql.hive.metastore.version	The Hive Metastore version	3.1.2
spark.driver.extraLibraryPath	The path to extra native libraries for driver	/usr/lib/hadoop/lib/native/
spark.yarn.am.extraLibraryPath	The path to extra native libraries for Application Master	/usr/lib/hadoop/lib/native/
spark.executor.extraLibraryPath	The path to extra native libraries for Executor	/usr/lib/hadoop/lib/native/
spark.yarn.appMasterEnv.HIVE_CONF_DIR	A directory on the Application Master with Hive configs required for running Hive in the cluster mode	/etc/spark3/conf
spark.yarn.historyServer.allowTracking	Allows using Spark History Server for tracking UI even if web UI is disabled for a job	True

spark.connect.grpc.binding.port	The port number to connect to Spark Connect via gRPC	15002
spark.artifactory.dir.path	The path to an artifact directory used by Spark Connect	tmp
spark.sql.security.confblacklist	Prevents overriding specified parameters from an application point of view or for information security reasons	spark.sql.extensions
spark.history.kerberos.enabled	Indicates whether the History Server should use Kerberos to login. This is required if the History Server is accessing HDFS files on a secure Hyperwave cluster	false
spark.acls.enable	Defines whether Spark ACLs should be enabled. If enabled, checks to see if the user has access permissions to view or modify the job. Note this requires the user to be known, so if the user comes across as `null` no checks are done. Filters can be used within the UI to authenticate and set the user	false
spark.modify.acls	Defines who has access to modify a running Spark application	spark,hdfs
spark.modify.acls.groups	A comma-separated list of user groups that have modify access to the Spark application	spark,hdfs
spark.history.ui.acls.enable	Specifies whether ACLs should be checked to authorize users viewing the applications in the History Server. If enabled, access control checks are performed regardless of what the individual applications had set for `spark.ui.acls.enable`. If disabled, no access control checks are made for any application UIs available through the History Server	false
spark.history.ui.admin.acls	A comma-separated list of users that have view access to all the Spark applications in History Server	spark,hdfs,dr.who
spark.history.ui.admin.acls.groups	A comma-separated list of groups that have view access to all the Spark applications in History Server	spark,hdfs,dr.who
spark.ui.view.acls	A comma-separated list of users that have view access to the Spark application. By default, only the user that started the Spark job has view access. Using `*` as a value means that any user can have view access to this Spark job	spark,hdfs,dr.who
spark.ui.view.acls.groups	A comma-separated list of groups that have view access to the Spark web UI to view the Spark Job details. This can be used if you have a set of administrators or developers or users who can monitor the Spark job submitted. Using `*` in the list means any user in any group can view the Spark job details on the Spark web UI. The user groups are obtained from the instance of the groups mapping provider specified by `spark.user.groups.mapping`	spark,hdfs,dr.who
spark.ssl.keyPassword	The password to the private key in the keystore	—
spark.ssl.keyStore	Path to the keystore file. The path can be absolute or relative to the directory in which the process is started	—
spark.ssl.keyStoreType	The type of keystore used	JKS
spark.ssl.trustStorePassword	The password to the private key in the truststore	—
spark.ssl.trustStoreType	The type of the truststore	JKS
spark.ssl.enabled	Defines whether to use SSL for Spark	—
spark.ssl.protocol	Defines the TLS protocol to use. The protocol must be supported by JVM	TLSv1.2
spark.ssl.ui.port	The port number used by Spark web UI in case of active SSL	4141
spark.ssl.historyServer.port	The port number used by Spark History Server web UI in case of active SSL	18092

Custom log4j.properties

Parameter	Description	Default value
Spark3 spark-log4j2.properties	Stores the Log4j configuration used for logging Spark3's activity	spark-log4j2.properties
Livy livy-log4j.properties	Stores the Log4j configuration used for logging Livy's activity	livy-log4j.properties

livy.conf

Parameter	Description	Default value
livy.server.host	The host address to start the Livy server. By default, Livy will bind to all network interfaces	0.0.0.0
livy.server.port	The port to run the Livy server	8999
livy.spark.master	The Spark master to use for Livy sessions	yarn
livy.impersonation.enabled	Defines if Livy should impersonate users when creating a new session	true
livy.server.csrf-protection.enabled	Defines whether to enable the CSRF protection. If enabled, clients should add the `X-Requested-By` HTTP header for POST/DELETE/PUT/PATCH HTTP methods	true
livy.repl.enable-hive-context	Defines whether to enable HiveContext in the Livy interpreter. If set to `true`, *hive-site.xml* and the Livy server classpath will be detected on user request automatically	true
livy.server.recovery.mode	Sets the recovery mode for Livy	recovery
livy.server.recovery.state-store	Defines where Livy should store the state for recovery	filesystem
livy.server.recovery.state-store.url	For the `filesystem` state store, the path of the state store directory. Do not use a filesystem that does not support atomic rename like S3. For example: `file:///tmp/livy` or `hdfs:///`. For ZooKeeper, specify the address to the ZooKeeper servers. For example: `host1:port1,host2:port2`	/livy-recovery
livy.server.auth.type	Sets the Livy authentication type	—
livy.server.access_control.enabled	Defines whether to enable the access control for a Livy server. If set to `true`, then all the incoming requests will be checked if the requested user has permission	false
livy.server.access_control.users	Users allowed to access Livy. By default, any user is allowed to access Livy. If a user wants to limit the access, the user should list all the permitted users separated by a comma	livy,hdfs,spark
livy.superusers	A list of comma-separated users that have the permissions to change other user's submitted sessions, for example, submitting statements, deleting the session, and so on	livy,hdfs,spark
livy.keystore	A path to the keystore file. The path can be absolute or relative to the directory in which the process is started	—
livy.keystore.password	The password to access the keystore	—
livy.key-password	The password to access the key in the keystore	—

livy.server.thrift.ssl.protocol.blacklist	The list of banned TLS protocols	SSLv2,SSLv3,TLSv1,TLSv1.1

Spark heap memory settings

Parameter	Description	Default value
Spark History Server Heap Memory	Sets the maximum Java heap size for Spark History Server	1G
Spark3 Connect Heap Memory	Sets the maximum Java heap size for a Spark Connect server	1G

ranger-spark-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/hdfs_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Uses in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal's secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—
xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service that wants to log in	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal's key to be stored in the subject's private credentials	false
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal's key from the keytab	false

ranger-spark-security.xml

Parameter	Description	Default value

ranger.plugin.spark.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.spark.service.name	The name of the Ranger service containing policies for this instance	—
ranger.plugin.spark.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	/srv/ranger/spark/policycache
ranger.plugin.hive.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	The directory where Ranger policies for Hive are cached after successful retrieval from the source
ranger.plugin.spark.policy.pollIntervalMs	Defines how often to poll for changes in policies	30000
ranger.plugin.spark.policy.rest.client.connection.timeoutMs	The Spark plugin RangerRestClient connection timeout (in milliseconds)	120000
ranger.plugin.spark.policy.rest.client.read.timeoutMs	The Spark plugin RangerRestClient read timeout (in milliseconds)	30000
ranger.add-yarn-authorization	Set `true` to use only Ranger ACLs (i.e. ignore YARN ACLs)	false
ranger.plugin.spark.enable.implicit.userstore.enricher	Enables UserStoreEnricher for fetching user and group attributes when using macros or scripts in row filters starting Ranger 2.3	true
ranger.plugin.spark.policy.rest.ssl.config.file	The path to the RangerRestClient SSL configuration file for the Spark plugin	/etc/spark3/conf/ranger-spark-policymgr-ssl.xml

ranger-spark3-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/spark3/conf/ranger-spark3.jceks
xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/spark3/conf/ranger-spark3.jceks
xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

Other

Parameter	Description	Default value

Custom spark-defaults.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *spark-defaults.conf*	—
spark-env.sh	The contents of the *spark-env.sh* file used to initialize environment variables on worker nodes	spark-env.sh
Custom livy.conf	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *livy.conf*	—
livy-env.sh	The contents of the *livy-env.sh* file used to initialize environment variables for the Livy server operation	livy-env.sh
spark-history-env.sh	The contents of the *spark-history-env.sh* file used to initialize environment variables for the Spark3 History Server operation	spark-history-env.sh
Ranger plugin enabled	Enables or disables the Ranger plugin for Spark3	false

SSM

Credentials Encryption

Parameter	Description	Default value
Encryption enable	Set to `true` to enable credentials encryption	false
Credential provider path	The path to a keystore file used to encrypt credentials	jceks://file/etc/ssm/conf/ssm.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the auto-generated JCEKS keystore	false

smart-site.xml

Parameter	Description	Default value
smart.hadoop.conf.path	The path to the Hadoop configuration directory	/etc/hadoop/conf
smart.conf.dir	The path to the SSM configuration directory	/etc/ssm/conf
smart.server.rpc.address	The RPC address of the SSM Server	0.0.0.0:7042
smart.file.access.count.aggregator.failover	A failover strategy for the file access event aggregator. Possible values: `FAIL` — throws an exception, no failover. `SAVE_FAILED_WITH_RETRY` — saves all file access events that caused the exception	SAVE_FAILED_WITH_RETRY
smart.agent.master.address	The active SSM server's address	<hostname>
smart.agent.address	Defines the address of SSM Agent components on each host	0.0.0.0
smart.agent.port	The port number used by SSM agents to communicate with the SSM Server	7048
smart.agent.master.port	The port number used by the SSM Server to communicate with SSM agents	7051
smart.rest.server.port	The port of the SSM REST server	7045
smart.rest.server.security.enabled	The parameter enables or disables the SSM REST server security	false

smart.rest.server.auth.spnego.enabled	The parameter enables or disables the SPNEGO authentication for the SSM REST server	false
smart.rest.server.auth.predefined.enabled	The parameter enables or disables the basic authentication for users, listed in the `smart.rest.server.auth.predefined.users` option	false
smart.rest.server.auth.predefined.users	The list of users and their credentials that have access to the SSM REST server if the `smart.rest.server.auth.predefined.enabled` parameter is set to `True`	—
smart.ignore.dirs	A list of comma-separated HDFS directories to ignore. SSM will ignore all files under the given HDFS directories	—
smart.cover.dirs	A list of comma-separated HDFS directories where SSM scans for files. By default, all HDFS files are covered	—
smart.work.dir	The HDFS directory used by SSM as a working directory to store temporary files. SSM will ignore HDFS `inotify` events for all files under the working directory. Only one directory can be set	/system/ssm
smart.client.concurrent.report.enabled	Used to enable/disable concurrent reports for Smart Client. If enabled, Smart Client concurrently attempts to connect to multiple configured Smart Servers to find the active Smart Server, which is an optimization. Only the active Smart Server will respond to establish the connection. If the report has been successfully delivered to the active Smart Server, connection attempts to other Smart Servers are canceled	—
smart.server.rpc.handler.count	The number of RPC handlers on the server	80
smart.namespace.fetcher.batch	The batch size of the namespace fetcher. SSM fetches namespaces from the NameNode during the startup. Large namespaces may lead to long startup time. A larger batch size can speed up the fetcher efficiency and reduce the startup time	500
smart.namespace.fetcher.producers.num	The number of producers in the namespace fetcher	3
smart.namespace.fetcher.consumers.num	The number of consumers in the namespace fetcher	6
smart.rule.executors	The maximum number of rules that can be executed in parallel	5
smart.cmdlet.executors	The maximum number of cmdlets that can be executed in parallel	10
smart.dispatch.cmdlets.extra.num	The number of extra cmdlets dispatched by Smart Server	10
smart.cmdlet.dispatchers	The maximum number of cmdlet dispatchers that work in parallel	3
smart.cmdlet.mover.max.concurrent.blocks.per.srv.inst	The maximum number of file mover cmdlets that can be executed in parallel per SSM service. The `0` value removes the limit	0
smart.action.move.throttle.mb	The throughput limit (in MB) for the SSM move operation	0
smart.action.copy.throttle.mb	The throughput limit (in MB) for the SSM copy operation	0
smart.action.ec.throttle.mb	The throughput limit (in MB) for the SSM EC operation	0
smart.action.local.execution.disabled	Defines whether the active Smart Server can also execute actions like an agent. If set to `true`, the active SSM Server will NOT be able to execute actions. This configuration has no impact on a standby Smart Server	false

smart.cmdlet.max.num.pending	The maximum number of pending cmdlets in an SSM Server	20000
smart.cmdlet.hist.max.num.records	The maximum number of historic cmdlet records kept in an SSM server. SSM deletes the oldest cmdlets when this threshold is exceeded	100000
smart.cmdlet.hist.max.record.lifetime	The maximum lifetime of historic cmdlet records kept in an SSM server. The SSM Server deletes cmdlet records after the specified interval. Valid time units are `day`, `hour`, `min`, `sec`. The minimum update granularity is `5sec`	30day
smart.cmdlet.cache.batch	The maximum batch size of the cmdlet batch insert	600
smart.copy.scheduler.base.sync.batch	The maximum batch size of the Copy Scheduler base sync batch insert	500
smart.file.diff.max.num.records	The maximum file diff records with useless state	10000
smart.status.report.period	The status report period for actions in milliseconds	10
smart.status.report.period.multiplier	The report period multiplied by this value defines the largest report interval	50
smart.status.report.ratio	If the finished actions ratio equals or exceeds this value, a status report will be triggered	0.2
smart.top.hot.files.num	The number of top hot files displayed in web UI	200
smart.cmdlet.dispatcher.log.disp.result	Defines whether to log dispatch results for each cmdlet dispatched	false
smart.cmdlet.dispatcher.log.disp.metrics.interval	The time interval in milliseconds to log statistic metrics of the cmdlet dispatcher. If no cmdlets were dispatched within this interval, no output is generated for this interval. The `0` value disables the logger	5000
smart.compression.codec	The default compression codec for SSM compression (Zlib, Lz4, Bzip2, snappy). You can also specify codecs as action arguments, which overrides this setting	Zlib
smart.compression.max.split	The maximum number of chunks split for compression	1000
smart.compact.batch.size	The maximum number of small files to be compacted by the compact action	200
smart.compact.container.file.threshold.mb	The maximum size of a container file in MB	1024
smart.access.count.day.tables.num	The maximum number of tables that can be created in the Metastore database to store the file access count per day	30
smart.access.count.hour.tables.num	The maximum number of tables that can be created in the Metastore database to store the file access count per hour	48
smart.access.count.minute.tables.num	The maximum number of tables that can be created in the Metastore database to store the file access count per minute	120
smart.access.count.second.tables.num	The maximum number of tables that can be created in the Metastore database to store the file access count per second	30
smart.access.event.fetch.interval.ms	The interval in milliseconds between access event fetches	1000
smart.cached.file.fetch.interval.ms	The interval in milliseconds between fetches of cached files from HDFS	5000

smart.namespace.fetch.interval.ms	The interval in milliseconds between namespace fetches from HDFS	1
smart.mover.scheduler.storage.report.fetch.interval.ms	The interval in milliseconds between fetches of storage reports from HDFS DataNodes in the mover scheduler	120000
smart.metastore.small-file.insert.batch.size	The maximum size of the Metastore insert batch with information about small files	200
smart.agent.master.ask.timeout.ms	The maximum time in milliseconds for a Smart Agent to wait for a response from the Smart Server during the submission action	5000
smart.ignore.path.templates	A list of comma-separated regex templates of HDFS paths to be completely ignored by SSM	—
smart.internal.path.templates	A list of comma-separated regex templates of internal files to be completely ignored by SSM	.*/\..*,.*/__.*,.*_COPYING_.*
smart.security.enable	Enables Kerberos authentication for SSM	false
smart.server.keytab.file	The path to the SSM Server's keytab file	—
smart.server.kerberos.principal	The SSM Server's Kerberos principal	—
smart.agent.keytab.file	The path to the SSM Agent's keytab file	—
smart.agent.kerberos.principal	The SSM Agent's Kerberos principal	—
smart.rest.server.auth.spnego.principal	SSM REST server Kerberos principal	—
smart.rest.server.auth.spnego.keytab	SSM REST server keytab	—
smart.proxy.user.strategy	The scope of the LDAP user search. Possible values: ▪ `DISABLED` — impersonation is disabled, all actions are performed by the SSM node user (either the Kerberos principal or the user who started SSM). ▪ `NODE_SCOPE` — impersonation is enabled at the node level, all actions are performed by the user specified in the `smart.proxy.user` option. ▪ `CMDLET_SCOPE` — impersonation is enabled at the cmdlet level, all actions are performed by the cmdlet owner (currently, the cmdlet creator).	DISABLED
smart.proxy.users.cache.ttl	The minimum amount of time that must pass after the last access to a proxy users cache entry before it is evicted. The value must be specified in the `[Amount][TimeUnit]` format, where `Amount` is a number and `TimeUnit` is one of the following: ▪ `day` or `d` — for days; ▪ `hour` or `h` — for hours; ▪ `min` or `m` — for minutes; ▪ `sec` or `s` — for seconds.	2m
smart.proxy.users.cache.size	The maximum size of the proxy users cache	20

smart-env.sh

Parameter	Description	Default value
LD_LIBRARY_PATH	The path to extra native libraries for SSM	/usr/lib/hadoop/lib/native
HADOOP_HOME	The path to the Hadoop home directory	/usr/lib/hadoop

Other

Parameter	Description	Default value
Enable SmartFileSystem for Hadoop	When enabled, requests from different clients (Spark, HDFS, Hive, etc.) are taken into account when calculating `AccessCount` for files. Otherwise, the `AccessCount` value gets incremented only when a file is accessed from SSM	false
log4j.properties	The contents of the *log4j.properties* configuration file	—
Custom smart-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *smart-site.xml*	—
Custom smart-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *smart-env.sh*	—

SSM Server component

Druid configuration

Parameter	Description	Default value
db_url	The URL to the Metastore database	jdbc:postgresql://{{ groups['adpg.adpg'][0] | d(omit) }}:5432/ssm
db_user	The user name to connect to the database	ssm
db_password	The user password to connect to the database	—
initialSize	The initial number of connections created when the pool is started	10
minIdle	The minimum number of established connections that should be kept in the pool at all times. The connection pool can shrink below this number if validation queries fail	4
maxActive	The maximum number of active connections that can be allocated from this pool at the same time	50
maxWait	The maximum time in milliseconds the pool will wait (when there are no available connections) for a connection to be returned before throwing an exception	60000
timeBetweenEvictionRunsMillis	The time in milliseconds to sleep between the runs of the idle connection validation/cleaner thread. This value should not be set less than 1 second. It specifies how often to check for idle and abandoned connections, and how often to validate idle connections	90000
minEvictableIdleTimeMillis	The minimum amount of time an object may remain idle in the pool before it is eligible for eviction	300000
validationQuery	The SQL query used to validate connections from the pool before returning them to the caller	SELECT 1

testWhileIdle	Indicates whether connection objects are validated by the idle object evictor (if any)	true
testOnBorrow	Indicates whether objects are validated before being borrowed from the pool	false
testOnReturn	Indicates whether objects are validated before being returned to the pool	false
poolPreparedStatements	Enables the prepared statement pooling	true
maxPoolPreparedStatementPerConnectionSize	The maximum number of prepared statements that can be pooled per connection	30
removeAbandoned	A flag to remove abandoned connections if they exceed `removeAbandonedTimeout`	true
removeAbandonedTimeout	The timeout in seconds before an abandoned (in use) connection can be removed	180
logAbandoned	A flag to log stack traces for application code which abandoned a connection. Logging of abandoned connections adds extra overhead for every borrowed connection	true
filters	Sets the filters that are applied to the data source	stat

Trino

Hive configuration

Parameter	Description	Default value
connector.name	Connector type	hive
hive.metastore.uri	URI for the Hive Metastore service	—
hive.storage-format	Storage format for the Hive data	PARQUET
hive.compression-codec	Compression codec for the Hive data	SNAPPY
hive.metastore.thrift.impersonation.enabled	Defines whether the Thrift impersonation mechanism is enabled for the Hive Metastore requests	true
fs.hadoop.enabled	Defines whether the HDFS access is supported	false
hive.config.resources	Optional comma-separated list of HDFS configuration files	/etc/hadoop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml
hive.non-managed-table-writes-enabled	Defines whether writing to non-managed (external) Hive tables is enabled	true
hive.metastore.authentication.type	Defines whether Kerberos authentication is enabled for Hive Metastore access	—
hive.metastore.service.principal	Kerberos principal for the Hive Metastore service	—
hive.metastore.client.principal	Kerberos principal for the Trino client connecting to Hive Metastore	—
hive.metastore.client.keytab	Path to the keytab file for the Trino Kerberos client	—
hive.hdfs.authentication.type	Defines whether the Kerberos authentication is enabled for HDFS access	—

hive.hdfs.impersonation.enabled	Defines whether the impersonation mechanism is used for end users to access the HDFS	false
hive.hdfs.trino.principal	Trino client Kerberos principal for connection to the Hive Metastore	—
hive.hdfs.trino.keytab	Path to the keytab file for the Trino Kerberos client	—
hive.metastore.thrift.client.ssl.enabled	Defines whether SSL is enabled for the Hive Metastore Thrift client	false
hive.metastore.thrift.client.ssl.key	Path to the keystore for SSL authentication	—
hive.metastore.thrift.client.ssl.key-password	Password for the keystore used in SSL authentication	—
hive.metastore.thrift.client.ssl.trust-certificate	Path to the truststore for SSL authentication	—
hive.metastore.thrift.client.ssl.trust-certificate-password	Password for the truststore used in SSL authentication	—

Iceberg configuration

Parameter	Description	Default value
connector.name	Connector type	iceberg
hive.metastore.uri	URI for the Hive Metastore service	—
iceberg.catalog.type	Catalog type for Iceberg	hive_metastore
iceberg.file-format	File format for Iceberg tables	PARQUET
iceberg.compression-codec	Compression codec for Iceberg tables	SNAPPY
hive.metastore.thrift.impersonation.enabled	Defines whether the Thrift impersonation mechanism is enabled for the Hive Metastore requests	true
fs.hadoop.enabled	Defines whether the HDFS access is supported	false
hive.config.resources	Optional comma-separated list of HDFS configuration files	/etc/hadoop/conf/core-site.xml,/etc/hadoop/conf/hdfs-site.xml
hive.metastore.authentication.type	Defines whether Kerberos authentication is enabled for Hive Metastore access	—
hive.metastore.service.principal	Kerberos principal for the Hive Metastore service	—
hive.metastore.client.principal	Kerberos principal for the Trino client connecting to Hive Metastore	—
hive.metastore.client.keytab	Path to the keytab file for the Trino Kerberos client	—
hive.hdfs.authentication.type	Defines whether the Kerberos authentication is enabled for HDFS access	—
hive.hdfs.impersonation.enabled	Defines whether the impersonation mechanism is used for end users to access the HDFS	false

hive.hdfs.trino.principal	Trino client Kerberos principal for connection to the Hive Metastore	—
hive.hdfs.trino.keytab	Path to the keytab file for the Trino Kerberos client	—
hive.metastore.thrift.client.ssl.enabled	Defines whether SSL is enabled for the Hive Metastore Thrift client	false
hive.metastore.thrift.client.ssl.key	Path to the keystore for SSL authentication	—
hive.metastore.thrift.client.ssl.key-password	Password for the keystore used in SSL authentication	—
hive.metastore.thrift.client.ssl.trust-certificate	Path to the truststore for SSL authentication	—
hive.metastore.thrift.client.ssl.trust-certificate-password	Password for the truststore used in SSL authentication	—

Custom catalogs

Parameter	Description	Default value
Connector properties	In this section you can add custom catalog records by clicking ***+1*** and entering the properties for the new connector	—

resource-groups.properties

Parameter	Description	Default value
resource-groups.configuration-manager	Source of the configuration for the resource group manager: JSON file or database	file
resource-groups.config-db-url	URL of the database to load the configuration from	—
resource-groups.config-db-user	Username for the database connection	—
resource-groups.config-db-password	Password for the database connection	—
resource-groups.config-file	Path to the JSON configuration file for the resource group manager	/etc/trino/conf/resource-groups.json
resource-groups.max-refresh-interval	Maximum time period for which the cluster will continue to accept queries after configuration refresh failures. After this period the configuration will become invalidated	—
resource-groups.refresh-interval	Time interval (in seconds) for reloading the configuration	—

session-property-config.properties

Parameter	Description	Default value
Session property managers	Source of the configuration for the session property managers	file

session-property-manager.config-file	Path to the JSON configuration file for the session property managers	/etc/trino/conf/session-property-config.json

Fault-tolerant execution

Parameter	Description	Default value
retry-policy	Defines whether Trino should retry entire query, its individual tasks, or do nothing in the event of failure. The parameter accepts the following values: - `NONE` — disables FTE (default). - `QUERY` — if a Trino worker fails, Trino retries the entire query. This policy is recommended when the majority of Trino requests are small queries. - `TASK` — if a Trino worker fails, Trino retries individual query tasks. This policy is best suited for retrying large batch queries, however, it can result in higher latency for big number of short-running queries.	NONE
exchange.deduplication-buffer-size	Data size of the coordinator in-memory buffer used by fault-tolerant execution to store output of query stages	32MB
fault-tolerant-execution-exchange-encryption-enabled	Defines whether to use encryption for spooling data	false
task.low-memory-killer.policy	Defines whether to kill tasks in case of low memory availability. Takes one of the following values: - `NONE` — do not kill tasks. - `total-reservation-on-blocked-nodes` — kill the tasks that are part of the queries which are currently using the most memory specifically on nodes that are now out of memory. - `least-waste` — kill the tasks that are part of the queries which use significant amount of memory on nodes which are now out of memory. This policy avoids killing tasks which are executing for a long time in order not to waste the work done by them. This parameter applies only if the `retry-policy` parameter has the `TASK` value	total-reservation-on-blocked-nodes
query-retry-attempts	Maximum number of times that Trino can attempt to retry a query before it is considered failed. This parameter applies only if the `retry-policy` parameter has the `QUERY` value	4
task-retry-attempts-per-task	Maximum number of times that Trino can attempt to retry a single task before its parent query is considered failed. This parameter applies only if the `retry-policy` parameter has the `TASK` value	4
retry-initial-delay	Minimum interval of time that a failed query or task must wait before it is retried. This parameter applies only if the `retry-policy` parameter has the `TASK` or `QUERY` value	10s
retry-max-delay	Maximum interval of time that a failed query or task must wait before it is retried. Wait time is increased on each subsequent failure. This parameter applies only if the `retry-policy` parameter has the `TASK` or `QUERY` value	1m
retry-delay-scale-factor	Factor by which retry delay is increased on each query or task failure. This parameter applies only if the `retry-policy` parameter has the `TASK` or `QUERY` value	2

exchange-manager.properties

Parameter	Description	Default value
exchange-manager.name	Type of storage for the intermediate data exchange between stages of a distributed query: HDFS or local filesystem	hdfs
exchange.base-directories	Comma-separated list of location URIs that the Exchange manager uses to store spooling data	hdfs://<adh_cluster_nameservice>/exchange-spooling-directory
hdfs.config.resources	Comma-separated list of paths to HDFS configuration files. The files must exist on all nodes of the Trino cluster	/etc/hadoop/conf/core-site.xml, /etc/hadoop/conf/hdfs-site.xml
exchange.hdfs.auto-create-storage-dirs	Defines whether to automatically create HDFS storage directories if they do not exist	false
exchange.hdfs.auto-create-storage-dirs-perms	File system permissions for automatically created storage directories	755
exchange.sink-buffer-pool-min-size	Minimum buffer pool size for an exchange sink. The larger the buffer pool size, the larger the write parallelism and memory usage	10
exchange.sink-buffers-per-partition	Number of buffers per partition in the buffer pool. The larger the buffer pool size, the larger the write parallelism and memory usage	2

log.properties

Parameter	Description	Default value
io.trino	Global logging level for Trino	INFO
io.trino.server	Logging level for Trino server components	INFO

ranger-trino-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	Spool directory path	/srv/ranger/trino_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Whether to use in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	Name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	Name of the keytab file to get the principal's secret key	—

xasecure.audit.jaas.Client.option.principal	Name of the principal to be used	—
xasecure.audit.jaas.Client.option.serviceName	Name of a user or a service for login	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal's key to be stored in the subject's private credentials	—
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal's key from the keytab	—
xasecure.audit.jaas.Client.option.useTicketCache	Set this to `true` if you want the module to get the principal's key from the cache	—

ranger-trino-security.xml

Parameter	Description	Default value
ranger.plugin.trino.access.cluster.name	Name to identify the Trino cluster. It is recorded in audit logs generated by the plugin	—
ranger.plugin.trino.super.users	Comma-separated list of usernames that have the superuser privileges	—
ranger.plugin.trino.super.groups	Comma-separated list of group names. Users from these groups are treated as superusers	—
ranger.plugin.trino.use.rangerGroups	Defines whether to obtain user-to-groups mapping from Apache Ranger	—
ranger.plugin.trino.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.trino.service.name	Name of the Ranger service containing policies for this Trino instance	—
ranger.plugin.trino.policy.cache.dir	Directory where Ranger policies are cached after a successful retrieval from the source	/srv/ranger/trino/policycache
ranger.plugin.trino.policy.pollIntervalMs	Defines how often to poll for changes in policies (in milliseconds)	30000
ranger.plugin.trino.policy.rest.client.connection.timeoutMs	Trino plugin connection timeout in milliseconds	120000
ranger.plugin.trino.policy.rest.client.read.timeoutMs	Trino plugin read timeout in milliseconds	30000
ranger.plugin.trino.enable.implicit.userstore.enricher	Defines whether to use the UserStoreEnricher for fetching user and group attributes if using macros or scripts in row-filters since Ranger 2.3	true
ranger.plugin.trino.policy.rest.ssl.config.file	Path to the RangerRestClient SSL config file for the Trino plugin	/etc/trino/conf/coordinator/ranger-trino-policymgr-ssl.xml
ranger.plugin.trino.ugi.initialize	Defines whether to initialize the Kerberos ID used to authenticate with Ranger Admin server	true
ranger.plugin.trino.ugi.login.type	Type of login. Use the `keytab` value	keytab

ranger.plugin.trino.ugi.keytab.principal	Kerberos principal	—
ranger.plugin.trino.ugi.keytab.file	Location of the keytab file	—

ranger-trino-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	Path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	Path to the keystore credentials file	/etc/trino/conf/ranger-trino.jceks
xasecure.policymgr.clientssl.truststore.credential.file	Path to the truststore credentials file	/etc/trino/conf/ranger-trino.jceks
xasecure.policymgr.clientssl.truststore	Path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	Password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	Password to the truststore file	—

access-control.properties

Parameter	Description	Default value
access-control.name	Type of access control. Possible values: ▪ `default` — all operations are permitted, except for user impersonation and triggering graceful shutdown. ▪ `allow-all` — all operations are permitted. ▪ `read-only` — reading data or metadata is permitted, writing data or metadata is forbidden. ▪ `file` — authorization rules are specified in a configuration file. ▪ `opa` — Open Policy Agent (OPA) is used for authorization. ▪ `ranger` — Apache Ranger policies are used for authorization.	default
ranger.service.name	Name of the Ranger service that has the policies to enforce	—
ranger.plugin.config.resource	Comma-separated list of Ranger plugin configuration files. Relative paths are resolved dynamically by searching in the classpath	—
ranger.hadoop.config.resource	Comma-separated list of Hadoop configuration files. Relative paths are resolved dynamically by searching in the classpath	—
security.config-file	Path to the configuration file with authorization rules	/etc/trino/conf/rules.json
opa.policy.uri	URI of the open policy agent (OPA)	—

Others

Parameter	Description	Default value
Custom Hive configuration	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for Hive configuration	—
Custom Iceberg configuration	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for Iceberg configuration	—
Custom resource-groups.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for the resource group manager properties	—
Custom session-property-config.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for the session property managers properties	—
Custom exchange-manager.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for the Exchange manager properties	—
Custom log.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed for the logging properties	—
Custom ranger-trino-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-trino-audit.xml*	—
Custom ranger-trino-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-trino-security.xml*	—
Custom ranger-trino-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-trino-policymgr-ssl.xml*	—
Ranger plugin enabled	Defines whether to use the Ranger plugin for Trino	false
resource-groups.json	JSON file containing the configuration for the resource group manager	resource-groups.json
session-property-config.json	JSON file containing the configuration for the session property managers	session-property-config.json
ad-runtime-utils	YAML configuration file that contains the Java installation settings: version, installation path, and the environment variable	ad-runtime-utils.yml

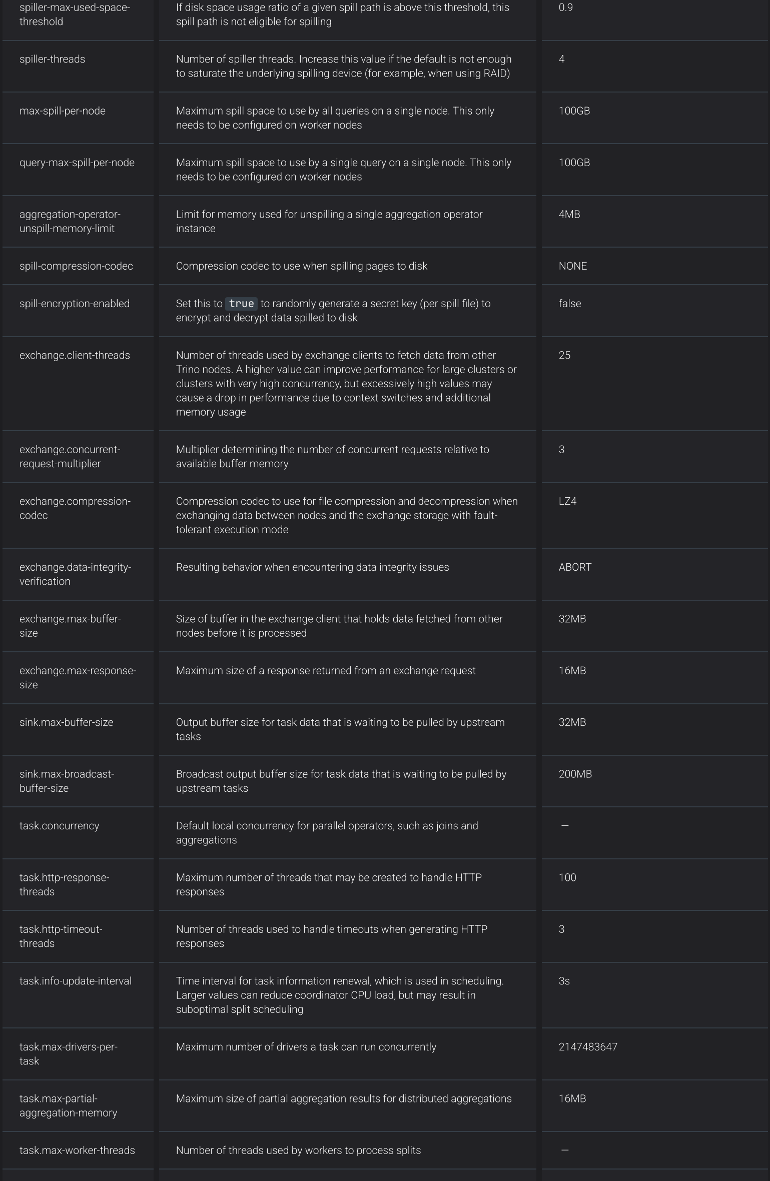
The Trino Coordinator component

config.properties

Parameter	Description	Default value
coordinator	Specifies whether this node acts as a coordinator	true
http-server.authentication.allow-insecure-over-http	Defines whether non-secure HTTP connections are allowed when authentication is enabled	true

node-scheduler.include-coordinator	Defines whether the coordinator node should also schedule and run queries	true
http-server.http.port	HTTP port of the server	18188
discovery.uri	URI for the discovery service. Typically, the fully qualified domain name (FQDN) with the server port	—
join-distribution-type	Type of distributed JOIN to use	AUTOMATIC
redistribute-writes	Defines whether to redistribute data before writing	true
query.max-cpu-time	Maximum amount of CPU time that a query can use across the entire cluster. Queries that exceed this limit are killed	1_000_000_000d
query.max-memory-per-node	Maximum amount of user memory a query can use on a worker	—
query.max-memory	Maximum amount of user memory a query can use across the entire cluster	20GB
query.max-total-memory	Maximum amount of memory a query can use across the entire cluster, including revocable memory	40GB
memory.heap-headroom-per-node	Amount of memory set aside as headroom (buffer) in the JVM heap for allocations that are not tracked by Trino	—
exchange.deduplication-buffer-size	Size of the buffer used for spooled data during fault-tolerant execution	—
query.client.timeout	Timeout for the cluster to run without contact from the client application before it cancels its work	5m
query.execution-policy	Algorithm for organization of the processing of all the stages in a query	phased
query.determine-partition-count-for-write-enabled	Determines whether the number of partitions is based on amount of data read and processed by the query for write queries	false
query.max-hash-partition-count	Maximum number of partitions to use for processing distributed operations	100
query.min-hash-partition-count-for-write	Minimum number of partitions to use for processing distributed operations in write queries	50
query.max-writer-task-count	Maximum number of tasks that will take part in writing data during `INSERT`, `CREATE TABLE AS SELECT`, and `EXECUTE` queries. The limit is only applicable when the `redistribute-writes` or `scale-writers` parameter is enabled	100
query.low-memory-killer.policy	Defines the mode of dealing with running queries in the event of low memory availability	total-reservation-on-blocked-nodes
query.max-execution-time	Maximum time of query execution in the cluster before it is terminated. Execution time does not include analysis, query planning, or waiting in queue	100d
query.max-length	Maximum number of characters allowed for the SQL query text. Longer queries are not processed and terminated with the `QUERY_TEXT_TOO_LARGE` error	1000000
query.max-planning-time	Maximum allowed time for a query to plan the execution	10m
query.max-run-time	Maximum allowed time for a query to be processed in the cluster before it is terminated	100d
query.max-scan-physical-bytes	Maximum number of bytes that can be scanned by a query during its execution	—

query.remote-task.max-error-duration	Timeout for the remote tasks that fail to communicate with the coordinator. If the coordinator is unable to receive updates from a remote task before this value is reached, the coordinator treats the task as failed	1m
query.max-stage-count	Maximum number of stages allowed to be generated per query	150
query.max-history	Maximum number of queries to keep in the query history to provide statistics and other information, and make the data available in the web UI	100
query.min-expire-age	Minimum age of a query in the history before it is expired. An expired query is removed from the query history buffer and is no longer available in the web UI	15m
query.remote-task.enable-adaptive-request-size	Defines whether dynamic splitting up of server requests sent by tasks is enabled. This can prevent out-of-memory errors for large schemas	true
query.remote-task.guaranteed-splits-per-task	Minimum number of splits that should be assigned to each remote task to ensure that each task has a minimum amount of work to perform	3
query.remote-task.max-request-size	Maximum size of a single request made by a remote task. Requires the `query.remote-task.enable-adaptive-request-size` parameter to be enabled	8MB
query.remote-task.request-size-headroom	Determines the amount of headroom that should be allocated beyond the size of the request data. Requires the `query.remote-task.enable-adaptive-request-size` parameter to be enabled	2MB
query.info-url-template	URL for redirection of clients to an alternative location for query information	—
retry-policy	Retry policy to use for fault-tolerant execution	NONE
catalog.prune.update-interval	Interval for pruning dropped catalogs. Requires the `catalog.management` parameter to be set to `dynamic`. Dropping a catalog does not interrupt any running queries that use it, but makes it unavailable to any new queries	5s
catalog.store	When set to `file`, creating and dropping catalogs using the SQL commands adds and removes catalog property files on the coordinator node. Requires the `catalog.management` parameter to be set to `dynamic`	—
sql.forced-session-time-zone	Defines whether to force set the time zone for any query processing to the configured value, and therefore override the time zone of the client. The time zone must be specified as a string	—
sql.default-catalog	Default catalog for all clients. Any default catalog configuration provided by a client overrides this default	—
sql.default-schema	Default schema for all clients. Must be set to a schema name that is valid for the default catalog. Any default schema configuration provided by a client overrides this default	—
sql.default-function-catalog	Default catalog for user-defined functions storage for all clients	—
sql.default-function-schema	Default schema for UDF storage for all clients	—
sql.path	Default collection of paths to functions or table functions in specific catalogs and schemas. Paths are specified as `catalog_name.schema_name`	—
spill-enabled	Defines whether spilling memory to disk to avoid exceeding memory limits for the query is enabled	false
spiller-spill-path	Path for spilling memory to disk	—



spiller-max-used-space-threshold	If disk space usage ratio of a given spill path is above this threshold, this spill path is not eligible for spilling	0.9
spiller-threads	Number of spiller threads. Increase this value if the default is not enough to saturate the underlying spilling device (for example, when using RAID)	4
max-spill-per-node	Maximum spill space to use by all queries on a single node. This only needs to be configured on worker nodes	100GB
query-max-spill-per-node	Maximum spill space to use by a single query on a single node. This only needs to be configured on worker nodes	100GB
aggregation-operator-unspill-memory-limit	Limit for memory used for unspilling a single aggregation operator instance	4MB
spill-compression-codec	Compression codec to use when spilling pages to disk	NONE
spill-encryption-enabled	Set this to `true` to randomly generate a secret key (per spill file) to encrypt and decrypt data spilled to disk	false
exchange.client-threads	Number of threads used by exchange clients to fetch data from other Trino nodes. A higher value can improve performance for large clusters or clusters with very high concurrency, but excessively high values may cause a drop in performance due to context switches and additional memory usage	25
exchange.concurrent-request-multiplier	Multiplier determining the number of concurrent requests relative to available buffer memory	3
exchange.compression-codec	Compression codec to use for file compression and decompression when exchanging data between nodes and the exchange storage with fault-tolerant execution mode	LZ4
exchange.data-integrity-verification	Resulting behavior when encountering data integrity issues	ABORT
exchange.max-buffer-size	Size of buffer in the exchange client that holds data fetched from other nodes before it is processed	32MB
exchange.max-response-size	Maximum size of a response returned from an exchange request	16MB
sink.max-buffer-size	Output buffer size for task data that is waiting to be pulled by upstream tasks	32MB
sink.max-broadcast-buffer-size	Broadcast output buffer size for task data that is waiting to be pulled by upstream tasks	200MB
task.concurrency	Default local concurrency for parallel operators, such as joins and aggregations	—
task.http-response-threads	Maximum number of threads that may be created to handle HTTP responses	100
task.http-timeout-threads	Number of threads used to handle timeouts when generating HTTP responses	3
task.info-update-interval	Time interval for task information renewal, which is used in scheduling. Larger values can reduce coordinator CPU load, but may result in suboptimal split scheduling	3s
task.max-drivers-per-task	Maximum number of drivers a task can run concurrently	2147483647
task.max-partial-aggregation-memory	Maximum size of partial aggregation results for distributed aggregations	16MB
task.max-worker-threads	Number of threads used by workers to process splits	—

task.min-drivers	Target number of running leaf splits on a worker	—
task.min-drivers-per-task	Minimum number of drivers guaranteed to run concurrently for a single task given the task has remaining splits to process	3
task.min-writer-count	Minimum number of concurrent writer threads per worker per query when preferred partitioning and task writer scaling are not used	1
task.max-writer-count	Maximum number of concurrent writer threads per worker per query when preferred partitioning and task writer scaling are not used	—
task.interrupt-stuck-split-tasks-enabled	Set this to `true` to allow Trino to detect and fail tasks containing splits that have been stuck	true
task.interrupt-stuck-split-tasks-warning-threshold	Defines the splits running time threshold after which call stacks at */v1/maxActiveSplits* endpoint are printed out and JMX metrics are generated	10m
task.interrupt-stuck-split-tasks-timeout	Timeout for a blocked split processing thread before Trino fails the task	10m
task.interrupt-stuck-split-tasks-detection-interval	Interval of Trino checks for splits that have processing time exceeding the `task.interrupt-stuck-split-tasks-timeout` parameter value	2m
use-preferred-write-partitioning	Defines whether preferred write partitioning is enabled. When set to `true`, each partition is written by a separate writer	true
scale-writers	Defines whether writer scaling by dynamically increasing the number of writer tasks on the cluster is enabled	true
task.scale-writers.enabled	Defines whether scaling the number of concurrent writers within a task is enabled	true
writer-scaling-min-data-processed	Minimum amount of uncompressed data that must be processed by a writer before another writer can be added	100MB
optimizer.dictionary-aggregation	Defines whether optimization for aggregations on dictionaries is used	false
optimizer.optimize-hash-generation	Set this to `true` to compute hash codes for distribution, joins, and aggregations early during execution, allowing result to be shared between operations later in the query	false
optimizer.optimize-metadata-queries	Defines whether to push the aggregation below the outer join when an aggregation is above an outer join and all columns from the outer side of the join are in the grouping clause	true
optimizer.distinct-aggregations-strategy	Strategy to use for multiple distinct aggregations	AUTOMATIC
optimizer.push-table-write-through-union	Defines whether to parallelize writes when using `UNION ALL` in queries that write data	true
optimizer.join-reordering-strategy	Join reordering strategy to use. `NONE` maintains the order the tables are listed in the query	AUTOMATIC
optimizer.max-reordered-joins	Maximum number of joins that can be reordered at once when the `optimizer.join-reordering-strategy` is set to `cost-based`	8
optimizer.optimize-duplicate-insensitive-joins	Defines whether to reduce the number of rows produced by joins when optimizer detects that duplicated join output rows can be skipped	true
optimizer.use-exact-partitioning	Defines whether to re-partition the data unless the partitioning of the upstream stage exactly matches what the downstream stage expects	false
optimizer.use-table-scan-node-partitioning	Defines whether to use the connector provided table node partitioning when reading tables	true

optimizer.table-scan-node-partitioning-min-bucket-to-task-ratio	Specifies minimal bucket to task ratio that has to be matched or exceeded in order to use table scan node partitioning	0.5
optimizer.filter-conjunction-independence-factor	Scales the strength of independence assumption for estimating the selectivity of the conjunction of multiple predicates	0.75
optimizer.join-multi-clause-independence-factor	Scales the strength of independence assumption for estimating the output of a multi-clause join	0.25
optimizer.non-estimatable-predicate-approximation.enabled	Defines whether to use the cost based optimizer to determine if repartitioning the output of an already partitioned stage is necessary	true
optimizer.join-partitioned-build-min-row-count	Minimum number of join build side rows required to use partitioned join lookup	1000000
optimizer.min-input-size-per-task	Minimum input size required per task	5GB
optimizer.min-input-rows-per-task	Minimum number of input rows required per task	10000000
log.annotation-file	Name of an optional properties file that contains annotations to include with each log message for TCP output or file output in JSON format, defined by the `log.path` and `log.format` parameters	—
log.path	Path to the log file used by Trino. The path is relative to the data directory, configured to `var/log/server.log` by the launcher script	—
log.max-size	Maximum size for the general application log file	100MB
log.max-total-size	Maximum total size for the general application log files	1GB
log.compression	Compression format for rotated log files. Can be set to either `GZIP` or `NONE`. When set to `NONE`, compression is disabled	GZIP
http-server.log.enabled	Defines whether the logging for the HTTP server is enabled	true
http-server.log.compression.enabled	Defines whether the compression of the log files of the HTTP server is enabled	true
http-server.log.path	Path to the log file of the HTTP server	/var/log/trino/http-request.log
http-server.log.max-history	Maximum number of log files for the HTTP server to use before log rotation	15
http-server.log.max-size	Maximum file size for the log file of the HTTP server	—
re2j.dfa-states-limit	Maximum number of states to use when RE2J builds the fast, but potentially memory intensive, deterministic finite automaton (DFA) for regular expression matching	2147483647
re2j.dfa-retries	Number of times that RE2J retries the DFA algorithm, when it reaches a states limit before using the slower, but less memory intensive NFA algorithm, for all future inputs for that search	5
http-server.authentication.type	Authentication mechanism to allow user access to Trino	—
node.internal-address-source	Set this to `FQDN` to ensure correct operation and usage of valid DNS host names by Kerberos	FQDN
access-control.config-files	Path to the access control configuration file	/etc/trino/conf/access-control.properties

password-authenticator.config-files	Password Authenticator configuration file	/etc/trino/conf/coordinator/password-authenticator.properties
web-ui.user	User name for authentication in the web UI	trino
internal-communication.shared-secret	String used by coordinators and workers of the same cluster to authenticate within it	—
web-ui.authentication.type	Authentication mechanism that is used to allow user access to the UI	NONE

Kerberos Configuration

Parameter	Description	Default value
http-server.authentication.krb5.service-name	Kerberos service name for authentication	HTTP
http-server.authentication.krb5.principal-hostname	Principal hostname for Kerberos authentication	—
http-server.authentication.krb5.keytab	Path to the keytab file for Kerberos authentication	/etc/security/keytabs/HTTP.service.keytab
http.authentication.krb5.config	Path to the Kerberos configuration file	/etc/krb5.conf
http-server.authentication.krb5.user-mapping.file	File containing rules for mapping Kerberos principal names to local usernames	—

SSL Configuration

Parameter	Description	Default value
http-server.http.enabled	Defines whether HTTP is enabled for the HTTP server	true
internal-communication.https.required	Defines whether SSL/TLS is used for all internal communications	false
http-server.https.enabled	Defines whether HTTPS is enabled for the HTTP server	false
http-server.https.port	HTTPS port of the server	18188
http-server.https.keystore.path	Path to the keystore file for HTTPS	—
http-server.https.keystore.key	Password for the keystore used in HTTPS	—
http-server.https.truststore.path	Path to the truststore file for HTTPS	—
http-server.https.truststore.key	Password for the truststore used in HTTPS	—
internal-communication.https.keystore.path	Path to the keystore file for internal communication within Trino cluster via HTTPS	—

internal-communication.https.keystore.key	Password for the keystore used in internal communication within Trino cluster via HTTPS	—
internal-communication.https.truststore.path	Path to the truststore file for internal communication within Trino cluster via HTTPS	—
internal-communication.https.truststore.key	Password for the truststore used in internal communication within Trino cluster via HTTPS	—

Enable LDAP

Parameter	Description	Default value
password-authenticator.name	Specifies the name of the authenticator implementation	ldap
ldap.url	URL of the LDAP server	—
ldap.ssl.truststore.path	Path to the PEM or JKS keystore file	—
ldap.user-bind-pattern	LDAP user bind string for password authentication. Must contain the pattern `${USER}` which is replaced by the actual username during the password authentication. Can contain multiple patterns separated by a colon	${USER}@example.com
ldap.allow-insecure	Defines whether connecting to the LDAP server without TLS is allowed	false
ldap.user-base-dn	Base LDAP distinguished name for the user who tries to connect to the server. Example: `OU=Peoples,DC=example,DC=com`	—
ldap.bind-dn	Bind distinguished name used by Trino when issuing group membership queries. Example: `CN=admin,OU=CITY_OU,DC=domain`	—
ldap.bind-password	Bind password used by Trino when issuing group membership queries	—
ldap.group-auth-pattern	Specifies the LDAP query for the LDAP group membership authorization	(&(objectClass=person)(sAMAccountName=${USER})(memberof=CN=AuthorizedGroup,OU=Groups,DC=example,DC=com))

OAuth 2.0

Parameter	Description	Default value
http-server.authentication.oauth2.issuer	URL of the OAuth2 authorization server	—
http-server.authentication.oauth2.client-id	OAuth2 client identifier for Trino	—
http-server.authentication.oauth2.client-secret	OAuth2 client secret for authentication	—

JWT Auth

Parameter	Description	Default value
http-server.authentication.jwt.key-file	Specifies either the URL to a JWKS service or the path to a PEM or HMAC file	—

http-server.authentication.jwt.required-issuer	Specifies a string that must match the issuer of JWT (`iss`) field	—
http-server.authentication.jwt.required-audience	Specifies a string that must match the audience of JWT (`aud`) field	—
http-server.authentication.jwt.principal-field	Field in JWT that identifies the subject	sub
http-server.authentication.jwt.user-mapping.pattern	Regular expression pattern to map all user names for JWT authentication to the format expected by Trino	—
http-server.authentication.jwt.user-mapping.file	JSON file with user mapping rules for JWT authentication	—

Certificate Auth

Parameter	Description	Default value
http-server.authentication.certificate.user-mapping.pattern	Regular expression pattern to map all user names for certificate-based authentication to the format expected by Trino	—
http-server.authentication.certificate.user-mapping.file	Path to a JSON file with user mapping rules for certificate-based authentication	/etc/trino/conf/coordinator/user-mapping.json

Password-file Auth

Parameter	Description	Default value
password-authenticator.name	Specifies the name of the authenticator implementation	file
file.password-file	Path to the password file	—
file.refresh-period	Defines how often to reload the password file	5s
file.auth-token-cache.max-size	Maximum number of cached authenticated passwords	1000

Trino catalog management

Parameter	Description	Default value
catalog.management	Catalog manager type	static
Postgres JDBC URL	JDBC connection URL of the Postgres catalog store	jdbc:postgresql://{{ groups['adpg.adpg'][0] | d(omit) }}:5432/catalog
Postgres username	Postgres catalog store JDBC connection username	catalog
Postgres password	Postgres catalog store JDBC connection password	—
Enable cleanup	Defines whether the cleanup of the old version catalogs is enabled	true
Encryption password	Password for the properties encryption	—
RegEx pattern	Regular expression pattern for defining properties to be encrypted	—

Cleanup old version catalogs	Old version catalogs cleanup interval. Minimal value is `10d`	30d
Time to live of old version catalogs	Time-to-live (TTL) of old version catalogs. Minimal value is `1d`	365d

node.properties

Parameter	Description	Default value
node.environment	Name of the environment in which the node operates. Must be the same for all Trino nodes in a cluster	adh
node.id	Unique identifier of the node	—
node.data-dir	Path to the directory for node data storage	/srv/trino/data/coordinator
node.server-log-file	Path to the server log file for the node	/var/log/trino/coordinator/server.log
node.launcher-log-file	Path to the launcher log file for the node	/var/log/trino/coordinator/launcher.log

Others

Parameter	Description	Default value
user-mapping.json	Contents of the *user-mapping.json* file that contains regular expressions to map all user names to the format expected by Trino	user-mapping.json
env.sh	Contents of the *env.sh* file that contains Trino environment settings	env.sh
jvm.config	Contents of the *jvm.config* file that contains the Java virtual machine settings for Trino	jvm.config
Custom password-authenticator.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *password-authenticator.properties*	—
Custom config.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *config.properties*	—
Custom node.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *node.properties*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

Enable custom ulimits — default value:

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)
DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Trino Worker component

config.properties

Parameter	Description	Default value
coordinator	Specifies whether this node acts as a coordinator	false
http-server.authentication.allow-insecure-over-http	Defines whether non-secure HTTP connections are allowed when authentication is enabled	true
http-server.http.port	HTTP port of the server	18189
discovery.uri	URI for the discovery service. Typically, the fully qualified domain name (FQDN) with the server port	—
join-distribution-type	Type of distributed JOIN to use	AUTOMATIC
redistribute-writes	Defines whether to redistribute data before writing	true
query.max-cpu-time	Maximum amount of CPU time that a query can use across the entire cluster. Queries that exceed this limit are killed	—

query.max-memory-per-node	Maximum amount of user memory a query can use on a worker	—
query.max-memory	Maximum amount of user memory a query can use across the entire cluster	20GB
query.max-total-memory	Maximum amount of memory a query can use across the entire cluster, including revocable memory	40GB
memory.heap-headroom-per-node	Amount of memory set aside as headroom (buffer) in the JVM heap for allocations that are not tracked by Trino	—
exchange.deduplication-buffer-size	Size of the buffer used for spooled data during fault-tolerant execution	—
query.client.timeout	Timeout for the cluster to run without contact from the client application before it cancels its work	5m
query.execution-policy	Algorithm for organization of the processing of all the stages in a query	phased
query.determine-partition-count-for-write-enabled	Determines whether the number of partitions is based on amount of data read and processed by the query for write queries	false
query.max-hash-partition-count	Maximum number of partitions to use for processing distributed operations	100
query.min-hash-partition-count-for-write	Minimum number of partitions to use for processing distributed operations in write queries	50
query.max-writer-task-count	Maximum number of tasks that will take part in writing data during `INSERT`, `CREATE TABLE AS SELECT`, and `EXECUTE` queries. The limit is only applicable when the `redistribute-writes` or `scale-writers` parameter is enabled	100
query.low-memory-killer.policy	Defines the mode of dealing with running queries in the event of low memory availability	total-reservation-on-blocked-nodes
query.max-execution-time	Maximum time of query execution in the cluster before it is terminated. Execution time does not include analysis, query planning, or waiting in queue	100d
query.max-length	Maximum number of characters allowed for the SQL query text. Longer queries are not processed and terminated with the `QUERY_TEXT_TOO_LARGE` error	1000000
query.max-planning-time	Maximum allowed time for a query to plan the execution	10m
query.max-run-time	Maximum allowed time for a query to be processed in the cluster before it is terminated	100d
query.max-scan-physical-bytes	Maximum number of bytes that can be scanned by a query during its execution	—
query.remote-task.max-error-duration	Timeout for the remote tasks that fail to communicate with the coordinator. If the coordinator is unable to receive updates from a remote task before this value is reached, the coordinator treats the task as failed	1m
query.max-stage-count	Maximum number of stages allowed to be generated per query	150
query.max-history	Maximum number of queries to keep in the query history to provide statistics and other information, and make the data available in the web UI	100
query.min-expire-age	Minimum age of a query in the history before it is expired. An expired query is removed from the query history buffer and is no longer available in the web UI	15m

query.remote-task.enable-adaptive-request-size	Defines whether dynamic splitting up of server requests sent by tasks is enabled. This can prevent out-of-memory errors for large schemas	true
query.remote-task.guaranteed-splits-per-task	Minimum number of splits that should be assigned to each remote task to ensure that each task has a minimum amount of work to perform	3
query.remote-task.max-request-size	Maximum size of a single request made by a remote task. Requires the `query.remote-task.enable-adaptive-request-size` parameter to be enabled	8MB
query.remote-task.request-size-headroom	Determines the amount of headroom that should be allocated beyond the size of the request data. Requires the `query.remote-task.enable-adaptive-request-size` parameter to be enabled	2MB
query.info-url-template	URL for redirection of clients to an alternative location for query information	—
retry-policy	Retry policy to use for fault-tolerant execution	NONE
sql.forced-session-time-zone	Defines whether to force set the time zone for any query processing to the configured value, and therefore override the time zone of the client. The time zone must be specified as a string	—
sql.default-catalog	Default catalog for all clients. Any default catalog configuration provided by a client overrides this default	—
sql.default-schema	Default schema for all clients. Must be set to a schema name that is valid for the default catalog. Any default schema configuration provided by a client overrides this default	—
sql.default-function-catalog	Default catalog for user-defined functions storage for all clients	—
sql.default-function-schema	Default schema for UDF storage for all clients	—
sql.path	Default collection of paths to functions or table functions in specific catalogs and schemas. Paths are specified as `catalog_name.schema_name`	—
spill-enabled	Defines whether spilling memory to disk to avoid exceeding memory limits for the query is enabled	false
spiller-spill-path	Path for spilling memory to disk	—
spiller-max-used-space-threshold	If disk space usage ratio of a given spill path is above this threshold, this spill path is not eligible for spilling	0.9
spiller-threads	Number of spiller threads. Increase this value if the default is not enough to saturate the underlying spilling device (for example, when using RAID)	4
max-spill-per-node	Maximum spill space to use by all queries on a single node. This only needs to be configured on worker nodes	100GB
query-max-spill-per-node	Maximum spill space to use by a single query on a single node. This only needs to be configured on worker nodes	100GB
aggregation-operator-unspill-memory-limit	Limit for memory used for unspilling a single aggregation operator instance	4MB
spill-compression-codec	Compression codec to use when spilling pages to disk	NONE
spill-encryption-enabled	Set this to `true` to randomly generate a secret key (per spill file) to encrypt and decrypt data spilled to disk	false

exchange.client-threads	Number of threads used by exchange clients to fetch data from other Trino nodes. A higher value can improve performance for large clusters or clusters with very high concurrency, but excessively high values may cause a drop in performance due to context switches and additional memory usage	25
exchange.concurrent-request-multiplier	Multiplier determining the number of concurrent requests relative to available buffer memory	3
exchange.compression-codec	Compression codec to use for file compression and decompression when exchanging data between nodes and the exchange storage with fault-tolerant execution mode	LZ4
exchange.data-integrity-verification	Resulting behavior when encountering data integrity issues	ABORT
exchange.max-buffer-size	Size of buffer in the exchange client that holds data fetched from other nodes before it is processed	32MB
exchange.max-response-size	Maximum size of a response returned from an exchange request	16MB
sink.max-buffer-size	Output buffer size for task data that is waiting to be pulled by upstream tasks	32MB
sink.max-broadcast-buffer-size	Broadcast output buffer size for task data that is waiting to be pulled by upstream tasks	200MB
task.concurrency	Default local concurrency for parallel operators, such as joins and aggregations	—
task.http-response-threads	Maximum number of threads that may be created to handle HTTP responses	100
task.http-timeout-threads	Number of threads used to handle timeouts when generating HTTP responses	3
task.info-update-interval	Time interval for task information renewal, which is used in scheduling. Larger values can reduce coordinator CPU load, but may result in suboptimal split scheduling	3s
task.max-drivers-per-task	Maximum number of drivers a task can run concurrently	2147483647
task.max-partial-aggregation-memory	Maximum size of partial aggregation results for distributed aggregations	16MB
task.max-worker-threads	Number of threads used by workers to process splits	—
task.min-drivers	Target number of running leaf splits on a worker	—
task.min-drivers-per-task	Minimum number of drivers guaranteed to run concurrently for a single task given the task has remaining splits to process	3
task.min-writer-count	Minimum number of concurrent writer threads per worker per query when preferred partitioning and task writer scaling are not used	1
task.max-writer-count	Maximum number of concurrent writer threads per worker per query when preferred partitioning and task writer scaling are not used	—
task.interrupt-stuck-split-tasks-enabled	Set this to `true` to allow Trino to detect and fail tasks containing splits that have been stuck	true
task.interrupt-stuck-split-tasks-warning-threshold	Defines the splits running time threshold after which call stacks at */v1/maxActiveSplits* endpoint are printed out and JMX metrics are generated	10m
task.interrupt-stuck-split-tasks-timeout	Timeout for a blocked split processing thread before Trino fails the task	10m

task.interrupt-stuck-split-tasks-detection-interval	Interval of Trino checks for splits that have processing time exceeding the `task.interrupt-stuck-split-tasks-timeout` parameter value	2m
use-preferred-write-partitioning	Defines whether preferred write partitioning is enabled. When set to `true`, each partition is written by a separate writer	true
scale-writers	Defines whether writer scaling by dynamically increasing the number of writer tasks on the cluster is enabled	true
task.scale-writers.enabled	Defines whether scaling the number of concurrent writers within a task is enabled	true
writer-scaling-min-data-processed	Minimum amount of uncompressed data that must be processed by a writer before another writer can be added	100MB
node-scheduler.max-splits-per-node	Target value for the total number of splits that can be running on each worker node, assuming all splits have the standard split weight	256
node-scheduler.min-pending-splits-per-task	Minimum number of outstanding splits with the standard split weight guaranteed to be scheduled on a node (even when the node is already at the limit for total number of splits) for a single task given the task has remaining splits to process	16
node-scheduler.max-adjusted-pending-splits-per-task	Maximum number of outstanding splits with the standard split weight guaranteed to be scheduled on a node (even when the node is already at the limit for total number of splits) for a single task given the task has remaining splits to process	2000
node-scheduler.max-unacknowledged-splits-per-task	Maximum number of splits that are either queued on the coordinator but not yet sent, or confirmed to have been received by the worker	2000
node-scheduler.min-candidates	Minimum number of candidate nodes that are evaluated by the node scheduler when choosing the target node for a split	10
node-scheduler.policy	Node scheduler policy to use when scheduling splits	uniform
optimizer.dictionary-aggregation	Defines whether optimization for aggregations on dictionaries is used	false
optimizer.optimize-hash-generation	Set this to `true` to compute hash codes for distribution, joins, and aggregations early during execution, allowing result to be shared between operations later in the query	false
optimizer.optimize-metadata-queries	Defines whether to push the aggregation below the outer join when an aggregation is above an outer join and all columns from the outer side of the join are in the grouping clause	true
optimizer.distinct-aggregations-strategy	Strategy to use for multiple distinct aggregations	AUTOMATIC
optimizer.push-table-write-through-union	Defines whether to parallelize writes when using `UNION ALL` in queries that write data	true
optimizer.join-reordering-strategy	Join reordering strategy to use. `NONE` maintains the order the tables are listed in the query	AUTOMATIC
optimizer.max-reordered-joins	Maximum number of joins that can be reordered at once when the `optimizer.join-reordering-strategy` parameter is set to `cost-based`	8
optimizer.optimize-duplicate-insensitive-joins	Defines whether to reduce the number of rows produced by joins when optimizer detects that duplicated join output rows can be skipped	true
optimizer.use-exact-partitioning	Defines whether to re-partition the data unless the partitioning of the upstream stage exactly matches what the downstream stage expects	false
optimizer.use-table-scan-node-partitioning	Defines whether to use the connector provided table node partitioning when reading tables	true

optimizer.table-scan-node-partitioning-min-bucket-to-task-ratio	Specifies minimal bucket to task ratio that has to be matched or exceeded in order to use table scan node partitioning	0.5
optimizer.filter-conjunction-independence-factor	Scales the strength of independence assumption for estimating the selectivity of the conjunction of multiple predicates	0.75
optimizer.join-multi-clause-independence-factor	Scales the strength of independence assumption for estimating the output of a multi-clause join	0.25
optimizer.non-estimatable-predicate-approximation.enabled	Defines whether to use the cost based optimizer to determine if repartitioning the output of an already partitioned stage is necessary	true
optimizer.join-partitioned-build-min-row-count	Minimum number of join build side rows required to use partitioned join lookup	1000000
optimizer.min-input-size-per-task	Minimum input size required per task	5GB
optimizer.min-input-rows-per-task	Minimum number of input rows required per task	10000000
log.annotation-file	Name of an optional properties file that contains annotations to include with each log message for TCP output or file output in JSON format, defined with the `log.path` and `log.format` parameters	—
log.path	Path to the log file used by Trino. The path is relative to the data directory, configured to `var/log/server.log` by the launcher script	—
log.max-size	Maximum file size for the general application log file	100MB
log.max-total-size	Maximum total size for the general application log files	1GB
log.compression	Compression format for rotated log files. Can be set to either `GZIP` or `NONE`. When set to `NONE`, compression is disabled	GZIP
http-server.log.enabled	Defines whether the logging for the HTTP server is enabled	true
http-server.log.compression.enabled	Defines whether the compression of the log files of the HTTP server is enabled	true
http-server.log.path	Path to the log file of the HTTP server	/var/log/trino/http-request.log
http-server.log.max-history	Maximum number of log files for the HTTP server to use before log rotation	15
http-server.log.max-size	Maximum file size for the log file of the HTTP server	—
re2j.dfa-states-limit	Maximum number of states to use when RE2J builds the fast, but potentially memory intensive, deterministic finite automaton (DFA) for regular expression matching	2147483647
re2j.dfa-retries	Number of times that RE2J retries the DFA algorithm, when it reaches a states limit before using the slower, but less memory intensive NFA algorithm, for all future inputs for that search	5
node.internal-address-source	Set this to `FQDN` to ensure correct operation and usage of valid DNS host names by Kerberos	FQDN
access-control.config-files	Path to the access control configuration file	/etc/trino/conf/access-control.properties
internal-communication.shared-	String used by coordinators and workers of the same cluster to authenticate within it	—

secret

Kerberos Configuration

Parameter	Description	Default value
http-server.authentication.krb5.service-name	Kerberos service name for authentication	HTTP
http-server.authentication.krb5.principal-hostname	Principal hostname for Kerberos authentication	—
http-server.authentication.krb5.keytab	Path to the keytab file for Kerberos authentication	/etc/security/keytabs/HTTP.service.keytab
http.authentication.krb5.config	Path to the Kerberos configuration file	/etc/krb5.conf
http-server.authentication.krb5.user-mapping.file	File containing rules for mapping Kerberos principal names to local usernames	—

SSL Configuration

Parameter	Description	Default value
http-server.http.enabled	Defines whether HTTP is enabled for the HTTP server	true
internal-communication.https.required	Defines whether SSL/TLS is used for all internal communications	false
http-server.https.enabled	Defines whether HTTPS is enabled for the HTTP server	false
http-server.https.port	HTTPS port of the server	18189
http-server.https.keystore.path	Path to the keystore file for HTTPS	—
http-server.https.keystore.key	Password for the keystore used in HTTPS	—
http-server.https.truststore.path	Path to the truststore file for HTTPS	—
http-server.https.truststore.key	Password for the truststore used in HTTPS	—
internal-communication.https.keystore.path	Path to the keystore file for internal communication within Trino cluster via HTTPS	—
internal-communication.https.keystore.key	Password for the keystore used in internal communication within Trino cluster via HTTPS	—
internal-communication.https.truststore.path	Path to the truststore file for internal communication within Trino cluster via HTTPS	—
internal-communication.https.truststore.key	Password for the truststore used in internal communication within Trino cluster via HTTPS	—

node.properties

Parameter	Description	Default value
node.environment	Name of the environment in which the node operates. Must be the same for all Trino nodes in a cluster	adh
node.id	Unique identifier of the node	—
node.data-dir	Path to the directory for node data storage	/srv/trino/data/worker
catalog.config-dir	Path to the directory for catalog configurations	/etc/trino/conf/catalog
node.server-log-file	Path to the server log file for the node	/var/log/trino/worker/server.log
node.launcher-log-file	Path to the launcher log file for the node	/var/log/trino/worker/launcher.log

Others

Parameter	Description	Default value
env.sh	Contents of the *env.sh* file that contains Trino environment settings	env.sh
jvm.config	Contents of the *jvm.config* file that contains the Java virtual machine settings for Trino	jvm.config
Custom config.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *config.properties*	—
Custom node.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *node.properties*	—
Enable custom ulimits	Switch on the corresponding toggle button to specify resource limits (ulimits) for the current process. If you do not set these values, the default system settings are used. Ulimit settings are described in the table below	

```
[Service]
DefaultLimitCPU=
DefaultLimitFSIZE=
DefaultLimitDATA=
DefaultLimitSTACK=
DefaultLimitCORE=
DefaultLimitRSS=
DefaultLimitNOFILE=
DefaultLimitAS=
DefaultLimitNPROC=
DefaultLimitMEMLOCK=
DefaultLimitLOCKS=
DefaultLimitSIGPENDING=
DefaultLimitMSGQUEUE=
DefaultLimitNICE=
DefaultLimitRTPRIO=
DefaultLimitRTTIME=
```

Ulimit settings

Parameter	Description	Corresponding option of the ulimit command in CentOS
DefaultLimitCPU	A limit in seconds on the amount of CPU time that a process can consume	cpu time (-t)
DefaultLimitFSIZE	The maximum size of files that a process can create, in 512-byte blocks	file size (-f)
DefaultLimitDATA	The maximum size of a process's data segment, in kilobytes	data seg size (-d)

DefaultLimitSTACK	The maximum stack size allocated to a process, in kilobytes	stack size (-s)
DefaultLimitCORE	The maximum size of a core dump file allowed for a process, in 512-byte blocks	core file size (-c)
DefaultLimitRSS	The maximum of resident set size, in kilobytes	max memory size (-m)
DefaultLimitNOFILE	The maximum number of open file descriptors allowed for the process	open files (-n)
DefaultLimitAS	The maximum size of the process virtual memory (address space), in kilobytes	virtual memory (-v)
DefaultLimitNPROC	The maximum number of processes	max user processes (-u)
DefaultLimitMEMLOCK	The maximum memory size that can be locked for the process, in kilobytes. Memory locking ensures the memory is always in RAM and a swap file is not used	max locked memory (-l)
DefaultLimitLOCKS	The maximum number of files locked by a process	file locks (-x)
DefaultLimitSIGPENDING	The maximum number of signals that are pending for delivery to the calling thread	pending signals (-i)
DefaultLimitMSGQUEUE	The maximum number of bytes in POSIX message queues. POSIX message queues allow processes to exchange data in the form of messages	POSIX message queues (-q)
DefaultLimitNICE	The maximum NICE priority level that can be assigned to a process	scheduling priority (-e)
DefaultLimitRTPRIO	The maximum real-time scheduling priority level	real-time priority (-r)
DefaultLimitRTTIME	The maximum pipe buffer size, in 512-byte blocks	pipe size (-p)

The Haproxy Trino component

Parameter	Description	Default value
haproxy-trino.cfg	Jinja template with HAProxy configuration	haproxy-trino.cfg
Specifies the location of the certificate in standard .PEM format	Location of the certificate in the PEM format	/etc/ssl/serverkey.pem
/etc/syslog-ng/conf.d/haproxy-trino.conf	Syslog-ng configuration for Haproxy Trino. It is used if syslog-ng is installed in your OS	haproxy-trino.conf
/etc/rsyslog.d/haproxy-trino.conf	Rsyslog configuration for Haproxy Trino. It is used if rsyslog is installed in your OS	haproxy-trino.conf

The Trino Client component

config.properties

Parameter	Description	Default value
catalog	Default Trino catalog	—
client-info	Extra information about the client submitting a query	—
client-request-timeout	Client request timeout	2m
client-tags	A comma-separated list of tags to provide extra information about the client and the CLI user	—

editing-mode	Sets the editing mode. Possible values are `VI` and `EMACS`	EMACS
encoding	Encoding for spooling protocol	json+zstd,json+lz4,json
external-authentication-redirect-handler	Specifies a handler for external authentication redirects	ALL
extra-credential	Extra credentials in the key/value format	—
history-file	Path to a file with commands history	${user.home}/.trino_history
network-logging	The detalization level of CLI network logs. The possible values are `NONE`, `BASIC`, `HEADERS`, and `BODY`	NONE
output-format	Output format for batch mode	CSV
output-format-interactive	Output format for interactive mode	ALIGNED
pager	Path to the pager program used to display the query results	less
path	Sets the default SQL path for the session	—
schema	Default schema. Must be specified along with `catalog`	—
server	Trino server (Trino Coordinator or Trino Haproxy) location	<TrinoCoordinatorHost> or <TrinoHaproxyHost>
session	Trino session properties in the key/value format	—
source	The name of an application or source connecting to Trino	trino-cli
timezone	Sets a session time zone	UTC
user	Trino user name to connect to the Trino server	—

Kerberos Configuration

Parameter	Description	Default value
krb5-remote-service-name	Remote peer’s Kerberos service name	HTTP

SSL configuration

Parameter	Description	Default value
keystore-password	Password to the keystore file	—
keystore-path	Location of the keystore file	—
truststore-password	Password to the truststore file	—
truststore-path	Location of the truststore file	—

YARN

mapred-site.xml

Parameter	Description	Default value

mapreduce.application.classpath	The classpath for MapReduce applications. A list of files/directories to be added to the classpath. To add more items to the classpath, click + . If `mapreduce.application.framework` is set, then this parameter must specify the appropriate classpath for that archive, and the name of the archive must be present in the classpath. If `mapreduce.app-submission.cross-platform=false`, the platform-specific environment variable expansion syntax would be used to construct the default classpath entries. If `mapreduce.app-submission.cross-platform=true`, the platform-agnostic default classpath for MapReduce applications would be used: {{HADOOP_MAPRED_HOME}}/share/hadoop/mapreduce/*, {{HADOOP_MAPRED_HOME}}/share/hadoop/mapreduce/lib/* Parameter expansion marker will be replaced by NodeManager on container launch, based on the underlying OS accordingly	/etc/hadoop/conf/* /usr/lib/hadoop/* /usr/lib/hadoop/lib/* /usr/lib/hadoop-hdfs/* /usr/lib/hadoop-hdfs/lib/* /usr/lib/hadoop-yarn/* /usr/lib/hadoop-yarn/lib/* /usr/lib/hadoop-mapreduce/* /usr/lib/hadoop-mapreduce/lib/*
mapreduce.cluster.local.dir	The local directory where MapReduce stores intermediate data files. May be a comma-separated list of directories on different devices in order to spread disk I/O. Directories that do not exist, are ignored	/srv/hadoop-yarn/mr-local
mapreduce.framework.name	The runtime framework for executing MapReduce jobs. Can be one of `local`, `classic`, or `yarn`	yarn
mapreduce.jobhistory.address	MapReduce JobHistory Server IPC (<host>:<port>)	—
mapreduce.jobhistory.bind-host	Setting the value to `0.0.0.0` will cause the MapReduce daemons to listen on all addresses and interfaces of the hosts in the cluster	0.0.0.0
mapreduce.jobhistory.webapp.address	MapReduce JobHistory Server Web UI (<host>:<port>)	—
mapreduce.map.env	Environment variables for the map task processes added by a user, specified as a comma separated list. Example: `VAR1=value1,VAR2=value2`	HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
mapreduce.reduce.env	Environment variables for the reduce task processes added by a user, specified as a comma separated list. Example: `VAR1=value1,VAR2=value2`	HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
yarn.app.mapreduce.am.env	Environment variables for the MapReduce App Master processes added by a user. Examples: `A=foo`. This sets the environment variable `A` to `foo`. `B=$B:c`. This inherits the tasktracker `B` environment variable.	HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
yarn.app.mapreduce.am.staging-dir	The staging directory used while submitting jobs	/user
mapreduce.jobhistory.keytab	The location of the Kerberos keytab file for the MapReduce JobHistory Server	/etc/security/keytabs/mapreduce-historyserver.service.keytab
mapreduce.jobhistory.principal	Kerberos principal name for the MapReduce JobHistory Server	mapreduce-historyserver/_HOST@REALM
mapreduce.jobhistory.http.policy	Configures the HTTP endpoint for JobHistoryServer web UI. The following values are supported: `HTTP_ONLY` — provides service only via HTTP; `HTTPS_ONLY` — provides service only via HTTPS.	HTTP_ONLY
mapreduce.jobhistory.webapp.https.address	The HTTPS address where MapReduce JobHistory Server WebApp is running	0.0.0.0:19890
mapreduce.shuffle.ssl.enabled	Defines whether to use SSL for for the Shuffle HTTP endpoints	false

ranger-yarn-audit.xml

Parameter	Description	Default value
xasecure.audit.destination.solr.batch.filespool.dir	The spool directory path	/srv/ranger/hdfs_plugin/audit_solr_spool
xasecure.audit.destination.solr.urls	A URL of the Solr server to store audit events. Leave this property value empty or set it to `NONE` when using ZooKeeper to connect to Solr	—
xasecure.audit.destination.solr.zookeepers	Specifies the ZooKeeper connection string for the Solr destination	—
xasecure.audit.destination.solr.force.use.inmemory.jaas.config	Uses in-memory JAAS configuration file to connect to Solr	—
xasecure.audit.is.enabled	Enables Ranger audit	true
xasecure.audit.jaas.Client.loginModuleControlFlag	Specifies whether the success of the module is `required`, `requisite`, `sufficient`, or `optional`	—
xasecure.audit.jaas.Client.loginModuleName	The name of the authenticator class	—
xasecure.audit.jaas.Client.option.keyTab	The name of the keytab file to get the principal’s secret key	—
xasecure.audit.jaas.Client.option.principal	The name of the principal to be used	—
xasecure.audit.jaas.Client.option.serviceName	The name of a user or a service that wants to log in	—
xasecure.audit.jaas.Client.option.storeKey	Set this to `true` if you want the keytab or the principal’s key to be stored in the subject’s private credentials	false
xasecure.audit.jaas.Client.option.useKeyTab	Set this to `true` if you want the module to get the principal’s key from the keytab	false

ranger-yarn-security.xml

Parameter	Description	Default value
ranger.plugin.yarn.policy.rest.url	The URL to Ranger Admin	—
ranger.plugin.yarn.service.name	The name of the Ranger service containing policies for this instance	—
ranger.plugin.yarn.policy.cache.dir	The directory where Ranger policies are cached after successful retrieval from the source	/srv/ranger/yarn/policycache
ranger.plugin.yarn.policy.pollIntervalMs	Defines how often to poll for changes in policies	30000
ranger.plugin.yarn.policy.rest.client.connection.timeoutMs	The YARN Plugin RangerRestClient connection timeout (in milliseconds)	120000
ranger.plugin.yarn.policy.rest.client.read.timeoutMs	The YARN Plugin RangerRestClient read timeout (in milliseconds)	30000

ranger.add-yarn-authorization	Set `true` to use only Ranger ACLs (i.e. ignore YARN ACLs)	false
ranger.plugin.yarn.policy.rest.ssl.config.file	The path to the RangerRestClient SSL config file for the YARN plugin	/etc/yarn/conf/ranger-yarn-policymgr-ssl.xml

yarn-site.xml

Parameter	Description	Default value
yarn.application.classpath	The classpath for YARN applications. A list of files/directories to be added to the classpath. To add more items to the classpath, click +	- /etc/hadoop/conf/* - /usr/lib/hadoop/* - /usr/lib/hadoop/lib/* - /usr/lib/hadoop-hdfs/* - /usr/lib/hadoop-hdfs/lib/* - /usr/lib/hadoop-yarn/* - /usr/lib/hadoop-yarn/lib/* - /usr/lib/hadoop-mapreduce/* - /usr/lib/hadoop-mapreduce/lib/* - /usr/lib/hive/lib/*.jar
yarn.cluster.max-application-priority	Defines the maximum application priority in a cluster. Leaf Queue-level priority: each leaf queue provides default priority by the administrator. The queue default priority will be used for any application submitted without a specified priority. *$HADOOP_HOME/etc/hadoop/capacity-scheduler.xml* is the configuration file for queue-level priority	0
yarn.log.server.url	The URL for log aggregation Server	—
yarn.log-aggregation-enable	Whether to enable log aggregation. Log aggregation collects logs from each container and moves these logs onto a file system, for example HDFS, after the application processing completes. Users can configure the `yarn.nodemanager.remote-app-log-dir` and `yarn.nodemanager.remote-app-log-dir-suffix` properties to determine, where these logs are moved to. Users can access the logs via the Application Timeline Server	true
yarn.log-aggregation.retain-seconds	Defines how long to keep aggregation logs before deleting them. The value of `-1` disables logs saving. Be careful: setting this value too small will spam the NameNode	172800
yarn.nodemanager.local-dirs	The list of directories to store localized. An application localized file directory will be found in: *${yarn.nodemanager.local-dirs}/usercache/${user}/appcache/application_${appid}*. Individual containers work directories, called *container_${contid}*, will be subdirectories of this	/srv/hadoop-yarn/nm-local
yarn.node-labels.enabled	Enables node labels feature	true
yarn.node-labels.fs-store.root-dir	The URI for NodeLabelManager. The default value is `/tmp/hadoop-yarn-${user}/node-labels/` in the local filesystem	hdfs:///system/yarn/node-labels
yarn.timeline-service.bind-host	The actual address the server will bind to. If this optional address is set, the RPC and Webapp servers will bind to this address and the port, specified in `yarn.timeline-service.address` and `yarn.timeline-service.webapp.address`, respectively. This is most useful for making the service listen to all interfaces by setting to `0.0.0.0`	0.0.0.0

yarn.timeline-service.leveldb-timeline-store.path	Stores file name for leveldb Timeline store	/srv/hadoop-yarn/leveldb-timeline-store
yarn.nodemanager.address	The address of the container manager in the NodeManager	0.0.0.0:8041
yarn.nodemanager.aux-services	A comma-separated list of services, where service name should only contain `a-zA-Z0-9_` and cannot start with numbers	mapreduce_shuffle,spark_shuffle
yarn.nodemanager.aux-services.mapreduce_shuffle.class	The auxiliary service class to use	org.apache.hadoop.mapred.ShuffleHandler
yarn.nodemanager.aux-services.spark_shuffle.class	The class name of *YarnShuffleService* — an external shuffle service for Spark3 on YARN	org.apache.spark.network.yarn.YarnShuffleService
yarn.nodemanager.aux-services.spark_shuffle.classpath	The classpath for external Spark3 shuffle-service in YARN. A list of files/directories to be added to the classpath. To add more items to the classpath, click +	■ /usr/lib/spark3/yarn/lib/*
yarn.nodemanager.recovery.enabled	Enables the NodeManager to recover after starting	true
yarn.nodemanager.recovery.dir	The local filesystem directory, in which the NodeManager will store state, when recovery is enabled	/srv/hadoop-yarn/nm-recovery
yarn.nodemanager.remote-app-log-dir	Defines a directory for logs aggregation	/logs
yarn.nodemanager.resource-plugins	Enables additional discovery/isolation of resources on the NodeManager. By default, this parameters is empty. Acceptable values: `yarn.io/gpu`, `yarn.io/fpga`	—
yarn.nodemanager.resource-plugins.gpu.path-to-discovery-executables	When `yarn.nodemanager.resource.gpu.allowed-gpu-devices=auto`, YARN NodeManager needs to run GPU discovery binary (now only support nvidia-smi) to get GPU-related information. When value is empty (default), YARN NodeManager will try to locate discovery executable itself. An example of the config value is: `/usr/local/bin/nvidia-smi`	/usr/bin/nvidia-smi
yarn.nodemanager.resource.detect-hardware-capabilities	Enables auto-detection of node capabilities such as memory and CPU	true
yarn.nodemanager.vmem-check-enabled	Whether virtual memory limits will be enforced for containers	false
yarn.resource-types	The resource types to be used for scheduling. Use *resource-types.xml* to specify details about the individual resource types	—
yarn.resourcemanager.bind-host	The actual address, the server will bind to. If this optional address is set, the RPC and Webapp servers will bind to this address and the port, specified in `yarn.resourcemanager.address` and `yarn.resourcemanager.webapp.address`, respectively. This is most useful for making Resource Manager listen to all interfaces by setting to `0.0.0.0`	0.0.0.0
yarn.resourcemanager.cluster-id	The name of the cluster. In the High Availability mode, this parameter is used to ensure that Resource Manager participates in leader election for this cluster and ensures that it does not affect other clusters	—
yarn.resource-types.memory-mb.increment-allocation	The FairScheduler grants memory equal to increments of this value. If you submit a task with a resource request which is not a multiple of `memory-mb.increment-allocation`, the request will be rounded up to the nearest increment	1024
yarn.resource-types.vcores.increment-allocation	The FairScheduler grants vcores in increments of this value. If you submit a task with resource request, that is not a multiple of `vcores.increment-allocation`, the request will be rounded up to the nearest increment	1
yarn.resourcemanager.ha.enabled	Enables Resource Manager High Availability. When enabled: ■ The Resource Manager starts in the Standby mode by default, and transitions to the Active mode when prompted to.	false

	- The nodes in the Resource Manager ensemble are listed in `yarn.resourcemanager.ha.rm-ids`. - The id of each Resource Manager either comes from `yarn.resourcemanager.ha.id`, if `yarn.resourcemanager.ha.id is explicitly` specified, or can be figured out by matching `yarn.resourcemanager.address.{id}` with local address. - The actual physical addresses come from the configs of the pattern `{rpc-config}.{id}`.	
yarn.resourcemanager.ha.rm-ids	The list of Resource Manager nodes in the cluster when the High Availability is enabled. See description of `yarn.resourcemanager.ha.enabled` for full details on how this is used	—
yarn.resourcemanager.hostname	The host name of the Resource Manager	—
yarn.resourcemanager.leveldb-state-store.path	The Local path, where the Resource Manager state will be stored, when using `org.apache.hadoop.yarn.server.resourcemanager.recovery.LeveldbRMStateStore` as the value for `yarn.resourcemanager.store.class`	/srv/hadoop-yarn/leveldb-state-store
yarn.resourcemanager.monitor.capacity.queue-management.monitoring-interval	The time between invocations of this QueueManagementDynamicEditPolicy policy (in milliseconds)	1500
yarn.resourcemanager.reservation-system.enable	Enables the ReservationSystem in the ResourceManager	false
yarn.resourcemanager.reservation-system.planfollower.time-step	The frequency of the PlanFollower timer (in milliseconds). A large value is expected	1000
Resource scheduler	The type of a pluggable scheduler for Hadoop. Available values: `CapacityScheduler` and `FairScheduler`. `CapacityScheduler` allows for multiple-tenants to securely share a large cluster such that their applications are allocated resources in a timely manner under constraints of allocated capacities. `FairScheduler` allows YARN applications to share resources in large clusters fairly	CapacityScheduler
yarn.resourcemanager.scheduler.monitor.enable	Enables a set of periodic monitors (specified in `yarn.resourcemanager.scheduler.monitor.policies`) that affect the Scheduler	false
yarn.resourcemanager.scheduler.monitor.policies	The list of SchedulingEditPolicy classes that interact with the Scheduler. A particular module may be incompatible with the Scheduler, other policies, or a configuration of either	org.apache.hadoop.yarn.server.resourcemanager.monitor.capacity.ProportionalCapacityPreemptionPolicy
yarn.resourcemanager.monitor.capacity.preemption.observe_only	If set to `true`, run the policy but do not affect the cluster with preemption and kill events	false
yarn.resourcemanager.monitor.capacity.preemption.monitoring_interval	The time between invocations of this ProportionalCapacityPreemptionPolicy policy (in milliseconds)	3000
yarn.resourcemanager.monitor.capacity.preemption.max_wait_before_kill	The time between requesting a preemption from an application and killing the container (in milliseconds)	15000
yarn.resourcemanager.monitor.capacity.preemption.total_preemption_per_round	The maximum percentage of resources, preempted in a single round. By controlling this value one can throttle the pace, at which containers are reclaimed from the cluster. After computing the total desired preemption, the policy scales it back within this limit	0.1
yarn.resourcemanager.monitor.capacity.preemption.max_ignored_over_capacity	The maximum amount of resources above the target capacity ignored for preemption. This defines a deadzone around the target capacity, that helps to prevent thrashing and oscillations around the computed target balance. High values would slow the time to capacity and (absent natural.completions) it might prevent convergence to guaranteed capacity	0.1

yarn.resourcemanager.monitor.capacity.preemption.natural_termination_factor	Given a computed preemption target, account for containers naturally expiring and preempt only this percentage of the delta. This determines the rate of geometric convergence into the deadzone (`MAX_IGNORED_OVER_CAPACITY`). For example, a termination factor of 0.5 will reclaim almost 95% of resources within 5 * `#WAIT_TIME_BEFORE_KILL`, even absent natural termination	0.2
yarn.resourcemanager.nodes.exclude-path	The path to the file with nodes to exclude	/etc/hadoop/conf/exclude-path.xml
yarn.resourcemanager.nodes.include-path	The path to the file with nodes to include	/etc/hadoop/conf/include-path
yarn.resourcemanager.recovery.enabled	Enables Resource Manager to recover state after starting. If set to `true`, then `yarn.resourcemanager.store.class` must be specified	true
yarn.resourcemanager.store.class	The class to use as the persistent store. If `org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore` is used, the store is implicitly fenced; meaning a single Resource Manager is able to use the store at any point in time. More details on this implicit fencing, along with setting up appropriate ACLs is discussed under `yarn.resourcemanager.zk-state-store.root-node.acl`	—
yarn.resourcemanager.system-metrics-publisher.enabled	The setting that controls whether YARN system metrics are published on the Timeline Server or not by Resource Manager	true
yarn.scheduler.fair.user-as-default-queue	Defines whether to use the username, associated with the allocation as the default queue name, in the event, that a queue name is not specified. If this is set to `false` or unset, all jobs have a shared default queue, named `default`. Defaults to `true`. If a queue placement policy is given in the allocations file, this property is ignored	true
yarn.scheduler.fair.preemption	Defines whether to use preemption	false
yarn.scheduler.fair.preemption.cluster-utilization-threshold	The utilization threshold after which the preemption kicks in. The utilization is computed as the maximum ratio of usage to capacity among all resources	0.8f
yarn.scheduler.fair.sizebasedweight	Defines whether to assign shares to individual apps based on their size, rather than providing an equal share to all apps regardless of size. When set to `true`, apps are weighted by the natural logarithm of one plus the app total requested memory, divided by the natural logarithm of 2	false
yarn.scheduler.fair.assignmultiple	Defines whether to allow multiple container assignments in one heartbeat	false
yarn.scheduler.fair.dynamic.max.assign	If `assignmultiple` is `true`, this parameter specifies whether to dynamically determine the amount of resources that can be assigned in one heartbeat. When turned on, about half of the non-allocated resources on the node are allocated to containers in a single heartbeat	true
yarn.scheduler.fair.max.assign	If `assignmultiple` is `true`, the maximum amount of containers that can be assigned in one heartbeat. Defaults to `-1`, which sets no limit	-1
yarn.scheduler.fair.locality.threshold.node	For applications that request containers on particular nodes, this parameter defines the number of scheduling opportunities since the last container assignment to wait before accepting a placement on another node. Expressed as a floating number between `0` and `1`, which, as a fraction of the cluster size, is the number of scheduling opportunities to pass up. The default value of `-1.0` means not to pass up any scheduling opportunities	-1.0
yarn.scheduler.fair.locality.threshold.rack	For applications, that request containers on particular racks, the number of scheduling opportunities since the last container assignment to wait before accepting a placement on another rack. Expressed as a floating point between `0` and `1`, which, as a fraction of the cluster size, is the number of scheduling opportunities to pass up. The default value of `-1.0` means not to pass up any scheduling opportunities	-1.0
yarn.scheduler.fair.allow-undeclared-pools	If set to `true`, new queues can be created at application submission time, whether because they are specified as the application queue by the submitter or because they are placed there by the `user-as-default-queue` property. If set to `false`, any time an app would be placed in a queue that is not specified in the allocations file, it is placed in the `default` queue instead. Defaults to `true`. If a queue placement policy is given in the allocations file, this property is ignored	true

yarn.scheduler.fair.update-interval-ms	The time interval, at which to lock the scheduler and recalculate fair shares, recalculate demand, and check whether anything is due for preemption	500
yarn.scheduler.minimum-allocation-mb	The minimum allocation for every container request at the Resource Manager (in MB). Memory requests, lower than this, will throw `InvalidResourceRequestException`	1024
yarn.scheduler.maximum-allocation-mb	The maximum allocation for every container request at the Resource Manager (in MB). Memory requests, higher than this, will throw `InvalidResourceRequestException`	4096
yarn.scheduler.minimum-allocation-vcores	The minimum allocation for every container request at the Resource Manager, in terms of virtual CPU cores. Requests, lower than this, will throw `InvalidResourceRequestException`	1
yarn.scheduler.maximum-allocation-vcores	The maximum allocation for every container request at the Resource Manager, in terms of virtual CPU cores. Requests, higher than this, will throw `InvalidResourceRequestException`	2
yarn.timeline-service.enabled	On the server side this parameter indicates, whether Timeline service is enabled or not. And on the client side, this parameter can be used to indicate whether client wants to use Timeline service. If this parameter is set on the client side along with security, then YARN Client tries to fetch the delegation tokens for the Timeline Server	true
yarn.timeline-service.hostname	The hostname of the Timeline service Web application	—
yarn.timeline-service.http-cross-origin.enabled	Enables cross-origin support (CORS) for Timeline Server	true
yarn.webapp.ui2.enable	In the Server side it indicates, whether the new YARN UI v2 is enabled or not	true
yarn.resourcemanager.proxy-user-privileges.enabled	If set to `true`, ResourceManager will have proxy-user privileges. For example: in a secure cluster, YARN requires the user hdfs delegation-tokens to do localization and log-aggregation on behalf of the user. If this is set to `true`, ResourceManager is able to request new hdfs delegation tokens on behalf of the user. This is needed by long-running-services, because the hdfs tokens will eventually expire and YARN requires new valid tokens to do localization and log-aggregation. Note that to enable this use case, the corresponding HDFS NameNode must have ResourceManager configured as a proxy-user so that ResourceManager can itself ask for new tokens on behalf of the user when tokens are past their max-life-time	false
yarn.resourcemanager.webapp.spnego-principal	The Kerberos principal to be used for SPNEGO filter for the Resource Manager web UI	HTTP/_HOST@REALM
yarn.resourcemanager.webapp.spnego-keytab-file	The Kerberos keytab file to be used for SPNEGO filter for the Resource Manager web UI	/etc/security/keytabs/HTTP.service.keytab
yarn.nodemanager.linux-container-executor.group	The UNIX group that the linux-container-executor should run as	yarn
yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled	A flag to enable override of the default Kerberos authentication filter with the RM authentication filter to allow authentication using delegation tokens (fallback to Kerberos if the tokens are missing). Only applicable when the http authentication type is `kerberos`	false
yarn.resourcemanager.principal	The Kerberos principal for the Resource Manager	yarn-resourcemanager/_HOST@REALM
yarn.resourcemanager.keytab	The keytab for the Resource Manager	/etc/security/keytabs/yarn-resourcemanager.service.keytab
yarn.resourcemanager.webapp.https.address	The https address of the Resource Manager web application. If only a host is provided as the value, the webapp will be served on a random port	${yarn.resourcemanager.hostname}:8090
yarn.nodemanager.principal	The Kerberos principal for the NodeManager	yarn-nodemanager/_HOST@REALM
yarn.nodemanager.keytab	Keytab for NodeManager	/etc/security/keytabs/yarn-

		nodemanager.service.keytab
yarn.nodemanager.webapp.spnego-principal	The Kerberos principal to be used for SPNEGO filter for the NodeManager web interface	HTTP/_HOST@REALM
yarn.nodemanager.webapp.spnego-keytab-file	The Kerberos keytab file to be used for SPNEGO filter for the NodeManager web interface	/etc/security/keytabs/HTTP.service.keytab
yarn.nodemanager.webapp.https.address	The HTTPS address of the NodeManager web application	0.0.0.0:8044
yarn.timeline-service.http-authentication.type	Defines the authentication used for the Timeline Server HTTP endpoint. Supported values are: `simple`, `kerberos`, `#AUTHENTICATION_HANDLER_CLASSNAME#`	simple
yarn.timeline-service.http-authentication.simple.anonymous.allowed	Indicates if anonymous requests are allowed by the Timeline Server when using `simple` authentication	true
yarn.timeline-service.http-authentication.kerberos.keytab	The Kerberos keytab to be used for the Timeline Server (Collector/Reader) HTTP endpoint	/etc/security/keytabs/HTTP.service.keytab
yarn.timeline-service.http-authentication.kerberos.principal	The Kerberos principal to be used for the Timeline Server (Collector/Reader) HTTP endpoint	HTTP/_HOST@REALM
yarn.timeline-service.principal	The Kerberos principal for the timeline reader. NodeManager principal would be used for timeline collector as it runs as an auxiliary service inside NodeManager	yarn/_HOST@REALM
yarn.timeline-service.keytab	The Kerberos keytab for the timeline reader. NodeManager keytab would be used for timeline collector as it runs as an auxiliary service inside NodeManager	/etc/security/keytabs/yarn.service.keytab
yarn.timeline-service.delegation.key.update-interval	The update interval for delegation keys	86400000
yarn.timeline-service.delegation.token.renew-interval	The time to renew delegation tokens	86400000
yarn.timeline-service.delegation.token.max-lifetime	The maxim token lifetime	86400000
yarn.timeline-service.client.best-effort	Defines, whether a failure to obtain a delegation token should be considered as an application failure (`false`), or the client should attempt to continue to publish information without it (`true`)	false
yarn.timeline-service.webapp.https.address	The HTTPS address of the Timeline service web application	${yarn.timeline-service.hostname}:8190
yarn.http.policy	This configures the HTTP endpoint for Yarn Daemons. The following values are supported: ▪ `HTTP_ONLY` — provides service only via HTTP; ▪ `HTTPS_ONLY` — provides service only via HTTPS.	HTTP_ONLY
yarn.nodemanager.container-executor.class	The name of the container-executor Java class	org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor

container-executor.cfg

CAUTION

In AstraLinux, regular user UIDs can start from 100. For YARN to work correctly on AstraLinux, set the `min.user.id` parameter value to 100.

Parameter	Description	Default value
banned.users	A comma-separated list of users who cannot run applications	bin
min.user.id	Prevents other super-users	500

Enable CORS

Parameter	Description	Default value
yarn.nodemanager.webapp.cross-origin.enabled	Enables cross-origin support for NodeManager web-services	true
yarn.resourcemanager.webapp.cross-origin.enabled	Enables cross-origin support for ResourceManager web-services	true
yarn_site.enable_cors.active	Enables CORS (Cross-Origin Resource Sharing)	true

yarn-env.sh

Parameter	Description	Default value
YARN_RESOURCEMANAGER_OPTS	YARN ResourceManager heap memory. Sets initial (-Xms) and maximum (-Xmx) Java heap size for ResourceManager	-Xms1G -Xmx8G
YARN_NODEMANAGER_OPTS	YARN NodeManager heap memory. Sets initial (-Xms) and maximum (-Xmx) Java heap size for NodeManager	—
YARN_TIMELINESERVER_OPTS	YARN Timeline Server heap memory. Sets initial (-Xms) and maximum (-Xmx) Java heap size for Timeline Server	-Xms700m -Xmx8G

Lists of decommissioned hosts

Parameter	Description	Default value
DECOMMISSIONED	The list of hosts in the `DECOMMISSIONED` state	—

ranger-yarn-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	The path to the keystore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.credential.file	The path to the keystore credentials file	/etc/yarn/conf/ranger-yarn.jceks

xasecure.policymgr.clientssl.truststore.credential.file	The path to the truststore credentials file	/etc/yarn/conf/ranger-yarn.jceks
xasecure.policymgr.clientssl.truststore	The path to the truststore file used by Ranger	—
xasecure.policymgr.clientssl.keystore.password	The password to the keystore file	—
xasecure.policymgr.clientssl.truststore.password	The password to the truststore file	—

mapred-env.sh

Parameter	Description	Default value
HADOOP_JOB_HISTORYSERVER_OPTS	MapReduce History Server heap memory. Sets initial (-Xms) and maximum (-Xmx) Java heap size for MapReduce History Server	-Xms700m -Xmx8G

Other

Parameter	Description	Default value
GPU on YARN	Defines, whether to use GPU on YARN	false
capacity-scheduler.xml	The content of *capacity-scheduler.xml*, which is used by CapacityScheduler	Default capacity-scheduler.xml
fair-scheduler.xml	The content of *fair-scheduler.xml*, which is used by FairScheduler	Default fair-scheduler.xml
Custom mapred-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *mapred-site.xml*	—
Ranger plugin enabled	Whether or not Ranger plugin is enabled	false
Custom yarn-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *yarn-site.xml*	—
Custom ranger-yarn-audit.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-yarn-audit.xml*	—
Custom ranger-yarn-security.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-yarn-security.xml*	—
Custom ranger-yarn-policymgr-ssl.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *ranger-yarn-policymgr-ssl.xml*	—
Custom mapred-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *mapred-env.sh*	—
Custom yarn-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *yarn-env.sh*	—

container-executor.cfg template	The template for the *container-executor.cfg* configuration file	—

YARN NodeManager component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display YARN NodeManager metrics in the Prometheus format	9205
Mapping config path	Path to the metrics mapping configuration file	/etc/hadoop/conf/jmx_yarn_nodemanager_metric_config.yml
Mapping config	Metrics mapping configuration file	yarn-mapping-config.yml

YARN ResourceManager component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display YARN ResourceManager metrics in the Prometheus format	9204
Mapping config path	Path to the metrics mapping configuration file	/etc/hadoop/conf/jmx_yarn_resourcemanager_metric_config.yml
Mapping config	Metrics mapping configuration file	yarn-mapping-config.yml

YARN Timeline Server component

Monitoring

Parameter	Description	Default value
Java agent path	Path to the JMX Prometheus Java agent	/usr/lib/adh-utils/jmx/jmx_prometheus_javaagent.jar
Prometheus metrics port	Port on which to display YARN Timeline Server metrics in the Prometheus format	9206
Mapping config path	Path to the metrics mapping configuration file	/etc/hadoop/conf/jmx_yarn_timelineserver_metric_config.yml
Mapping config	Metrics mapping configuration file	yarn-mapping-config.yml

Zeppelin

User-managed interpreters

Parameter	Description	Default value
Allow user-managed interpreters	Allows you to use Zeppelin interpreters with the `user-managed=true` property. If enabled, ADCM will preserve custom user properties when restarting Zeppelin	true
Custom interpreter.json	Enables the interpreter configuration defined in the ***interpreter.json*** ADCM field	false
interpreter.json	The custom JSON definition of interpreters to be available in the Zeppelin web UI. Defining interpreters in this way overwrites all (both user and system) interpreters settings	interpreters.json
interpreter.sh	The custom contents of the *interpreter.sh* script. This script is invoked on the Zeppelin startup and is used to prepare the environment for proper Zeppelin operation	interpreters.sh

zeppelin-site.xml

Parameter	Description	Default value
zeppelin.dep.localrepo	The local repository for the dependency loader	/srv/zeppelin/local-repo
zeppelin.server.addr	The Zeppelin server binding address	0.0.0.0
zeppelin.server.port	The server port	8180
zeppelin.server.kerberos.principal	The principal name to load from the keytab	—
zeppelin.server.kerberos.keytab	The path to the keytab file	—
zeppelin.shell.auth.type	Sets the authentication type. Possible values are `SIMPLE` and `KERBEROS`	—
zeppelin.shell.principal	The principal name to load from the keytab	—
zeppelin.shell.keytab.location	The path to the keytab file	—
zeppelin.jdbc.auth.type	Sets the authentication type. Possible values are `SIMPLE` and `KERBEROS`	—
zeppelin.jdbc.keytab.location	The path to the keytab file	—
zeppelin.jdbc.principal	The principal name to load from the keytab	—
zeppelin.jdbc.auth.kerberos.proxy.enable	When the `KERBEROS` authentication type is used, this parameter enables/disables proxy with the login user to get the connection	true
spark.yarn.keytab	The full path to the file that contains the keytab for the principal. This keytab will be copied to the node running the YARN Application Master via the Secure Distributed Cache, for renewing the login tickets and the delegation tokens periodically	—
spark.yarn.principal	The principal to be used to login to KDC, while running on secure HDFS	—
zeppelin.livy.keytab	The path to the keytab file	—
zeppelin.livy.principal	The principal name to load from the keytab	—
zeppelin.server.ssl.port	The port number for SSL communication	8180

zeppelin.ssl	Defines whether to use SSL	false
zeppelin.ssl.keystore.path	The path to the keystore used by Zeppelin	—
zeppelin.ssl.keystore.password	The password to access the keystore file	—
zeppelin.ssl.truststore.path	The path to the truststore used by Zeppelin	—
zeppelin.ssl.truststore.password	The password to access the truststore file	—

Zeppelin server heap memory settings

Parameter	**Description**	**Default value**
Zeppelin Server Heap Memory	Sets initial (-Xms) and maximum (-Xmx) Java heap size for Zeppelin Server	-Xms700m -Xmx1024m

Shiro Simple username/password auth

Parameter	**Description**	**Default value**
Users/password map	A map of type <username: password>. For example, `<myUser1: password1>`	—
sessionManager	A class that handles the creation, maintenance, and cleanup of all application sessions	org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager	The SessionManager component implementation for SecurityManager	$sessionManager
securityManager.sessionManager.globalSessionTimeout	The timeout in milliseconds after which a session gets expired and the user is required to log in again	86400000
shiro.loginUrl	Sets the `loginUrl` property for any Shiro’s default filters	/api/login
cookie	The class responsible for cookie handling	org.apache.shiro.web.servlet.SimpleCookie
cookie.name	The name of the session cookie used by Shiro. This parameter must be the same as the `responseExcludeHeaders` parameter in */usr/lib/knox/data/services/zeppelinui/{version}/service.xml*	JSESSIONID
cookie.httpOnly	Defines whether Shiro’s cookies should have the `HttpOnly` attribute	true
cookie.secure	Defines whether Shiro’s cookies should have the `Secure` attribute	false
sessionManager.sessionIdCookie	The cookie implementation for sessionManager	$cookie

Shiro LDAP auth

Parameter	**Description**	**Default value**

ldapRealm	Extends the Apache Shiro provider to allow for LDAP searches and to provide group membership to the authorization provider	org.apache.zeppelin.realm.LdapRealm
ldapRealm.contextFactory.authenticationMechanism	Specifies the authentication mechanism used by the LDAP service	simple
ldapRealm.contextFactory.url	The URL of the source LDAP. For example, *ldap://ldap.example.com:389*	—
ldapRealm.userDnTemplate	Optional. Knox uses this value to construct the UserDN for the authentication bind. Specify the UserDN where the first attribute is `{0}` indicating the attribute which matches the user log in token. For example, the UserDnTemplate for Apache DS bundled with Knox is `uid={0},ou=people,dc=hadoop,dc=apache,dc=org`	—
ldapRealm.pagingSize	Sets the LDAP paging size	100
ldapRealm.authorizationEnabled	Enables authorization for Shiro ldapRealm	true
ldapRealm.contextFactory.systemAuthenticationMechanism	Defines the authentication mechanism to use for Shiro ldapRealm context factory. Possible values are `simple` and `digest-md+5`	—
ldapRealm.userLowerCase	Forces username returned from LDAP to be lower-cased	true
ldapRealm.memberAttributeValueTemplate	The attribute that identifies a user in the group. For exmaple: `cn={0},ou=people,dc=hadoop,dc=apache,dc=org`	—
ldapRealm.searchBase	The starting DN in the LDAP DIT for the search. Only subtrees of the specified subtree are searched. For example: `dc=hadoop,dc=apache,dc=org`	—
ldapRealm.userSearchBase	Search base for user bind DN. Defaults to the value of `ldapRealm.searchBase` if no value is defined. If `ldapRealm.userSearchAttributeName` is defined, also define a value for either `ldapRealm.searchBase` or `ldapRealm.userSearchBase`	—
ldapRealm.groupSearchBase	Search base used to search for groups. Defaults to the value of `ldapRealm.searchBase`. Only set if `ldapRealm.authorizationEnabled=true`	—
ldapRealm.groupObjectClass	Set the value to the Objectclass that identifies group entries in LDAP	groupofnames
ldapRealm.userSearchAttributeName	Specify the attribute that corresponds to the user login token. This attribute is used with the search results to compute the UserDN for the authentication bind	sAMAccountName
ldapRealm.memberAttribute	Set the value to the attribute that defines group membership. When the value is `rememberer`, found groups are treated as dynamic groups	member
ldapRealm.userSearchScope	Defines searchScope. Possible values are `subtree`, `one`, `base`	subtree
ldapRealm.groupSearchScope	Defines groupSearchScope. Possible values are `subtree`, `one`, `base`	subtree
ldapRealm.contextFactory.systemUsername	Set to the LDAP Service Account that the Zeppelin uses for LDAP searches. If required, specify the full account UserDN. For example: `uid=guest,ou=people,dc=hadoop,dc=apache,dc=org`. This account requires read permission to the search base DN	—
ldapRealm.contextFactory.systemPassword	Sets the password for systemUsername. This password will be added to the keystore using `hadoop` credentials	—

ldapRealm.groupSearchEnableMatchingRuleInChain	Enables support for nested groups using the LDAP_MATCHING_RULE_IN_CHAIN operator	true
ldapRealm.rolesByGroup	Optional mapping from physical groups to logical application roles. For example: `"LDN_USERS":"user_role", "NYK_USERS":"user_role", "HKG_USERS":"user_role", "GLOBAL_ADMIN":"admin_role"`	—
ldapRealm.allowedRolesForAuthentication	Optional list of roles that are allowed to authenticate. If not specified, all groups are allowed to authenticate (login). This changes nothing for url-specific permissions that will continue to work as specified in [urls]. For example: `admin_role,user_role`	—
ldapRealm.permissionsByRole	Optional. Sets permissions by role. For example: `'user_role = :ToDoItemsJdo::*, :ToDoItem::*; admin_role = *'`	—
securityManager.realms	Specifies a list of Apache Shiro Realms	$ldapRealm
sessionManager	A class that handles the creation, maintenance, and cleanup of all application sessions	org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager	The SessionManager component implementation for SecurityManager	$sessionManager
securityManager.sessionManager.globalSessionTimeout	The timeout in milliseconds after which a session gets expired and the user is required to log in again	86400000
shiro.loginUrl	Sets the `loginUrl` property for any Shiro's default filters	/api/login
cookie	The class responsible for cookie handling	org.apache.shiro.web.servlet.SimpleCookie
cookie.name	The name of the session cookie used by Shiro. This parameter must be the same as the `responseExcludeHeaders` parameter in */usr/lib/knox/data/services/zeppelinui/{version}/service.xml*	JSESSIONID
cookie.httpOnly	Defines whether Shiro's cookies should have the `HttpOnly` attribute	true
cookie.secure	Defines whether Shiro's cookies should have the `Secure` attribute	false
sessionManager.sessionIdCookie	The cookie implementation for sessionManager	$cookie

Shiro Active Directory auth

Parameter	Description	Default value
activeDirectoryRealm	The Shiro realm to work with Active Directory	org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
activeDirectoryRealm.systemUsername	The user name for connecting to Active Directory	—
activeDirectoryRealm.systemPassword	The user password for connecting to Active Directory	—
activeDirectoryRealm.searchBase	The base DN of your Active Directory server. For example: `CN=Users,DC=SOME_GROUP,DC=COMPANY,DC=COM`	—
activeDirectoryRealm.url	The URL of your Active Directory server. For example: *ldap://ldap.example.com:389*	—

activeDirectoryRealm.groupRolesMap	A mapping of AD groups to Apache Shiro roles. For example: `'CN=aGroupName,OU=groups,DC=SOME_GROUP,DC=COMPANY,DC=COM':'group1', 'CN=bGroupName,OU=groups,DC=SOME_GROUP,DC=COMPANY,DC=COM':'group2'`	—
activeDirectoryRealm.authorizationCachingEnabled	Enables/disables caching of authorization decisions	false
securityManager.realms	The realm used by securityManager	$activeDirectoryRealm
sessionManager	A class that handles the creation, maintenance, and cleanup of all application sessions	org.apache.shiro.web.session.mgt.DefaultWebSessionManager
securityManager.sessionManager	The SessionManager component implementation for SecurityManager	$sessionManager
securityManager.sessionManager.globalSessionTimeout	The timeout in milliseconds after which a session gets expired and the user is required to log in again	86400000
shiro.loginUrl	Sets the `loginUrl` property for any Shiro's default filters	/api/login
cookie	The class responsible for cookie handling	org.apache.shiro.web.servlet.SimpleCookie
cookie.name	The name of the session cookie used by Shiro. This parameter must be the same as the `responseExcludeHeaders` parameter in */usr/lib/knox/data/services/zeppelinui/{version}/service.xml*	JSESSIONID
cookie.httpOnly	Defines whether Shiro's cookies should have the `HttpOnly` attribute	true
cookie.secure	Defines whether Shiro's cookies should have the `Secure` attribute	false
sessionManager.sessionIdCookie	The cookie implementation for sessionManager	$cookie

Shiro SSO Knox

Parameter	**Description**	**Default value**
knoxJwtRealm	The Shiro realm to work with Knox	org.apache.zeppelin.realm.ActiveDirectoryGroupRealm
knoxJwtRealm.providerUrl	The URL of your Knox server. For example: *https://<knox-host>:8443/*	—
knoxJwtRealm.login	The URL to perform the login action	gateway/knoxssout/api/v1/webssout
knoxJwtRealm.logout	The URL to perform the logout action	gateway/knoxssout/api/v1/webssout
knoxJwtRealm.logoutAPI	Enables the logout API action	true
knoxJwtRealm.redirectParam	A URL to redirect the client after the logout	originalUrl
knoxJwtRealm.cookieName	The name of the cookie used by Knox SSO	hadoop-jwt
knoxJwtRealm.publicKeyPath	A path to the public key (certificate) used by Knox	/usr/lib/zeppelin/gateway.cer
sessionManager	A class that handles the creation, maintenance, and cleanup of all application sessions	org.apache.shiro.web.session.mgt.DefaultWebSessionManager

securityManager.sessionManager	The SessionManager component implementation for SecurityManager	$sessionManager
securityManager.sessionManager.globalSessionTimeout	The timeout in milliseconds after which a session gets expired and the user is required to log in again	86400000
shiro.loginUrl	Sets the `loginUrl` property for any Shiro's default filters	/api/login
cookie	The class responsible for cookie handling	org.apache.shiro.web.servlet.SimpleCookie
cookie.httpOnly	Defines whether Shiro's cookies should have the `HttpOnly` attribute	true
cookie.secure	Defines whether Shiro's cookies should have the `Secure` attribute	false
sessionManager.sessionIdCookie	The cookie implementation for sessionManager	$cookie
cookie.name	The name of the session cookie used by Shiro. This parameter must be the same as the `responseExcludeHeaders` parameter in */usr/lib/knox/data/services/zeppelinui/{version}/service.xml*	WWW-AUTHENTICATE
knoxJwtRealm.groupPrincipalMapping	Defines principal and group mapping rules	group.principal.mapping
knoxJwtRealm.principalMapping	Defines principal mapping rules	principal.mapping

Additional configuration Shiro.ini

Parameter	Description	Default value
Additional main section in shiro.ini	Allows adding additional key/value pairs to the `main` section of the *shiro.ini* file	—
Additional roles section in shiro.ini	Allows adding additional key/value pairs to the `roles` section of the *shiro.ini* file	—
Additional urls section in shiro.ini	Allows adding additional key/value pairs to the `urls` section of the *shiro.ini* file	—

Credential Encryption

Parameter	Description	Default value
Encryption enable	Enables or disables the credential encryption feature. When enabled, Zeppelin stores configuration passwords and credentials required for interacting with other services in the encrypted form	false
Credential provider path	The path to a keystore file with secrets	jceks://file/etc/zeppelin/conf/zeppelin.jceks
Custom jceks	Set to `true` to use a custom JCEKS file. Set to `false` to use the default auto-generated JCEKS file	false

Other

Parameter	Description	Default value

Custom zeppelin-site.xml	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *zeppelin-site.xml*	—
Custom zeppelin-env.sh	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *zeppelin-env.sh*	zeppelin-env.sh
Custom log4j.properties	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *log4j.properties*	log4j.properties

ZooKeeper

Main

Parameter	Description	Default value
connect	The ZooKeeper connection string used by other services or clusters. It is generated automatically	—
dataDir	The location where ZooKeeper stores the in-memory database snapshots and, unless specified otherwise, the transaction log of updates to the database	/var/lib/zookeeper

zoo.cfg

Parameter	Description	Default value
clientPort	The port to listen for client connections, that is the port that clients attempt to connect to	2181
admin.serverPort	Embedded Jetty server port	5181
admin.enableServer	Enables the AdminServer, which is an embedded Jetty server that provides an HTTP interface to the four-letter-word commands. Set to `false` to use the system property	false
tickTime	The basic time unit used by ZooKeeper (in milliseconds). It is used for heartbeats. The minimum session timeout will be twice the `tickTime`	2000
initLimit	The timeouts that ZooKeeper uses to limit the length of the time for ZooKeeper servers in quorum to connect to the leader	5
syncLimit	Defines the maximum date skew between server and the leader	2
maxClientCnxns	This property limits the number of active connections from the host, specified by IP address, to a single ZooKeeper Server	0
autopurge.snapRetainCount	When enabled, ZooKeeper auto-purge feature retains the `autopurge.snapRetainCount` most recent snapshots and the corresponding transaction logs in the `dataDir` and `dataLogDir` respectively and deletes the rest. The minimum value is `3`	3
autopurge.purgeInterval	The time interval, for which the purge task has to be triggered (in hours). Set to a positive integer (`1` and above) to enable the auto-purging	24
Add key,value	In this section you can define values for custom parameters that are not displayed in ADCM UI, but are allowed in the configuration file *zoo.cfg*	—

SSl configuration

Parameter	Description	Default value
sslQuorum	Enables encrypted quorum communication	false
serverCnxnFactory	Specifies `ServerCnxnFactory` implementation. This should be set to `NettyServerCnxnFactory` in order to use TLS-based server communication	org.apache.zookeeper.server.NettyServerCnxnFactory
ssl.quorum.keyStore.location	Specifies the absolute path to the server keystore file (for example: */etc/zookeeper/ssl/keystore.jks*)	—
ssl.quorum.keyStore.password	Specifies the password used when the keystore was created	—
ssl.quorum.trustStore.location	Specifies the absolute path to the server truststore file (for example: */etc/zookeeper/ssl/truststore.jks*)	—
ssl.quorum.trustStore.password	Specifies the password used when the truststore is created	—
ssl.protocol	Specifies the protocol to be used in client TLS negotiation	TLSv1.2
ssl.quorum.protocol	Specifies the protocol to be used in quorum TLS negotiation	TLSv1.2

Parameter	Description	Default value
Myid matching	A mapping of the quorum hosts to internal ZooKeeper IDs	—

zookeeper-env.sh

Parameter	Description	Default value
ZOO_LOG_DIR	The directory to store logs	/var/log/zookeeper
ZOOPIDFILE	The directory to store the ZooKeeper process ID	/var/run/zookeeper/zookeeper_server.pid
SERVER_JVMFLAGS	Used for setting different JVM parameters connected, for example, with garbage collecting	-Xmx1024m
JAVA	A path to Java	$JAVA_HOME/bin/java
logback.xml template	Used for setting the logging level and defines which log appenders to turn on. Enabling the log appender `CONSOLE` directs logs to stdout. Enabling `ROLLINGFILE` creates the *zookeeper.log* file, then this file gets rotated, and expired	—

Monitoring

Prometheus settings

Group	Parameter	Description	Default value
—	scrape_interval	Specifies how frequently to scrape targets	1m
—	scrape_timeout	Specifies how long to wait until a scrape request times	10s

		out	
—	Password for grafana connection	Password of a Grafana user (`admprom_grafana`) to connect to Prometheus	—
—	Prometheus users to login/logout to Prometheus	User credentials for logging into the Prometheus web interface	—
Service parameters	config.file	Path to the Prometheus server configuration file	/etc/admprom/prometheus/prometheus.yml
	storage.tsdb.path	Path to the Prometheus server database	/var/lib/admprom/prometheus
	web.console.libraries	Path to console management libraries	/usr/share/admprom/prometheus/console_libraries
	web.console.templates	Path to Prometheus server console templates	/usr/share/admprom/prometheus/consoles
	web.config.file	Specifies which web configuration file to load. The file is written in YAML format	/etc/admprom/prometheus/prometheus-auth.yml
	storage.tsdb.retention.time	Specifies how long to retain samples in the storage. Supported units: `y`, `w`, `d`, `h`, `m`, `s`, `ms`	15d
	web.listen-address	Address to access the Prometheus web interface	0.0.0.0:11200

Grafana settings

Parameter	Description	Default value
Grafana administrator’s password	Password of a Grafana administrator user	—
Grafana listen port	Port to access the Grafana web interface	11210

Node Exporter settings

Parameter	Description	Default value
Listen port	Port to access ADH host system metrics in the Prometheus format	11203
Metrics endpoint	Endpoint to which the Node Exporter exports system metrics in the Prometheus format	/metrics

SSL configuration

Parameter	Description	Default value
[Prometheus] → Enable SSL	Defines whether SSL is enabled for Prometheus	false

[Prometheus] → Certificate file	Path to the Prometheus server SSL certificate file in the PEM format	/etc/admprom/prometheus/server.crt
[Prometheus] → Private key file	Path to the private key file of the Prometheus server SSL certificate	/etc/admprom/prometheus/server.key
[Prometheus] → Certificate authority file	Path to the certificate authority file	/etc/admprom/prometheus/ca.crt
[Grafana] → Enable SSL	Defines whether SSL is enabled for Grafana	false
[Grafana] → Certificate file	Path to the Grafana server SSL certificate file in the PEM format	/etc/admprom/grafana/server.crt
[Grafana] → Private key file	Path to the private key file of the Grafana server SSL certificate	/etc/admprom/grafana/server.key
[Grafana] → Certificate authority file	Path to the certificate authority file	/etc/admprom/grafana/ca.crt
[Node-exporter] → Enable SSL	Defines whether SSL is enabled for Node Exporter	false
[Node-exporter] → Certificate file	Path to the Node Exporter server SSL certificate file in the PEM format	/etc/ssl/server.crt
[Node-exporter] → Private key file	Path to the private key file of the Node Exporter server SSL certificate	/etc/ssl/server.key
Set SSL rights for certs/key	Enables changing the owner and permissions to the SSL certificate and key files	false

Scrape config for HDFS NameNode (hdfs_namenode_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	hdfs-namenode
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for HDFS DataNode (hdfs_datanode_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	hdfs-datanode
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for HDFS JournalNode (hdfs_journalnode_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	hdfs-journalnode
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Hue (hue-scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	hue_exporter
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Impala Daemon (impala_daemon_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	impala-daemon
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Impala Statestore (impala_statestore_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	impala-statestore
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Impala Catalog (impala_catalog_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	impala-catalog
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Hive Metastore (hive_metastore_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	hive-metastore
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Hive Server (hive_server_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	hive-server
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Ozone (ozone_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	ozone_exporter
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for Kyuubi (kyuubi_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	kyuubi_exporter
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for YARN nodemanager (yarn_nodemanager_scraper.yml)

Parameter	Description	Default value

job_name	The name of the job within which metrics will be collected	yarn-nodemanager
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for YARN resourcemanager (yarn_resourcemanager_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	yarn-resourcemanager
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

Scrape config for YARN timelineserver (yarn_timelineserver_scraper.yml)

Parameter	Description	Default value
job_name	The name of the job within which metrics will be collected	yarn-timelineserver
scrape_interval	Specifies how frequently to scrape targets	—
scrape_timeout	Specifies how long to wait until a scrape request times out. Cannot be greater than the value of the `scrape_interval` parameter	—

ADH monitoring metrics

Konstantin Alpashkin

Contents

To Table of Contents

- Metric types
- Metrics visualization
  - Graphite
  - Grafana

This article describes the monitoring metrics reported by ADH to a Monitoring cluster.

Metric types

All the metrics are divided into two groups: system metrics and service metrics. Both metric types are available in Graphite/Grafana UIs after the installation of the Monitoring cluster.

System metrics indicate general characteristics of cluster hosts, usually related to resource consumption. The reported system metrics are presented below.

System metrics

Metrics group	Description
cpu	CPU utilization
diskspace	Disk usage
files	File statistics
iostat	I/O performance
loadavg	System load averages
memory	Memory usage
netstat	Network connection statistics
network	Network interface performance
uptime	Uptime statistics

Service metrics reflect the characteristics of individual ADH services and their list is below.

HBase

Component	Metrics group	Metric name	Description
RegionServer	IPC	numActiveHandler	The number of RPC handlers that serve incoming requests
		numCallsInGeneralQueue	The number of user requests in the queue
		numOpenConnections	The number of open connections (RPC)
	JvmMetrics	GcCount	The total number of GC iterations
		GcCountConcurrentMarkSweep	The number of GC iterations done with the Concurrent Mark Sweep algorithm
		GcCountParNew	The number of GC iterations done with the ParNew algorithm
		GcTimeMillis	The total GC time in milliseconds
		GcTimeMillisConcurrentMarkSweep	The total GC time (CMS) in milliseconds
		GcTimeMillisParNew	The total GC time (ParNew) in milliseconds
		MemHeapCommittedM	The currently allocated heap memory in MB
		MemHeapMaxM	The maximum heap memory in MB
		MemHeapUsedM	The currently utilized heap memory in MB
	Server	Append_95th_percentile	The 95 percentile latency for the append operation in the RegionServer
		Append_median	The median latency for the append operation in the RegionServer
		Delete_95th_percentile	The 95 percentile latency for the delete operation in the RegionServer
		Delete_median	The median latency for the delete operation in the RegionServer
		Get_95th_percentile	The 95 percentile latency for the get operation in the RegionServer
		Get_median	The median latency for the get operation in the RegionServer
		Increment_95th_percentile	The 95 percentile latency for the increment operation in the RegionServer
		Increment_median	The median latency for the increment operation in the RegionServer
		Put_95th_percentile	The 95 percentile latency for the put operation in the RegionServer
		Put_median	The median latency for the put operation in the RegionServer
		ScanTime_95th_percentile	The 95 percentile latency for the scan operation in the RegionServer
		ScanTime_median	The median latency for the scan operation in the RegionServer
		percentFilesLocal	The percent of store file data that can be read from the local DataNode

HDFS

Component	Metrics group	Metric name	Description
datanode	FSDatasetState	NumFailedVolumes	The number of failed disks in a cluster
	JvmMetrics	GcCount	The total number of GC iterations
		GcCountPSMarkSweep	The number of GC iterations done with the PS MarkSweep algorithm
		GcCountPSScavenge	The number of GC iterations done with the PS Scavenge algorithm
		GcTimeMillis	The total GC time in milliseconds
		GcTimeMillisPSMarkSweep	The total GC time (PS MarkSweep) in milliseconds
		GcTimeMillisPSScavenge	The total GC time (PS Scavenge) in milliseconds
		MemHeapCommittedM	The currently allocated heap memory in MB
		MemHeapMaxM	The maximum heap memory in MB
		MemHeapUsedM	The currently utilized heap memory in MB
	RpcActivityForPort	NumOpenConnections	The number of open RPC connections
		RpcProcessingTimeAvgTime	The average request processing time
		RpcQueueTimeAvgTime	The average time of pending requests in a queue
namenode	FSNamesystem	BlockCapacity	The total space available for storing data blocks
		BlocksTotal	The total number of data blocks
		CapacityTotal	The total space available for storing data
		CapacityUsed	The total capacity used for storing data
		CorruptReplicatedBlocks	The number of blocks that are corrupted or have been copied repeatedly to HDFS
		FilesTotal	The total number of files in HDFS
		MissingBlocks	The number of missing data blocks
		UnderReplicatedBlocks	The number of data blocks that do not have sufficient replicas in HDFS
	JvmMetrics	GcCount	The total number of GC iterations
		GcCountPSMarkSweep	The number of GC iterations done with the PS MarkSweep algorithm
		GcCountPSScavenge	The number of GC iterations done with the PS Scavenge algorithm
		GcTimeMillis	The total GC time in milliseconds
		GcTimeMillisPSMarkSweep	The total GC time (PS MarkSweep) in milliseconds
		GcTimeMillisPSScavenge	The total GC time (PS Scavenge) in milliseconds
		MemHeapCommittedM	The currently allocated heap memory in MB
		MemHeapMaxM	The maximum heap memory in MB
		MemHeapUsedM	The currently utilized heap memory in MB
	RpcActivityForPort	NumOpenConnections	The number of open RPC connections
		RpcProcessingTimeAvgTime	The average request processing time
		RpcQueueTimeAvgTime	The average time of pending requests in a queue

Hive

Component	Metrics group	Metric name	Description
hiveserver2	ClassLoading	LoadedClassCount	The number of loaded classes
	GarbageCollector	PSMarkSweep	The GC time spent using the PS MarkSweep algorithm
		PSScavenge	The GC time spent using the PS Scavenge algorithm
	HS2	active_calls_api_Driver_execute	The `active_calls_api_<method_name>` metrics indicate the number of corresponding method invocations at the given time
		active_calls_api_Driver_run	
		active_calls_api_PostHook_org_apache_hadoop_hive_ql_stats_OperatorStatsReaderHook	
		active_calls_api_compile	
		active_calls_api_hs2_operation_INITIALIZED	The number of active operations in HiveServer2 (HS2) with the corresponding status
		active_calls_api_hs2_operation_PENDING	
		active_calls_api_hs2_operation_RUNNING	
		active_calls_api_hs2_sql_operation_PENDING	The number of active SQL operations with the corresponding status
		active_calls_api_hs2_sql_operation_RUNNING	
		active_calls_api_parse	The number of requests submitted to the HiveServer via the API, which were successfully parsed and ready for processing
		active_calls_api_releaseLocks	The `active_calls_api_<method_name>` metrics indicate the number of corresponding method invocations at the given time
		active_calls_api_runTasks	
		active_calls_api_semanticAnalyze	
		active_calls_api_waitCompile	
		active_calls_hs2_compiling_queries	The number of requests that are currently compiling
		active_calls_hs2_executing_queries	The number of requests that are currently executing
		active_calls_hs2_submitted_queries	The number of requests submitted for execution
		api_Driver_execute	The `api_<method_name>` metrics indicate the number of the corresponding method invocations
		api_Driver_run	
		api_PostHook_org_apache_hadoop_hive_ql_stats_OperatorStatsReaderHook	
		api_compile	
		api_hs2_operation_INITIALIZED	
		api_hs2_operation_PENDING	
		api_hs2_operation_RUNNING	
		api_hs2_sql_operation_PENDING	
		api_hs2_sql_operation_RUNNING	
		api_parse	
		api_releaseLocks	
		api_runTasks	
		api_semanticAnalyze	
		api_waitCompile	
		cumulative_connection_count	The total number of established connections to HiveServer2 since the server startup
		exec_async_pool_size	The current size of the HiveServer2 asynchronous thread pool
		exec_async_queue_size	The current size of the HiveServer2 asynchronous operation queue
		hs2_active_sessions	The number of active sessions on HiveServer2
		hs2_compiling_queries	The number of queries being compiled on HiveServer2
		hs2_completed_operation_CLOSED	The number of completed operations with the corresponding status
		hs2_completed_operation_FINISHED	
		hs2_completed_sql_operation_CLOSED	The number of completed SQL operations with the corresponding status
		hs2_completed_sql_operation_FINISHED	
		hs2_executing_queries	The number of queries being executed on HiveServer2
		hs2_open_sessions	The number of open sessions on HiveServer2
		hs2_sql_operation_active_user	The current number of active users performing SQL operations on HiveServer2
		hs2_submitted_queries	The number of queries submitted to HiveServer2
		hs2_succeeded_queries	The number of queries succeeded on HiveServer2
		buffers_direct_count	JVM metrics
		buffers_direct_used	
		buffers_mapped_capacity	
		buffers_mapped_count	
		buffers_mapped_used	
		classLoading_loaded	
		classLoading_unloaded	
		gc_PS-MarkSweep_count	
		gc_PS-MarkSweep_time	
		gc_PS-Scavenge_count	
		gc_PS-Scavenge_time	
		jvm_pause_extraSleepTime	
		memory_heap_committed	
		memory_heap_init	
		memory_heap_max	
		memory_heap_usage	
		memory_heap_used	
		memory_non-heap_committed	
		memory_non-heap_init	
		memory_non-heap_max	
		memory_non-heap_usage	
		memory_non-heap_used	
		memory_pools_Code-Cache_usage	
		memory_pools_Compressed-Class-Space_usage	
		memory_pools_Metaspace_usage	
		memory_pools_PS-Eden-Space_usage	
		memory_pools_PS-Old-Gen_usage	
		memory_pools_PS-Survivor-Space_usage	
		memory_total_committed	
		memory_total_init	
		memory_total_max	
		memory_total_used	
		open_connections	
		open_operations	
		qc_current_size	
		qc_max_size	
		threads_blocked_count	
		threads_count	
		threads_daemon_count	
		threads_deadlock_count	
		threads_new_count	
		threads_runnable_count	
		threads_terminated_count	
		threads_timed_waiting_count	
		threads_waiting_count	
		waiting_compile_ops	
	Memory	HeapMemoryUsage_committed	The amount of memory allocated for heap
		HeapMemoryUsage_init	The initial heap memory size
		HeapMemoryUsage_max	The maximum heap memory size
		HeapMemoryUsage_used	The utilized heap memory size
		NonHeapMemoryUsage_committed	The amount of memory allocated for non-heap JVM areas
		NonHeapMemoryUsage_init	The initial memory size allocated for non-heap JVM areas
		NonHeapMemoryUsage_max	The maximum memory size allocated for non-heap JVM areas
		NonHeapMemoryUsage_used	The utilized memory for non-heap JVM areas
	MemoryPool	CodeCache	The code cache size
		CompressedClassSpace	The size of the compressed class space
		Metaspace	The metaspace size
		PSEdenSpace	The size of the Eden space in the Spark History Server using the PS algorithm
		PSOldGen	The size of the Old Generation space in the Spark History Server using the PS algorithm
		PSSurvivorSpace	The size of the Survivor space in the Spark History Server using the PS algorithm
	OperatingSystem	ProcessCpuLoad	The CPU load
	Threading	DaemonThreadCount	The number of daemon threads
		PeakThreadCount	The maximum number of threads
		ThreadCount	The current number of threads
		TotalStartedThreadCount	The total threads count
metastore	ClassLoading	LoadedClassCount	The number of loaded classes
	GarbageCollector	PSMarkSweep	The GC time spent using the PS MarkSweep algorithm
		PSScavenge	The GC time spent using the PS Scavenge algorithm
	Memory	HeapMemoryUsage_committed	The amount of memory allocated for heap
		HeapMemoryUsage_init	The initial heap memory size
		HeapMemoryUsage_max	The maximum heap memory size
		HeapMemoryUsage_used	The utilized heap memory size
		NonHeapMemoryUsage_committed	The amount of memory allocated for non-heap JVM areas
		NonHeapMemoryUsage_init	The initial memory size allocated for non-heap JVM areas
		NonHeapMemoryUsage_max	The maximum memory size allocated for non-heap JVM areas
		NonHeapMemoryUsage_used	The utilized memory for non-heap JVM areas
	MemoryPool	CodeCache	The code cache size
		CompressedClassSpace	The size of the compressed class space
		Metaspace	The metaspace size
		PSEdenSpace	The size of the Eden space in the Spark History Server using the PS algorithm
		PSOldGen	The size of the Old Generation space in the Spark History Server using the PS algorithm
		PSSurvivorSpace	The size of the Survivor space in the Spark History Server using the PS algorithm
	OperatingSystem	ProcessCpuLoad	The CPU load
	Threading	DaemonThreadCount	The number of daemon threads
		PeakThreadCount	The maximum number of threads
		ThreadCount	The current number of threads
		TotalStartedThreadCount	The total threads count
	metastore	PS-MarkSweep_count	The number of GC iterations done using the Concurrent Mark Sweep algorithm
		PS-MarkSweep_time	The total GC time elapsed using the PS MarkSweep algorithm
		PS-Scavenge_count	The number of GC iterations done using the PS Scavenge algorithm
		PS-Scavenge_time	The total GC time elapsed using the PS Scavenge algorithm
		active_calls_create_table	The `active_calls_api_<method_name>` metrics indicate the number of coreesponding method invocation at the given time
		active_calls_drop_table	
		active_calls_get_all_functions	
		active_calls_get_config_value	
		active_calls_get_database	
		active_calls_get_databases	
		active_calls_get_functions	
		active_calls_get_multi_table	
		active_calls_get_table	
		active_calls_get_tables	
		active_calls_get_tables_by_type	
		api_create_table	The `api_<method_name>` metrics indicate the number of corresponding method invocations
		api_create_table_with_environment_context	
		api_drop_table	
		api_drop_table_with_environment_context	
		api_flushCache	
		api_get_all_databases	
		api_get_all_functions	
		api_get_config_value	
		api_get_current_notificationEventId	
		api_get_database	
		api_get_databases	
		api_get_functions	
		api_get_multi_table	
		api_get_next_notification	
		api_get_table	
		api_get_table_objects_by_name_req	
		api_get_table_req	
		api_get_tables	
		api_get_tables_by_type	
		api_init	
		api_set_ugi	
		api_shutdown	
		blocked_count	The number of threads blocked
		create_total_count_dbs	The number of databases created
		create_total_count_partitions	The number of partitions created
		create_total_count_tables	The number of tables created
		daemon_count	The daemon count
		deadlock_count	The number of deadlocks detected
		delete_total_count_dbs	The total number of deleted databases
		delete_total_count_partitions	The total number of deleted partitions
		delete_total_count_tables	The total number of deleted tables
		direct_count	JVM metrics
		direct_used	
		directsql_errors	
		heap_committed	
		heap_init	
		heap_max	
		heap_usage	
		heap_used	
		jvm_pause_extraSleepTime	
		jvm_pause_info-threshold	
		jvm_pause_warn-threshold	
		loaded	
		mapped_capacity	
		mapped_count	
		mapped_used	
		new_count	
		non-heap_committed	
		non-heap_init	
		non-heap_max	
		non-heap_usage	
		non-heap_used	
		open_connections	
		pools_Code-Cache_usage	
		pools_Compressed-Class-Space_usage	
		pools_Metaspace_usage	
		pools_PS-Eden-Space_usage	
		pools_PS-Old-Gen_usage	
		pools_PS-Survivor-Space_usage	
		runnable_count	
		terminated_count	
		timed_waiting_count	
		total_committed	
		total_count_dbs	
		total_count_partitions	
		total_count_tables	
		total_init	
		total_max	
		total_used	
		unloaded	
		waiting_count	

Spark/Spark3

Component	Metrics group	Metric name	Description
historyserver	ClassLoading	LoadedClassCount	The number of loaded classes
	GarbageCollector	PSMarkSweep	The GC time spent using the PS MarkSweep algorithm
		PSScavenge	The GC time spent using the PS Scavenge algorithm
	Memory	HeapMemoryUsage_committed	The amount of memory allocated for heap
		HeapMemoryUsage_init	The initial heap memory size
		HeapMemoryUsage_max	The maximum heap memory size
		HeapMemoryUsage_used	The utilized heap memory size
		NonHeapMemoryUsage_committed	The amount of memory allocated for non-heap JVM areas
		NonHeapMemoryUsage_init	The initial memory size allocated for non-heap JVM areas
		NonHeapMemoryUsage_max	The maximum memory size allocated for non-heap JVM areas
		NonHeapMemoryUsage_used	The utilized memory for non-heap JVM areas
	MemoryPool	CodeCache	The code cache size
		CompressedClassSpace	The size of the compressed class space
		Metaspace	The metaspace size
		PSEdenSpace	The size of the Eden space in the Spark History Server using the PS algorithm
		PSOldGen	The size of the Old Generation space in the Spark History Server using the PS algorithm
		PSSurvivorSpace	The size of the Survivor space in the Spark History Server using the PS algorithm
	OperatingSystem	ProcessCpuLoad	The CPU load
	Threading	DaemonThreadCount	The number of daemon threads
		PeakThreadCount	The maximum number of threads
		ThreadCount	The current number of threads
		TotalStartedThreadCount	The total threads count

YARN

Component	Metrics group	Metric name	Description
historyserver	JvmMetrics	GcCount	The total number of GC iterations
		GcCountPSMarkSweep	The number of GC iterations done with the PS MarkSweep algorithm
		GcCountPSScavenge	The number of GC iterations done with the PS Scavenge algorithm
		GcTimeMillis	The total GC time in milliseconds
		GcTimeMillisPSMarkSweep	The total GC time (PS MarkSweep) in milliseconds
		GcTimeMillisPSScavenge	The total GC time (PS Scavenge) in milliseconds
		MemHeapCommittedM	The currently allocated heap memory in MB
		MemHeapMaxM	The maximum heap memory in MB
		MemHeapUsedM	The currently utilized heap memory in MB
nodemanager	JvmMetrics	GcCount	The total number of GC iterations
		GcCountPSMarkSweep	The number of GC iterations done with the PS MarkSweep algorithm
		GcCountPSScavenge	The number of GC iterations done with the PS Scavenge algorithm
		GcTimeMillis	The total GC time in milliseconds
		GcTimeMillisPSMarkSweep	The total GC time (PS MarkSweep) in milliseconds
		GcTimeMillisPSScavenge	The total GC time (PS Scavenge) in milliseconds
		MemHeapCommittedM	The currently allocated heap memory in MB
		MemHeapMaxM	The maximum heap memory in MB
		MemHeapUsedM	The currently utilized heap memory in MB
	NodeManagerMetrics	AllocatedContainers	The number of allocated containers
		AllocatedGB	The size of allocated memory in GB
		AllocatedVCores	The number of allocated cores
		AvailableGB	The size of available memory in GB
		AvailableVCores	The number of available cores
		BadLocalDirs	The number of directories available on the local disk that can be used for storing task data due to errors
		BadLogDirs	The number of directories on the local disk that can not be used for storing task log files due to errors
		ContainerLaunchDurationAvgTime	The average time spent on launching a task container
		ContainersCompleted	The number of task containers that were completed successfully
		ContainersFailed	The number of task containers that failed to complete
		ContainersIniting	The number of task containers in the initialization state
		ContainersKilled	The number of task containers that were forcibly stopped
		ContainersLaunched	The number of task containers that were started successfully
		ContainersRunning	The number of task containers that are currently running
		GoodLocalDirsDiskUtilizationPerc	The percentage of disk space utilization in directories on a local disk that can be used for storing task data
		GoodLogDirsDiskUtilizationPerc	The percentage of disk space utilization in directories on a local disk that can be used for storing task log files
resourcemanager	JvmMetrics	GcCount	The total number of GC iterations
		GcCountPSMarkSweep	The number of GC iterations done with the PS MarkSweep algorithm
		GcCountPSScavenge	The number of GC iterations done with the PS Scavenge algorithm
		GcTimeMillis	The total GC time in milliseconds
		GcTimeMillisPSMarkSweep	The total GC time (PS MarkSweep) in milliseconds
		GcTimeMillisPSScavenge	The total GC time (PS Scavenge) in milliseconds
		MemHeapCommittedM	The currently allocated heap memory in MB
		MemHeapMaxM	The maximum heap memory in MB
		MemHeapUsedM	The currently utilized heap memory in MB
	QueueMetrics	AllocatedVCores	The number of allocated cores
		AppsFailed	The number of applications exited with an error
		AppsKilled	The number of applications killed by a user
		AppsPending	The number of applications pending for resources
		AppsRunning	The number of started applications
		AppsSubmitted	The number of applications submitted to a queue
		AvailableVCores	The number of available cores

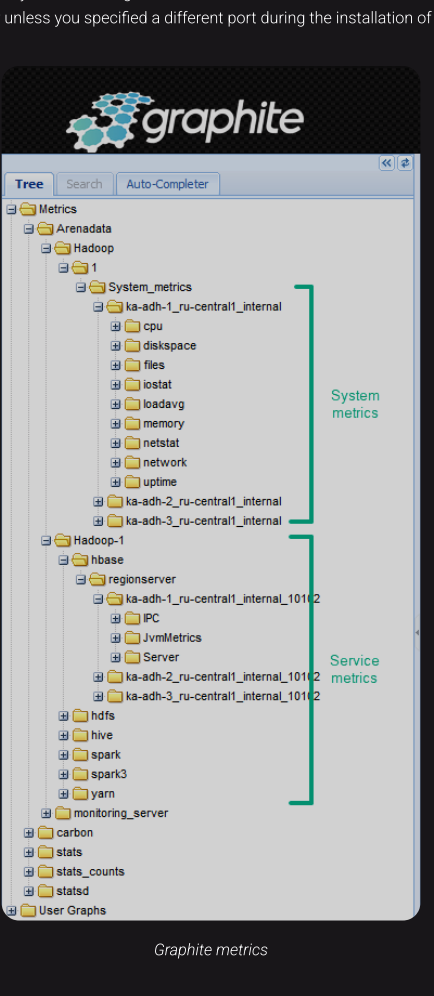
Metrics visualization

To visualize monitoring metrics as graphs and charts, use the Graphite and Grafana UIs available after the installation of a monitoring cluster.

Graphite

Graphite is a monitoring tool that stores numeric time-series data and visualizes this data on graphs in web UI.

To view metrics in Graphite, enter the address of the host, where your monitoring cluster is installed, into the browser address bar. For example, *http://10.20.30.444:<port>*. By default, Graphite runs on the 80 port, so you can omit the port number unless you specified a different port during the installation of your monitoring cluster.



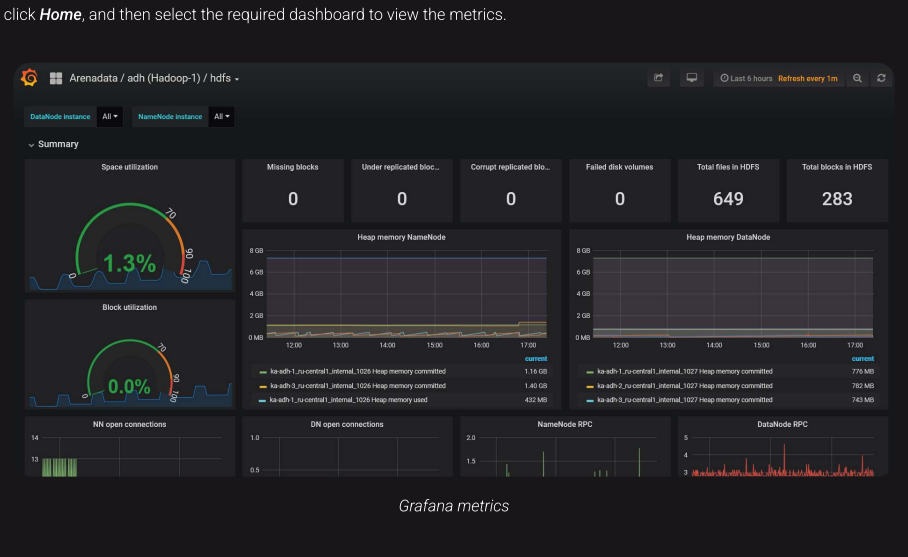
Graphite metrics

Grafana

Grafana allows you to query data and visualize metrics stored in Graphite.

To view the Grafana web UI, enter the address of the host, where your monitoring cluster is installed, into the browser address bar. For example, *http://10.20.30.444:<port>*. By default, Grafana UI is available on the 3000 port.

On the Grafana home page, click ***Home***, and then select the required dashboard to view the metrics.



Grafana metrics

Planning guide

Albert Bagdasaryan

To Table of Contents

ADH is a software framework that supports distributed data analysis on standard servers. Arenadata recommends the following design principles that ensure reliable deployment using various hardware. It is very important to predict accurately the volume, type, and number of tasks for ADH clusters. Also, there are metrics for measuring actual workloads during a pilot project. You can easily scale the pilot environment without making significant changes to existing hardware, software, deployment strategies, and network connectivity.

In this guide you will find:

- Software requirements
- Hardware requirements
- Filesystem requirements
- ADH network ports
- Typical configurations

Software requirements

Albert Bagdasaryan

Contents

To Table of Contents

Requirements table

Install Java from the Arenadata repository

User permissions

The full list of software requirements is in the table below. Beyond that, ADH requires the user who installs it to have full sudo privileges.

Requirements table

Component	Requirements
Platform	Intel x86_64
OS	■ Centos 7.9. ■ RHEL 7.9. ■ AltLinux 8.4 SP (for the Enterprise edition of ADH, starting with the version 2.1.6.b1). ■ AltLinux 10 SP (for the Enterprise edition of ADH, starting with the version 4.1.0.b1). ■ Astra Linux 1.7 "Orel" SE (for the Enterprise edition of ADH, starting with the version 3.2.4). The `LC_TIME` locale should be set to `en_US.utf8`. ■ Astra Linux 1.7 "Voronezh" (for the Enterprise edition of ADH, starting with the version 4.1.0.b1). ■ Ubuntu 22.04.2 LTS. ■ RED OS 7.3 Certified edition (for the Enterprise edition of ADH).
Browser	■ Internet Explorer ■ Firefox ■ Google Chrome ■ Safari
Software	■ RPM + YUM / DPKG + APT ■ scp ■ curl ■ unzip ■ tar
Java	OpenJDK 8u252 or later for all services except Trino (Java 23), Ozone (Java 21), Flink2 (Java 17), and Spark4 (Java 17)
Arenadata Cluster Manager (ADCM)	See the version compatibility matrix
Antivirus	It is recommended to switch off any antivirus software before ADH installation and during the ADH operation process. In case of using antivirus, please contact your antivirus software vendor to find out the settings necessary to allow ADH

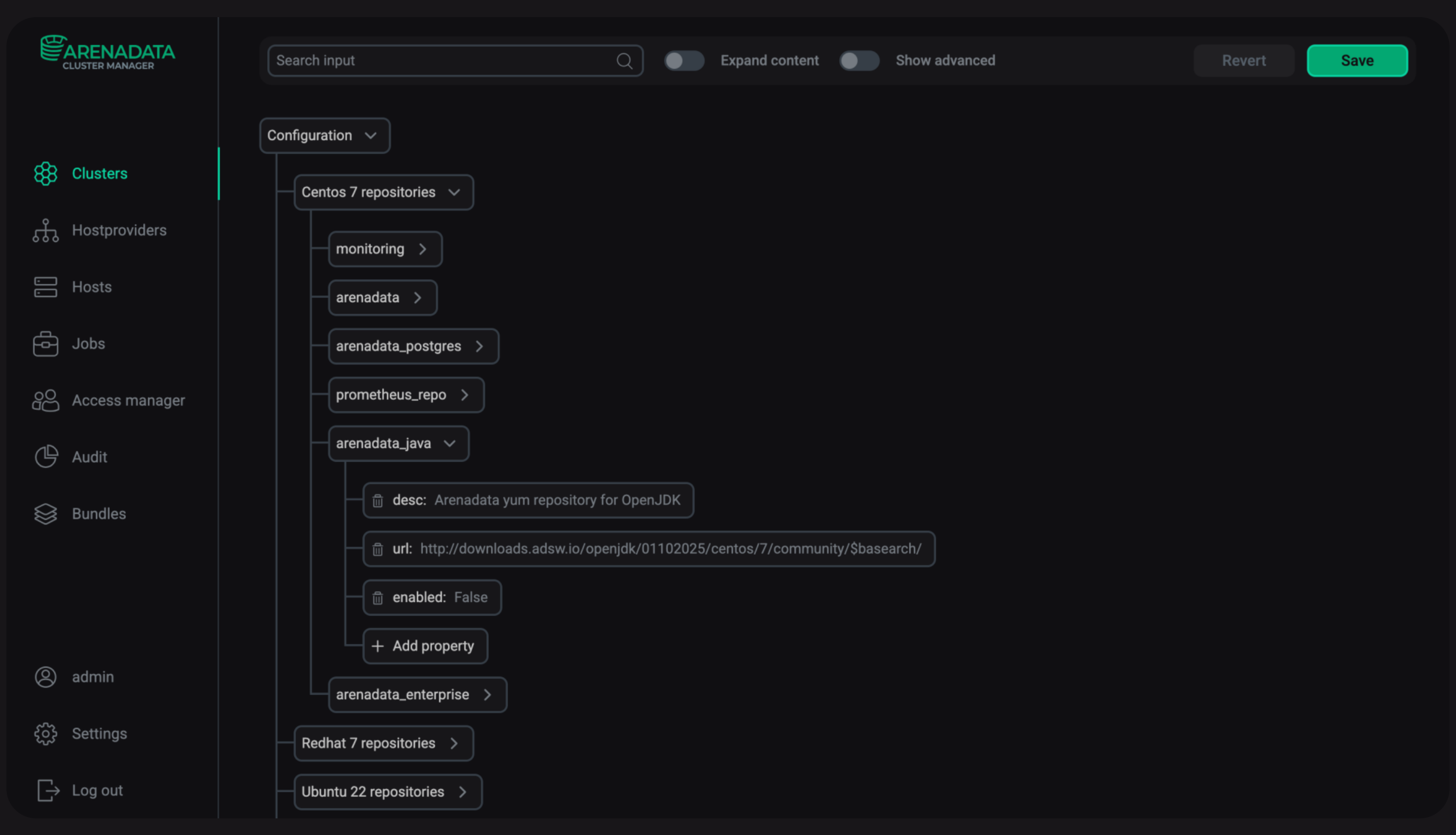
Install Java from the Arenadata repository

When installing or updating the distribution, all required Java versions are installed on all hosts under the */usr/lib/jvm/* path. This is determined by the ***ad-runtime-utils*** parameter and controlled by the cluster/services settings. You can still use the Java version supplied with the operating system or a custom installation.

Starting with ADH 4.0.0, you can opt to install Java on cluster hosts from the Arenadata repository. The corresponding option is available on the cluster configuration page and, if enabled, installs the up-to-date JDK version, required to run certain ADH services. The option must be activated before running the ***Install***, ***Expand***, or ***Upgrade*** cluster actions.

To install Java from the Arenadata repository:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Configuration*** and expand the group with the repositories of the selected OS (for example, ***Centos 7 repositories***).
3. Select the ***arenadata_java*** repository and expand its parameters.
4. Set the ***enabled*** parameter to `true`.
5. Click ***Apply*** and confirm changes to cluster configuration by clicking ***Save***.
6. Run the cluster action you need (***Install***, ***Expand***, or ***Upgrade***).



desc: Arenadata yum repository for OpenJDK

url: http://downloads.adsw.io/openjdk/01102025/centos/7/community/$basearch/

enabled: False

Configuring Java installation from Arenadata repository for Centos 7

Once this option is enabled, Java will be installed from the Arenadata repository regardless of what is specified in the JAVA_HOME variable (***Custom JDK settings*** group).

Starting with ADH 4.1.0, you can select which Java version to install on hosts via cluster configuration using the ***ad-runtime-utils*** parameter. It contains a configuration file that specifies what Java versions will be used for the whole cluster and which versions will be installed for specific ADH services. As per requirements, Java 8 is installed for the whole cluster, Java 23 for Trino, Java 21 for Ozone, and Java 17 for Flink2.

To change Java versions or installation paths:

1. On the ***Clusters*** page, select the desired cluster.
2. Navigate to ***Configuration*** and find the ***ad-runtime-utils*** option.
3. Edit the configuration file as necessary.
4. Click ***Apply*** and confirm changes to cluster configuration by clicking ***Save***.
5. Run the cluster action you need (***Install***, ***Expand***, or ***Upgrade***).

> **CAUTION**
>
> Arenadata is not an official JDK provider, and this feature is provided to ensure a fully automatic installation on systems that do not have the necessary dependencies. In production environments, we recommend using a JDK from official providers.

User permissions

To grant the user the necessary permissions, do the following:

1. Grant the `root` privileges for the user:

   CentOS 7/RHEL 7/RED OS 7.3/Alt Linux SP 8

   ```
   $ sudo usermod -aG wheel <user>
   ```

   where `<user>` is a username.

   Ubuntu 22.04/Astra Linux SE 1.7 Orel

   ```
   $ sudo usermod -aG sudo <user>
   ```

   where `<user>` is a username.

2. Make sure that the */etc/sudoers* file contains the record specified below depending on the operating system with the following command:

```
$ sudo vi /etc/sudoers
```

   To return to the command line, press `Esc`, type `:q!`, and press `Enter`.

CentOS 7/RHEL 7/RED OS 7.3

```
## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL
```

Ubuntu 22.04/Astra Linux SE 1.7 Orel

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

Alt Linux SP 8

```
## Uncomment to allow members of group wheel to execute any command
WHEEL_USERS ALL=(ALL:ALL) ALL
```

Hardware requirements

Albert Bagdasaryan

Contents

To Table of Contents

Power supply and consumption

Network

Hardware planning cannot be full without clear understanding of the workload. When you are planning your ADH cluster, it is very important to accurately predict the volume, type, and number of tasks. You can use a lot of metrics to get actual workloads during a pilot project. In this way, you can easily scale the pilot environment without making significant changes to existing infrastructure.

The number of machines and their specifications depend on a few factors:

- the total volume of data;
- the data retention policy (by default the replication index is 3);
- the type of workload you have (see Hardware requirements depending on workload patterns);
- the data storage mechanism (data container, type of compression used if any).

There are different recommendations for a pilot cluster.

Power supply and consumption

Power is a major concern when designing ADH clusters. Before purchasing the biggest and fastest nodes, analyze the power consumption of your existing hardware. We have observed huge savings in pricing and power by avoiding fastest CPUs, redundant power supplies, and other hardware.

Vendors today are building machines for cloud data centers that are designed to reduce cost, power, and weight. Supermicro, Dell, and HP have such product lines for cloud providers. So, if you are buying equipment for large clusters, take a look at these stripped-down "cloud servers".

For DataNodes, a single power supply unit (PSU) is sufficient, but for a NameNode, use redundant PSUs. Server designs that share PSUs across adjacent servers can offer increased reliability without increased cost.

Some co-location sites bill based on the maximum-possible power budget and not the actual budget. In such a location, the benefits of the power saving features of the latest CPUs are not realized completely. We therefore recommend checking the power billing options of the site in advance.

Network

This is the most challenging parameter to estimate because ADH workloads vary a lot. The key is buying enough network capacity at reasonable cost so that all nodes in the cluster can communicate with each other at reasonable speeds. Large clusters typically use dual 1 Gbps links for all nodes in each 20-node rack and 2*10 Gbps interconnect links per rack going up to a pair of central switches.

A good network design considers the possibility of unacceptable congestion at critical points in the network under realistic loads. Generally accepted oversubscription ratios are around 4:1 at the server access layer and 2:1 between the access layer and the aggregation layer or core. Lower oversubscription ratios can be considered if higher performance is required. Additionally, we also recommend having 1 Gbps oversubscription between racks.

It is critical to have dedicated switches for the cluster instead of trying to allocate a VC (virtual connect) in existing switches — the load of an ADH cluster would impact the rest of the users of the switch. It is also equally critical to work with the networking team to ensure that the switches suit both ADH and their monitoring tools.

Design the networking so as to retain the option of adding more racks of ADH servers. Getting the networking wrong can be expensive to fix. The quoted bandwidth of a switch is analogous to the miles per gallon ratings of an automobile — you are unlikely to replicate it. "Deep buffering" is preferable to low-latency in switches. Enabling Jumbo Frames across the cluster improves bandwidth through better checksums and possibly may also provide packet integrity.

NameNode

Albert Bagdasaryan

Contents

To Table of Contents

Storage options
Memory options
Processors
Network

To ensure high availability, use both primary and secondary NameNodes. They are crucial parts of any Hyperwave cluster and should be highly available. Both servers keep the HDFS state in the *fsimage* file and logs in the *edits* file.

NameNodes perform the following:

- all operations with files in the HDFS;
- mapping files and DataNode blocks;
- storing metadata of HDFS files and folders;
- storing DataNode block locations;
- controlling data replication.

The secondary NameNode is a reserve storage of the *fsimage* and *edits* files. It periodically updates the *fsimage* file from the *edits* log, thus preventing the latter from becoming too large.

Storage options

Both NameNode servers should have highly reliable storage for their namespace storage and edits log. Typically, hardware RAID and reliable network storage are justifiable options.

For ADH NameNodes, regardless of the number of DataNodes, the storage characteristics are consistent. Use four near 1 TB SAS drives with a RAID HDD controller configured for RAID 1+0. SAS drives are more expensive than SATA drives and have lower storage capacity, but they are faster and much more reliable.

Deploying your SAS drives as a RAID array ensures that the ADH management services have a fast, stable, and redundant storage for their mission-critical data.

Memory options

Memory requirements vary considerably depending on the scale of an ADH cluster. Memory is a critical factor for NameNodes, because the active and standby NameNode servers rely heavily on RAM to manage HDFS. As such, use error-correcting memory (ECC) in ADH NameNodes. NameNodes usually require from 64 to 128 GB of RAM.

The NameNode memory requirement is a direct function of the number of file blocks stored in HDFS. As a rule, a NameNode uses roughly 1 GB of RAM per million HDFS blocks.

> NOTE
>
> Remember that files are broken down into individual blocks and replicated so that you have three copies of each block, not the file.

Processors

It is recommended to use motherboards with two CPU sockets, each with eight cores 2.5—3 GHz. The Intel architecture is commonly used.

Network

Fast communication is vital for the services on NameNodes, so it is recommended using a pair of bonded 10 Gbps connections. This bonded pair provides redundancy and also doubles throughput to 20 Gbps. For smaller clusters (less than 50 nodes) 1 Gbps connectors can also be used.

DataNode

Albert Bagdasaryan

Contents

To Table of Contents

Do we need RAID

Storage options

Memory options

Processors

Network

DataNode is a Hadoop primary storage and computational resource. Many specialists may think that since DataNodes play such an important role in a cluster, we should use the best hardware available for them. This is not entirely true. Hadoop was designed with an idea that DataNodes are like "disposable workers" that are fast enough to do useful work as a part of the cluster, but cheap enough to be easily replaced if they fail. Frequency of hardware failures in large clusters is probably one of the most important considerations that core Hadoop developers had in mind. Hadoop addresses this issue by moving the redundancy implementation from the cluster hardware to the cluster software itself.

DataNode does the following operations:

- writing and reading of data blocks by client requests;
- executing MapReduce tasks;
- sending a heartbeat and the block states to the NameNode (block reports);
- participating in the replication process.

Do we need RAID

Hadoop provides redundancy on many levels. Each DataNode stores only some blocks of HDFS files and those blocks are replicated multiple times to different nodes, so in the event of a single server failure, data remains accessible. The cluster can even tolerate multiple nodes' failure, depending on the configuration you choose. Hadoop goes beyond that and allows you to specify which servers reside on which racks and tries to store copies of data on separate racks. In this way, it significantly increases probability that your data remains accessible even if the whole rack goes down (though this is not a strict guarantee). This design means that there is no reason to invest into the RAID controller for Hadoop DataNodes.

Storage options

Usually one DataNode has 12-24 disks with 1 TB capacity. Instead of using RAID for local disks, a JBOD (Just a Bunch of Disks) is a preferred choice. It provides better performance for Hadoop workload and reduces hardware costs. You don't have to worry about individual disk failure since redundancy is provided by HDFS.

The number of DataNodes depends on the NameNode RAM size. Remember that the NameNode uses roughly 1 GB of RAM per million HDFS blocks. By default the size of one HDFS block is 128 MB.

A typical implementation is to use 19" racks or cabinets with 1U-2U servers.

Memory options

The second DataNode role is to serve as a data processing node and execute custom MapReduce code. MapReduce jobs are split into lots of separate tasks, which are executed in parallel on multiple DataNodes so that for a job to produce logically consistent results, all subtasks must be completed.

Processors

We recommend to use motherboards with two CPU sockets, each with eight cores 2.5-3 GHz. The Intel architecture is commonly used.

Network

Fast communication is as vital for DataNodes as for the NameNode, so we recommend to use a pair of bonded 10 Gbps connections. This bonded pair provides redundancy and also doubles throughput to 20 Gbps. For smaller clusters (less than 50 nodes) you could get away with using 1 Gbps connectors.

YARN

Albert Bagdasaryan

To Table of Contents

YARN stands for Yet Another Resource Negotiator. The critical function of Hadoop YARN is to supply the computational resources required for applications by distinguishing between resource management and job scheduling. For more details on this component, see YARN architecture.

Generally, hardware recommendations for a YARN Resource Manager machine are the same as for a NameNode. If happens that both NameNodes, primary and secondary, fail, you can always use the YARN Resource Manager node as a NameNode for a while.

PostgreSQL nodes used with Hive Metastore

Eugenia Kuzina

To Table of Contents

External database sizing for the Apache Hive Metastore requires careful consideration of various factors such as the number of tables, partitions, users, and overall workload. Although it is difficult to determine the exact size of an external database, the following guidelines can help you make an informed decision:

- **Estimate the database size**
  - **The number of tables**. Each table requires a storage for schema and metadata, ranging from a few kilobytes to several megabytes per table. It depends on the schema complexity.
  - **The number of partitions**. The storage requirements for partitions depend on the partition key data types and the number of partition columns. On average, you can estimate around 1 KB per partition.
  - **User information and permissions**. This requires a small amount of storage, usually a few kilobytes per user.
- **Historical and growth data**. Consider your data’s historical growth rate and expected future growth. Regularly monitor the database size and scale it as needed.
- **Replication and backup**. Allocate additional storage for backup and replication to maintain redundancy and ensure data durability. 50% of the total database size is recommended for backup quota.
- **Workload**. Estimate the concurrency of queries and the number of users who will access the system simultaneously. A higher workload may require more resources and storage for caching, query planning, and temporary storage.
- **Buffer for performance optimization**. Leave some additional storage space for indexing, caching, and other performance optimizations, typically around 20-30% of the total database size.

While these guidelines can help you estimate the size of the external database for the Hive Metastore, it is crucial to keep monitoring usage and adjust the size accordingly.

> **IMPORTANT**
>
> The following system requirements are **minimal**. The target sizing should be calculated based on the customer requirements.

The minimum hardware requirements for the external database hosts are listed in the table below.

Requirement	Small cluster	Medium cluster	Large cluster
Number of concurrent connections (user sessions)	5-10	20-50	50+
CPU	64bit is recommended *, 4+ dedicated cores	64bit, 4-8 cores	64bit, 8+ cores CPUs with larger L3 caches are more suitable for larger sets of data
RAM	8-16 GB	16-32 GB	32+ GB
Disk space (minimum 1 separate disk, SSD, RAID 1 or RAID 10)	50+ GB	100 GB	200+ GB

* If you need to use a 32-bit binary, set `LDR_CNTRL` to `MAXDATA=0xn0000000`, where 1 <= n <= 8, before starting the PostgreSQL server, try different values and *postgresql.conf* settings to find a configuration that operates satisfactorily. For more information, see Platform-Specific Notes.

Hardware requirements depending on workload patterns

Albert Bagdasaryan

Contents

To Table of Contents

- Patterns
  - Same wheels
  - Compute intensive
  - Storage intensive
  - Balanced workload
- General recommendations

The Hyperwave workloads present many resource-management challenges and potential conflicts. There are different types of conflicts, for example:

- between long-running, resource-intensive jobs and shorter interactive queries;
- between Hyperwave and non-Hyperwave workloads running on the same cluster and competing for the same resources.

In general, there are three main types of workload:

- compute intensive;
- storage intensive;
- balanced.

Also, there could be situations, when you may not know your eventual workload patterns at first. Initial actions with an ADH cluster are usually very different from the actual jobs you will run in your production environment. Therefore, it is recommended following the advice for the balanced workload in a pilot ADH cluster. After that, you can plan or rescale the cluster according to the real workload.

Patterns

Same wheels

ADH nodes are like wheels that move everything. If the wheels are identical, then the movement is carried out smoothly and without jerks. But if they are different, then there may be different problems in the uniformity of movement. It is recommended to use the same configuration on all cluster machines with minor differences between DataNodes and between NameNodes.

The following characteristics must be the same for all cluster machines:

- CPU;
- RAM;
- network.

See more details in General recommendations.

Compute intensive

This workload type is CPU-related and characterized by the need of a large number of CPU cores and large amount of memory to store in-process data. This usage pattern is typical for natural language processing or HPCC workloads. For the compute intensive patterns, it is necessary to use at least 10 CPU cores per machine.

Storage intensive

For this type of workload, it is recommended to invest in more disks per machine. The number of machines depends on the volume of data to store and analyze, which determines the required number of disks per machine. The latter is usually uniformal throughout the cluster.

For a low density server, the main aim is to keep the cost low to be able to afford a large number of machines. Eight CPU cores match this requirement and will give reasonable processing power. Each map or reduce task will utilize a single CPU core, but since some time will be spent waiting for I/O operations, it makes sense to oversubscribe for CPU cores. With 8 cores available, it is possible to configure about 12 map and reduce slots per node.

The optimal size of one hard disk is 2-3 TB. On average, you can install 12 disks in each machine.

Balanced workload

The logic behind choosing and balancing hardware components for a high density cluster is the same as for a lower density one. As an example of such a configuration, you can choose a machine with 12 2-3 TB hard drives or 24 1 TB hard drives. Having lower capacity disks per server is preferable, because it will provide better input and output throughput and better fault tolerance. To increase the computational power of an individual machine, use 8 CPU cores and 128-256 GB of RAM.

General recommendations

IMPORTANT

The following system requirements are **minimal**. The target sizing should be calculated based on the customer requirements.

!

In this table, there are the most common hardware recommendations based on the workload pattern for an ADH cluster.

Server	Load pattern	Storage	CPU	RAM	Network
DataNodes	Balanced workload	12 Disks 2-3 TB JBOD	8 cores	128-256 GB	1 Gbps onboard, 2x10 Gbps interconnect
	Compute-intensive workload	12 Disks 1-2 TB JBOD	10 cores	128-256 GB	10 Gbps onboard, 2x10 Gbps interconnect
	Storage-heavy workload	12 Disks 4+ TB JBOD	8 cores	128-256 GB	10 Gbps onboard, 2x10 Gbps interconnect
NameNode	Balanced workload	4+ Disks 2-3 TB RAID 10	8 cores	128-256 GB	10 Gbps onboard, 2x10 Gbps interconnect
Resource Manager/YARN	Balanced workload	4+ Disks 2-3 TB RAID 10	8 cores	128-256 GB	10 Gbps onboard, 2x10 Gbps interconnect

Filesystem requirements

Eugenia Kuzina

Contents

To Table of Contents

Supported file systems
Use noatime option to improve performance
Limitations of using file system mount options
Umask settings

Supported file systems

The Hadoop Distributed File System (HDFS) is designed to operate on top of the underlying file system in the operating system. The following operating systems are supported:

- **ext3** — the most tested underlying file system for HDFS;
- **ext4** — a scalable extension of ext3;
- **XFS** — the default file system in RHEL 7.

If you are choosing between ext3 and ext4, ext4 is recommended.

Use noatime option to improve performance

Linux file systems store metadata that records when each file was accessed. It means that each read operation also writes to the disk. Disable this functionality to speed up file reads. To do this, add the `noatime` mount option to each line that addresses a file system in the */etc/fstab* file, for example:

```
/dev/sdb1 /data1 xfs defaults,noatime 0
```

Use the following command to apply changes without rebooting:

```
$ mount -o remount /data1
```

Limitations of using file system mount options

- The `sync` mount option allows you to write synchronously. Using `sync` reduces performance for services that write data to disks (for example, HDFS and YARN). In ADH, most writes are replicated. That is why synchronous writing to disk is unnecessary, expensive, and does not provide a noticeable improvement in the cluster stability. It is not recommended to use `sync`.
- The `nfs` and `nas` options are not supported to mount a data directory of DataNode.
- Mounting */tmp* as a file system with the `noexec` option is not supported. This approach is used to prevent the execution of stored files.

Umask settings

UNIX-based systems use umask (user file-creation mode mask) to set default permissions for created files and directories. In most Linux distributions, the umask default value is `0022 (022)` or `0002 (002)`. The basic rights for a directory are `0777 (rwxrwxrwx)`, and for a file — `0666 (rw-rw-rw)`. Subtract umask from basic rights to determine rights when umask is applied. The default umask `0002` is used for an ordinary user. With this mask, the default rights for a directory are `775`, and for a file `664`. For the superuser (root), the default umask is `0022`. With this mask, the default rights for a directory are `755`, and for a file `644`.

You can set umask in the *etc/bashrc* or */etc/profile* file.

ADH supports umask values listed in the table below.

Supported umask values

The umask value	File				Directory			
	Result	Owner	Group	Others	Result	Owner	Group	Others
0022 (recommended)	644	rw-	r--	r--	755	rwx	r-x	r-x
0002	664	rw-	rw-	r--	775	rwx	rwx	r-x
0000	666	rw-	rw-	rw-	777	rwx	rwx	rwx

ADH network ports

Eugenia Kuzina

Contents

To Table of Contents

- ADH network port mapping
  - Arenadata PostgreSQL
  - Airflow2
  - Flink
  - Flink2
  - HBase
  - HBaseThrift2 Server
  - HDFS
  - Hive
  - HUE
  - Impala
  - Kyuubi
  - Monitoring (service)
  - Monitoring (cluster)
  - Ozone
  - Solr
  - Spark3
  - SSM
  - Trino
  - YARN
  - Zeppelin
  - Zookeeper
- JMX ports
- Ports redefined by Kerberos
- Client ports
  - Examples
- ADCM ports
- Arenadata Enterprise Tools ports
- Arenadata Monitoring ports

ADH network port mapping

The tables below list ports of the ADH services and components. The ***Config*** column contains a file name where the port number can be redefined.

Arenadata PostgreSQL

Component/Server	Port	Protocol	Description	Parameter	Config
Arenadata PostgreSQL	5432	TCP	PostgreSQL port for DB connections	port	postgresql.conf

Airflow2

Component/Server	Port	Protocol	Description	Parameter	Config
Airflow2 Broker	6379	TCP	Redis service port	broker_url (redis_port in ADCM)	airflow.cfg
Airflow2 Flower	5555/5555	HTTP/HTTPS	Flower Web UI port	flower_port	airflow.cfg
Airflow2 Scheduler	5432	TCP	PostgreSQL service port	sql_alchemy_conn, result_backend	airflow.cfg
Airflow2 Server	8080/8080	HTTP/HTTPS	Airflow Server Web UI port	web_server_port (server_port in ADCM)	airflow.cfg
Airflow2 Worker	8793/8793	HTTP/HTTPS	Celery Worker API port	worker_log_server_port (worker_port in ADCM)	airflow.cfg

Flink

Component/Server	Port	Protocol	Description	Parameter	Config
Flink History Server	8082	HTTP/HTTPS	Flink History Server web UI port	historyserver.web.port	flink-conf.yaml
Flink JobManager	8081/8081	HTTP/HTTPS	Flink Job Manager web UI port	rest.port	flink-conf.yaml
Flink JobManager	6123	TCP	Flink Job Manager RPC port (non-HA mode)	jobmanager.rpc.port	flink-conf.yaml
Flink JobManager	Range generated by Zookeeper (for example, 50100)	TCP	Flink Job Manager port (HA mode)	high-availability.jobmanager.port	—
Flink SQL-Gateway	8083	TCP	Port of the SQL Gateway REST endpoint	sql-gateway.endpoint.rest.port	flink-conf.yaml
Flink TaskManager	Range (for example, 35122)	TCP	Data port (range)	taskmanager.data.port	—
Flink TaskManager	Range (for example, 32390)	IPC	IPC port (range)	taskmanager.rpc.port	—

Flink2

Component/Server	Port	Protocol	Description	Parameter	Config
Flink2 History Server	8092	HTTP/HTTPS	Port of the History Server's web interface	historyserver.web.port	config.yaml
Flink2 SQL-Gateway	8093	TCP	Port of the SQL Gateway REST endpoint	sql-gateway.endpoint.rest.port	config.yaml

HBase

Component/Server	Port	Protocol	Description	Parameter	Config
HBase Master Server	16000	IPC	HBase Master IPC port	hbase.master.port	A custom parameter in hbase-site.xml
HBase Master Server	16010/16010	HTTP/HTTPS	HBase Master Web UI port	hbase.master.info.port	A custom parameter in hbase-site.xml
HBase Master Server	9211	HTTP/HTTPS	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring
HBase Phoenix Query Server	8765/8765	HTTP/HTTPS	API Port	phoenix.queryserver.http.port	A custom parameter in hbase-site.xml
HBase Phoenix Query Server	9213	HTTP	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring
HBase REST Server	60080/60080	HTTP/HTTPS	API Port	hbase.rest.port	hbase-site.xml
HBase REST Server	8085	HTTP/HTTPS	Web UI Port	hbase.rest.info.port	A custom parameter in hbase-site.xml
HBase REST Server	9215	HTTP	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring
HBase RegionServer	16030/16030	HTTP/HTTPS	HBase RegionServer Web UI port	hbase.regionserver.info.port	A custom parameter in hbase-site.xml
HBase RegionServer	16020	IPC	HBase RegionServer IPC port	hbase.regionserver.port	A custom parameter in hbase-site.xml
HBase RegionServer	9212	HTTP	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring

HBaseThrift2 Server

Component/Server	Port	Protocol	Description	Parameter	Config
HBase Thrift2 Server	9090	TCP	Thrift2 Server port	—	—
HBase Thrift2 Server	9095/9095	HTTP/HTTPS	Thrift2 Server Web UI port	hbase.thrift.info.port	A custom parameter in hbase-site.xml
HBase Thrift2 Server	9214	HTTP	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring

HDFS

Component/Server	Port	Protocol	Description	Parameter	Config
HDFS DataNode	9864/9865	HTTP/HTTPS	DataNode Web UI port	dfs.datanode.https.address	hdfs-site.xml
HDFS DataNode	9866	TCP	Data transfer	dfs.datanode.address	hdfs-site.xml
HDFS DataNode	9867	IPC	File system metadata operations	dfs.datanode.ipc.address	hdfs-site.xml
HDFS DataNode	9202	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
HDFS HttpFS server	14000/14000	HTTP/HTTPS	HTTPfs API port	httpfs.http.port	httpfs-site.xml
HDFS JournalNode	8480/8481	HTTP/HTTPS	JournalNode Web UI port	dfs.journalnode.http.address, dfs.journalnode.https.address	hdfs-site.xml
HDFS JournalNode	8485	IPC	File system metadata operations	dfs.journalnode.rpc-address	hdfs-site.xml
HDFS JournalNode	9203	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
HDFS NameNode	9870/9871	HTTP/HTTPS	NameNode Web UI port	dfs.namenode.http-address, dfs.namenode.https-address	hdfs-site.xml
HDFS NameNode	8020	IPC	File system metadata operations	Embedded in dfs.namenode.rpc-address.<namespace>.<namenode_id>	hdfs-site.xml
HDFS NameNode	9201	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
HDFS ZKFC	8019	IPC (TCP, UDP)	ZooKeeper FailoverController for NN HA	dfs.ha.zkfc.port	A custom parameter in hdfs-site.xml

Hive

Component/Server	Port	Protocol	Description	Parameter	Config
Hive HiveServer2	10000	TCP or HTTP	Thrift/JDBC port	hive.server2.thrift.port	hive-site.xml
Hive HiveServer2	10002/10002	HTTP/HTTPS	Hive Server Web UI port	hive.server2.webui.port	A custom parameter in hive-site.xml
Hive HiveServer2	9208	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
Hive Metastore	9083	TCP	Hive Metastore service port	hive.metastore.uris	hive-site.xml
Hive Metastore	9207	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
Hive Tez UI	8089	HTTP/HTTPS	Tez Web UI port	Embedded in tez.tez-ui.history-url.base	tez-site.xml

HUE

Component/Server	Port	Protocol	Description	Parameter	Config
HUE Server	8000/8000	HTTP/HTTPS	HUE Server web UI port	desktop.http_port	hue.ini
HUE Server	8000	HTTP	Port to share monitoring metrics	desktop.http_port	hue.ini

Impala

Component/Server	Port	Protocol	Description	Parameter	Config
Impala Catalog Service	23020	TCP	Internal use only. The Catalog Server daemon listens on this port for updates from the StateStore daemon	state_store_subscriber_port	/etc/impala/conf/catalogstore.conf
Impala Catalog Service	25020	HTTP/HTTPS	Port on which the Catalog Service web server is running. See Impala web UIs overview	webserver_port	/etc/impala/conf/catalogstore.conf
Impala Catalog Service	26000	TCP	Internal use only. The Catalog Server uses this port to communicate with the Impala daemons	catalog_service_port	/etc/impala/conf/catalogstore.conf
Impala Catalog Service	25020	HTTP	Port to share monitoring metrics	webserver_port	catalogstore.conf
Impala Daemon	21000	TCP	Port on which Impala daemons serve Beeswax client requests	beeswax_port	/etc/impala/conf/impalastore.conf
Impala Daemon	21050	TCP	Port on which Impala daemons serve HiveServer2 client requests	hs2_port	/etc/impala/conf/impalastore.conf
Impala Daemon	22000	TCP	Internal use only. Impala daemons use this port for Thrift-based communication with each other	be_port	/etc/impala/conf/impalastore.conf
Impala Daemon	23030	TCP	Internal use only. Impala daemons listen on this port for updates from the Statestore daemon	state_store_subscriber_port	/etc/impala/conf/impalastore.conf
Impala Daemon	25000	HTTP/HTTPS	Port where the Impala daemon web server is running. See Impala web UIs overview	webserver_port	/etc/impala/conf/impalastore.conf
Impala Daemon	27000	TCP	Internal use only. Impala daemons use this port for KRPC-based communication with each other	krpc_port	/etc/impala/conf/impalastore.conf
Impala Daemon	28000	HTTP	Port is used by client applications to transmit commands and receive results over HTTP via the HiveServer2 protocol	hs2_http_port	/etc/impala/conf/impalastore.conf
Impala Daemon	25000	HTTP	Port to share monitoring metrics	webserver_port	impalastore.conf
Impala Statestore	24000	TCP	Port on which the Impala Statestore component is running	state_store_port	/etc/impala/conf/statestore.conf
Impala Statestore	25010	HTTP/HTTPS	Port on which the Statestore web server is running. See Impala web UIs overview	webserver_port	/etc/impala/conf/statestore.conf
Impala Statestore	25010	HTTP	Port to share monitoring metrics	webserver_port	/etc/impala/conf/statestore.conf

Kyuubi

Component/Server	Port	Protocol	Description	Parameter	Config
Kyuubi Server	10099	HTTP/HTTPS	Port on which the REST frontend service runs	kyuubi.frontend.rest.bind.port	kyuubi-defaults.conf
Kyuubi Server	10009	Binary	Port on which the Thrift frontend service runs via a binary protocol	kyuubi.frontend.thrift.binary.bind.port	kyuubi-defaults.conf
Kyuubi Server	10010	HTTP/HTTPS	Port on which the Thrift frontend service runs via HTTP	kyuubi.frontend.thrift.http.bind.port	kyuubi-defaults.conf
Kyuubi Server	10019	HTTP	Port to share monitoring metrics	kyuubi.metrics.prometheus.port	kyuubi-defaults.conf

Monitoring (service)

Component/Server	Port	Protocol	Description	Parameter	Config
Prometheus	11200	HTTP	Port used by Prometheus for incoming HTTP requests (UI, API, metrics)	web.listen-address	Prometheus settings
Grafana	11210	HTTP	Port, on which the Grafana server listens for incoming requests	Grafana listen port	Grafana settings
Node exporter	11203	HTTP	Port, on which the Node exporter server listens for incoming requests	Listen port	Node Exporter settings

Monitoring (cluster)

Component/Server	Port	Protocol	Description	Parameter	Config
Graphite	2015	TCP	Port to send metrics	port	diamond.conf

Ozone

Component/Server	Port	Protocol	Description	Parameter	Config
Ozone Datanode	9882/9883	HTTP/HTTPS	HTTP address for the DataNode web interface	hdds.datanode.http-address/hdds.datanode.https-address	ozone-site.xml
Ozone Datanode	9855	RPC	Port used for Ratis DataStream on the DataNode	hdds.container.ratis.datastream.port	ozone-site.xml
Ozone Manager	9862	RPC	Port of the Ozone Manager	ozone.om.address	ozone-site.xml
Ozone Manager	9874/9875	HTTP/HTTPS	Specifies the HTTP address for the Ozone Manager web interface	ozone.om.http-address/ozone.om.https-address	ozone-site.xml
Ozone Manager	9858	RPC	Port for Ratis communication used by Ozone Manager	ozone.om.ratis.port	ozone-site.xml
Ozone Manager	9874	HTTP	Port to share monitoring metrics	ozone.om.http-address	ozone-site.xml
Ozone Recon	9888/9889	HTTP/HTTPS	HTTP address for the Recon web interface, used to view metrics and reports for the Ozone cluster	ozone.recon.http-address/ozone.recon.https-address	ozone-site.xml
Ozone S3G	9878/9879	HTTP/HTTPS	HTTP address for the S3 Gateway, which provides the S3-compatible API interface for Ozone	ozone.s3g.http-address/ozone.s3g.https-address	ozone-site.xml
Ozone Storage Container Manager	9876/9877	HTTP/HTTPS	Specifies the HTTP address for the SCM web interface	ozone.scm.http-address/ozone.scm.https-address	ozone-site.xml
Ozone Storage Container Manager	9894	RPC	Port for Ratis communication used by SCM for high availability	ozone.scm.ratis.port	ozone-site.xml
Ozone Storage Container Manager	9876	HTTP	Port to share monitoring metrics	ozone.scm.http-address	ozone-site.xml

Solr

Component/Server	Port	Protocol	Description	Parameter	Config
Solr Server	8983/8985	HTTP/HTTPS	Solr Server Web UI and API port	hostPort	solr.xml
Solr prometheus exporter	9983	HTTP	Solr exporter HTTP port	Option port	—

Spark3

Component/Server	Port	Protocol	Description	Parameter	Config
Spark3 Connect	15002	HTTP/HTTPS	Port number to connect to the Spark3 Connect gRPC endpoint	spark.connect.grpc.binding.port	spark-defaults.conf
Spark3 History Server	18092	HTTP/HTTPS	HS Web UI port	spark.history.ui.port, spark.ssl.historyServer.port	spark-defaults.conf
Spark3 Livy Server	8999	HTTP/HTTPS	Livy Server Web UI port	livy.server.port	livy.conf

SSM

Component/Server	Port	Protocol	Description	Parameter	Config
SSM Agent	7044	RPC	Port number used by SSM agents to communicate with the SSM Server	smart.agent.port	smart-site.xml
SSM Agent	7051	RPC	Port number used by the SSM Server to communicate with SSM agents	smart.agent.master.port	smart-site.xml
SSM Server	7042	RPC	Port number used by the SSM Server to interact with NameNodes	smart.server.rpc.address	smart-site.xml
SSM Server	7045	HTTP/HTTPS	Port number used by the SSM web UI	smart.server.rpc.address	smart-site.xml

Trino

Component/Server	Port	Protocol	Description	Parameter	Config
Coordinator	18188	HTTP/HTTPS	Prometheus metrics port	http-server.http.port	config.properties
Coordinator	9209	HTTP/HTTPS	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring
HAProxy	7001	HTTP/HTTPS	Sets the HTTP port for the server	http-server.http.port	config.properties
Worker	18189	HTTP/HTTPS	Sets the HTTP port for the server	http-server.http.port	config.properties
Worker	9210	HTTP/HTTPS	Sets the Prometheus metrics port	Prometheus metrics port	Monitoring

YARN

Component/Server	Port	Protocol	Description	Parameter	Config
MapReduce History Server	19888/19890	HTTP/HTTPS	HS Web UI port	mapreduce.jobhistory.webapp.address, mapreduce.jobhistory.webapp.https.address	mapred-site.xml
MapReduce History Server	10020	IPC	MapReduce JobHistory Server port	mapreduce.jobhistory.address	mapred-site.xml
MapReduce History Server	10033	IPC	Port of the History server admin interface	mapreduce.jobhistory.admin.address	A custom parameter in mapred-site.xml
MapReduce History Server	13562	IPC	MapReduce Shuffle port	mapreduce.shuffle.port	A custom parameter in mapred-site.xml
YARN NodeManager	8042/8044	HTTP/HTTPS	NM Web UI port	8042 is yarn.nodemanager.webapp.address, 8044 is yarn.nodemanager.webapp.https.address	yarn-site.xml (only 8044)
YARN NodeManager	8040	IPC	Port where the localizer IPC is	yarn.nodemanager.localizer.address	A custom parameter in yarn-site.xml
YARN NodeManager	8041	IPC	Port of container manager in the NM	yarn.nodemanager.address	yarn-site.xml
YARN NodeManager	7337	TCP	Spark shuffle service port (The service is disabled by default)	spark.shuffle.service.port	custom spark-defaults.conf
YARN NodeManager	7447	TCP	Spark shuffle service port	—	—
YARN NodeManager	9205	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
YARN ResourceManager	8088/8090	HTTP/HTTPS	RM Web UI Port	yarn.resourcemanager.webapp.address, yarn.resourcemanager.webapp.https.address	yarn-site.xml (only 8090)
YARN ResourceManager	8030	HTTP	Port of the scheduler interface	yarn.resourcemanager.scheduler.address	A custom parameter in yarn-site.xml
YARN ResourceManager	8031	IPC	Resource tracker port	yarn.resourcemanager.resource-tracker.address	A custom parameter in yarn-site.xml
YARN ResourceManager	8032	IPC	Port of the application manager interface in the RM	yarn.resourcemanager.address	A custom parameter in yarn-site.xml
YARN ResourceManager	8033	IPC	Port of the RM admin interface	yarn.resourcemanager.admin.address	A custom parameter in yarn-site.xml
YARN ResourceManager	9204	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring
YARN Timeline Server	8188/8190	HTTP/HTTPS	TS Web UI port	yarn.timeline-service.webapp.https.address	yarn-site.xml
YARN Timeline Server	10200	IPC	TS Server port	Embedded in mapreduce.jobhistory.address	mapred-site.xml
YARN Timeline Server	9206	HTTP	Port to share monitoring metrics	Prometheus metrics port	Monitoring

Zeppelin

Component/Server	Port	Protocol	Description	Parameter	Config
Zeppelin Server	8180/8180	HTTP/HTTPS	Zeppelin Web UI port	zeppelin.server.port, zeppelin.server.ssl.port	zeppelin-site.xml
Zeppelin Server	8180	HTTP	Prometheus metrics port	zeppelin.server.port	zeppelin-site.xml

Zookeeper

Component/Server	Port	Protocol	Description	Parameter	Config
Zookeeper Server	2888	TCP	Port for cluster coordinating (internal cluster use only)	server.*	zoo.cfg
Zookeeper Server	3888	TCP	Port for cluster coordinating (internal cluster use only)	server.*	zoo.cfg
Zookeeper Server	2181	TCP	Port for clients connection	clientPort (client_port in ADCM)	zoo.cfg
Zookeeper Server	5181	HTTP	Zookeeper HTTP API interface	admin.serverPort (admin.serverPort in ADCM)	zoo.cfg
Zookeeper Server	7000	HTTP	Sets the Prometheus metrics port	metricsProvider.httpPort	zoo.cfg

JMX ports

Only the YARN RM port is defined in ADCM UI. Other JMX ports are hardcoded in service environment configs.

Service	Component/Server	Port	Protocol	Description	Parameter	Config
HBase	HBase Region	10102	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	hbase-regionserver-env.sh
HDFS	NameNode	1026	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	hadoop-env.sh
HDFS	DataNode	1027	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	hadoop-env.sh
Hive	Hive Server	1031	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	hive-server2-env.sh
Hive	Metastore	1032	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	hive-metastore-env.sh
Monitoring	jmxtrans	2101	TCP	JMX port for clients connection	-Dcom.sun.management.jmxremote.port	/etc/jmxtrans/wrapper.conf
Spark	History Server	18083	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	spark-env.sh
Spark3	History Server	18093	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	spark-env.sh
YARN	Resource Manager	1028	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port (resource_manager_jmx_port in ADCM)	yarn-env.sh
YARN	NodeManager	1029	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	yarn-env.sh
YARN	MapReduce History Server	1030	TCP	JMX port (internal use with jmxtrans)	-Dcom.sun.management.jmxremote.port	mapred-env.sh

Ports redefined by Kerberos

When Kerberos is enabled, ADCM redefines ports from the table below by hardcoded values from Ansible config.

On disabling Kerberos, ADCM returns the default hardcoded values for these parameters.

Service	Component/Server	Port	Protocol	Description	Kerberos	Parameter
HDFS	DataNode	1022/50475	HTTP/HTTPS	DataNode Web UI port/ secure port	Enabled	dfs.datanode.http.address, dfs.datanode.https.address
HDFS	DataNode	1019	TCP	Data transfer port	Enabled	dfs.datanode.address

Client ports

Client ports are generated in the local port range. You can determine a specific port from this range using a known service port. Use the lsof command as follows:

```
$ sudo lsof -i -P -n | grep <known port namber>
```

Examples

1. The Diamond client is running on a local random port. It connects to Graphite that listens the `2015` port.

```
$ sudo lsof -i -P -n | grep 2015
```

The result:

```
python    6000       root    2u  IPv4 4811652      0t0  TCP 10.92.16.232:34758->10.92.16.30:2015 (ESTABLISHED)
```

2. The Hive client is running on a local random port. It connects to ADPG that listens the `5432` port.

```
$ sudo lsof -i -P -n | grep 5432
```

The result:

```
java       1161       hive  783u  IPv6 2043472      0t0  TCP 10.92.41.49:58094->10.92.41.242:5432 (ESTABLISHED)
```

Examples of client ports in the local port range

Component	Port	Protocol	Description
Diamond (python-process, running by root)	34758 or another random port from local port range	TCP	Local jmxtrans client port for sending metrics to Monitoring cluster on port 2015 (to Graphite)
Hive	58094 or another random port from local port range	TCP	Local client port to connect to ADPG database on 5432 port
HBase, Spark, and etc.	50382 or another random port from local port range	TCP	Local client port to connect to HDFS on port 8020
Flink JobManager	43162 or another random port from local port range	TCP	Flink client connects to Zookeeper: TCP 10.92.17.55:42844 → 10.92.17.55:2181 Flink internal connect between JobManager's nodes: TCP 10.92.17.55:60390 → 10.92.17.58:33769

ADCM ports

Port	Protocol	Description
22	TCP	ADCM connection port

Arenadata Enterprise Tools ports

Port	Protocol	Description
443	TCP	Port that is used by Docker Registry
81	TCP	Port that is used to access repositories during offline installation

Arenadata Monitoring ports

Port	Protocol	Description
3000	HTTP	HTTP port to access the Grafana web interface. You can change the default value when installing the monitoring cluster on the configuration page of the Grafana service in the ***ip_and_ports*** section
80	TCP	TCP port to access the Graphite web interface. You can change the default value when installing the monitoring cluster on the configuration page of the Graphite service in the ***ip_and_ports*** section

Pilot cluster

Albert Bagdasaryan

Contents

To Table of Contents

- Parameters
- Hardware
  - Data volume
  - DataNodes number
  - CPU and RAM

This article represents a plan of a pilot ADH cluster based on balanced workload pattern. Even a small and simple cluster needs to have at least three DataNodes and one NameNode. You can use one physical machine (or a virtual machine) or three different machines — it is up to you.

For more details about workloads, see Hardware requirements depending on workload patterns.

Parameters

For a pilot cluster example the following parameters will be used:

- Data volume — 500 TB.
- Replication factor — 3.
- Data retention period — one year.
- Balanced workload.
- Data formats distribution:
  - Plain text, AVRO, Parquet, Jason, ORC, and so on — 20%;
  - GZIP and Snappy — 80%.

Hardware

The specification for DataNodes depends on the stored and analyzed data volume.

Data volume

According to the replication factor of 3, the storage volume of $500\text{TB} \cdot 3 = 1500\text{TB}$ is required for storing data for one year. Assume 20% of the data is in the container storage and the other 80% is in the Snappy compressed Parquet format. Parquet Snappy compresses data to approximately 70-80% of the original volume. For this example, 80% will be assumed. Below is the formula for the storage requirement calculation:

$$\text{Total storage required for data} = \text{Total storage} \cdot \% \text{ in container storage} + \text{Total storage} \cdot \% \text{ in compressed format} \cdot \text{Expected compression ratio}$$

With the planned parameters, the required storage is $1500\text{TB} \cdot 0.2 + 1500\text{TB} \cdot 0.8 \cdot (1 - 0.8) = 540\text{TB}$

Apart from the data, space for processing the data and for some other tasks is also required. Assume that only 10% of the data is processed each day on average and this process creates three times temporary data. This means that approximately 30% of total data storage volume is required as extra storage.

The total storage required for data and other activities is $540\text{TB} + 540\text{TB} \cdot 0.3 \cdot 3 = 702\text{TB}$.

As for the DataNode, JBOD is recommended. 20% of data storage have to be allocated for the JBOD file system needs. Therefore, the data storage requirement will increase by 20%. So the final figure in this example is $702\text{TB} \cdot (1 + 0.2) \cong 842.5\text{TB}$, 845 TB if rounded up.

DataNodes number

Suppose a DataNode has a JBOD of 12 disks, each disk worth of 4 TB. To calculate the number of DataNodes required for the 845 TB storage, use the following formula: $\frac{845\text{TB}}{4\text{TB} \cdot 12} = 18$.

NOTE

Generally it is not necessary to build the whole cluster at once. For example, you can start with 25% of total nodes and scale up the cluster to 100% as data grows.

CPU and RAM

As mentioned in recommendations, the following CPU and RAM parameters fit for the pilot cluster:

- 8 CPU cores;
- 128 GB RAM.

Custom Java settings

Sergei Tikhomirov, Elena Kostyuchenko

Contents

To Table of Contents

Install Java

Change Java settings in Trino

This article describes the installation of a custom Java for a specific service. In this case, JDK version 23.0.2 is installed for Trino specifically.

Install Java

1. Download the Java version you need on all hosts:

```
$ wget https://download.oracle.com/java/23/archive/jdk-23.0.2_linux-x64_bin.tar.gz
```

2. Create a directory for custom Java on all hosts:

```
$ mkdir -p /tmp/jvm-example
```

3. Unpack the package by the created path on all hosts:

```
$ sudo tar -xzf jdk-23.0.2_linux-x64_bin.tar.gz -C /tmp/jvm-example
```

4. Check Java installation by running the command below to determine that the required Java version is used:

```
$ /tmp/jvm-example/jdk-23.0.2/bin/java --version
```

   The output:

```
java 23.0.2 2025-01-21
Java(TM) SE Runtime Environment (build 23.0.2+7-58)
Java HotSpot(TM) 64-Bit Server VM (build 23.0.2+7-58, mixed mode, sharing)
```

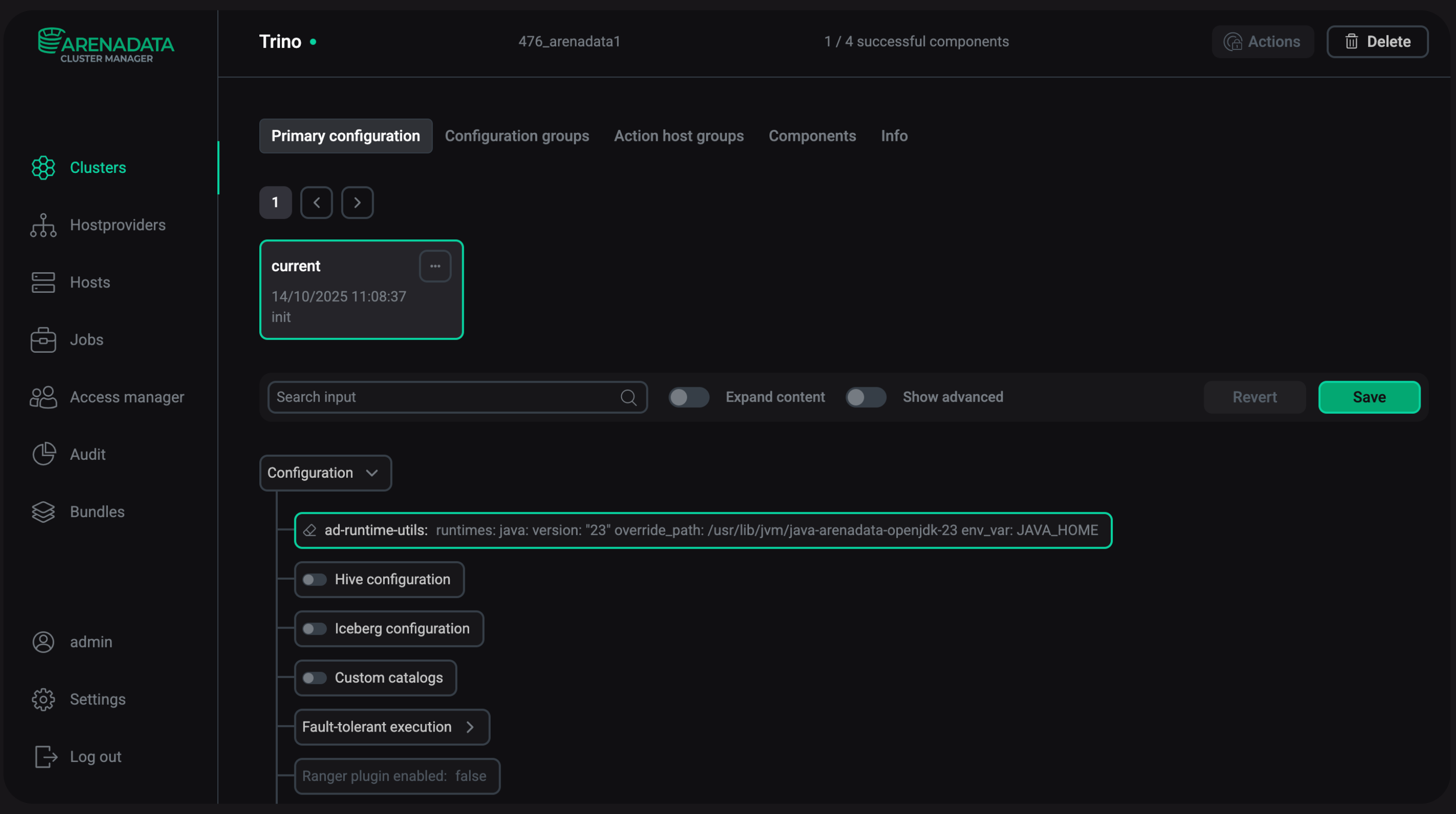
Change Java settings in Trino

Go to ADCM interface and follow the steps below:

1. On the ***Clusters*** page, select the desired cluster.
2. Go to the ***Services*** tab and click at ***Trino***.
3. Find and edit the ***ad-runtime-utils*** parameter as follows:

```
runtimes:
  java:
    version: "23.0.2"
    override_path: /tmp/jvm-example
    env_var: JAVA_HOME
```

4. Click ***Apply*** and confirm changes to cluster configuration by clicking ***Save***.
5. In the ***Actions*** drop-down menu, select ***Restart***, make sure the ***Apply configs from ADCM*** option is set to `true`, and click ***Run***.



ARENADATA CLUSTER MANAGER
Trino
476_arenadata1
Actions
Delete
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
Action host groups
Components
Info
1
Search input
Expand content
Show advanced
Revert
Save
Configuration
Hive configuration
Iceberg configuration
Custom catalogs
Fault-tolerant execution

1 / 4 successful components

current
14/10/2025 11:08:37
init

ad-runtime-utils: runtimes: java: version: "23" override_path: /usr/lib/jvm/java-arenadata-openjdk-23 env_var: JAVA_HOME

Ranger plugin enabled: false

ad-runtime-utils parameter in Trino configuration

Airflow

Daria Barysheva

Contents

To Table of Contents

- Data model
  - DAG (Directed acyclic graph)
  - Task
  - Operator
- Idempotence
- Architecture

Airflow is a platform that allows to develop, plan, run, and monitor complex workflows. It fits perfectly with ETL/ELT processes and also can be useful if you need to periodically run any processes and monitor their execution. In order to describe processes, Airflow uses the Python programming language.

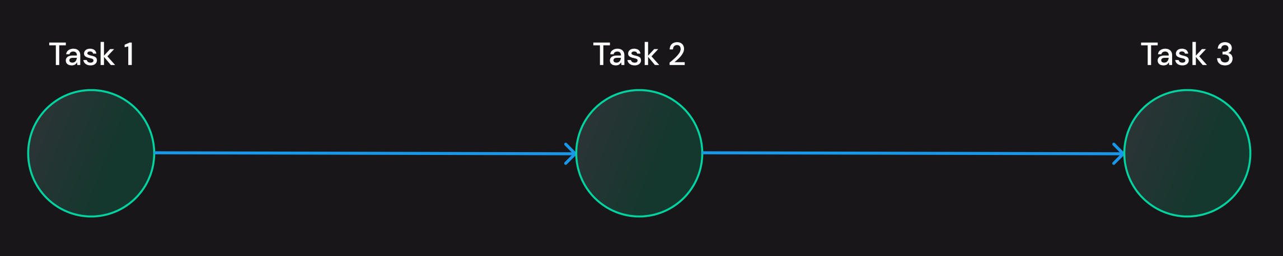
Data model

The Airflow data model uses the following basic objects.

DAG (Directed acyclic graph)

It is the core concept of Airflow representing a semantic association of Tasks that should be performed in the strictly defined order — according to the specified schedule. Visually, the DAG looks like a directed acyclic graph, i.e. a graph without cyclic dependencies.

The simple DAG shown below consists of three Tasks. You can see that it defines the order of running Tasks and dependencies between them. Using Airflow you can also configure the scheduler for this DAG, actions in case of failure, number of retries, etc. The DAG does not care about what happens with data — this function is performed by Tasks themselves.



A simple DAG

Task

A task is a DAG's node that describes operations to be applied to the data. For example, loading data from external sources, aggregating and indexing, removing duplicates, and other typical ETL operations.

Operator

This is a template for completing Tasks. While Tasks describe what actions to perform with data, Operators define how to perform these actions at the code level — using Python functions, Bash scripts, etc.

Airflow supports a lot of built-in core Operators. Besides that, you can use a huge set of community provider packages. Also, you can create your own Operators by extending the `BaseOperator` class.

A separate type of Operators — Sensors — allow users to define a reaction to certain events: the occurrence of a specific time, loading some data, starting of other DAGs/Tasks, etc.

Examples of useful Operators are listed below.

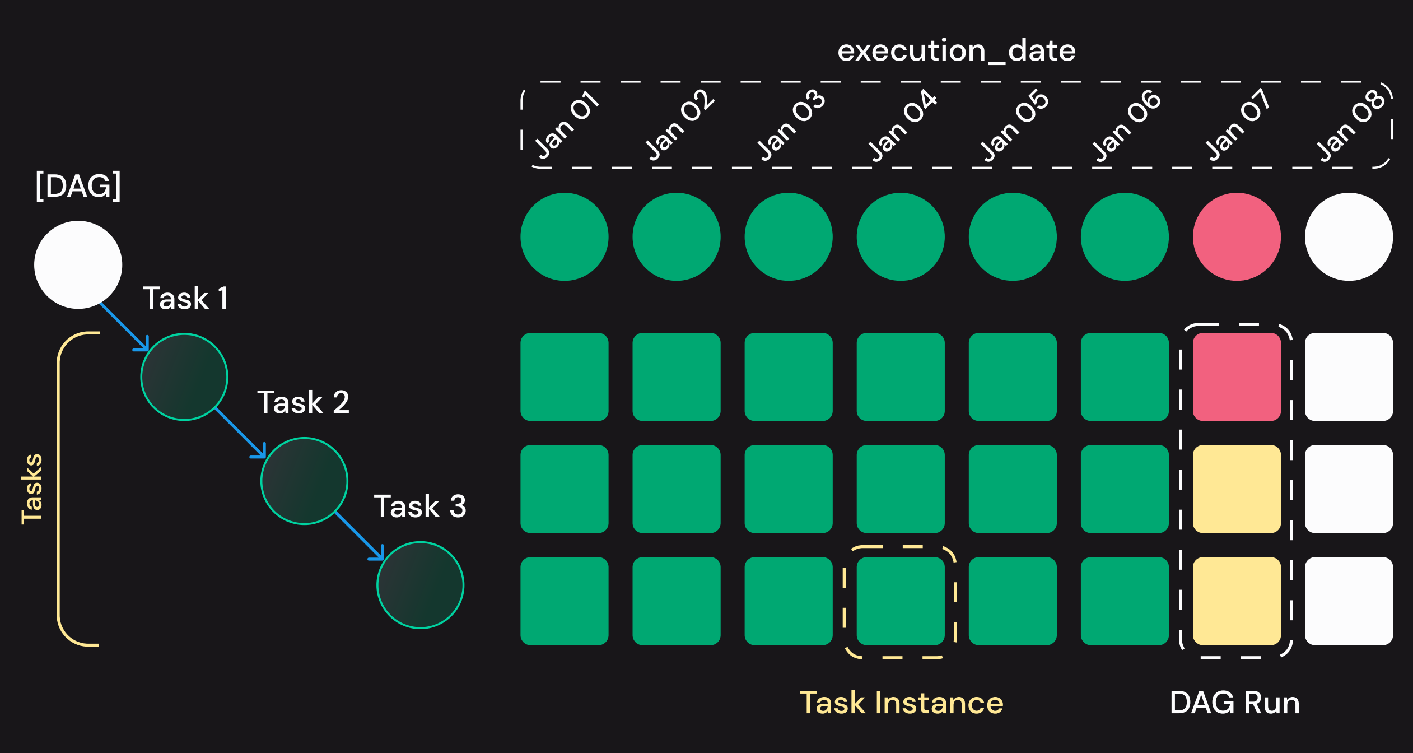
Airflow Operators

Operator	Description
PythonOperator	Execution of the Python code
BranchPythonOperator	Execution of the code depending on the specified conditions
BashOperator	Running Bash scripts
SimpleHttpOperator	Sending HTTP requests
PostgresOperator	Sending queries to a PostgreSQL database
MySqlOperator	Sending queries to a MySQL database
SqlSensor	Verifying the SQL query execution
DockerOperator	Running a Docker container for the Task
KubernetesPodOperator	Creating a separate Kubernetes pod for the Task
EmailOperator	Sending Emails
DummyOperator	The "empty" Operator that is used for grouping Tasks

Idempotence

One of the most important concepts underlying AirFlow is storing information about every DAG's start time according to the specified schedule. For example, if you specify that the DAG should start at 01.01.2022 00:00:00 once a day, AirFlow will store information about running of this DAG using the following timestamps: 01.01.2022 00:00:00, 02.01.2022 00:00:00, 03.01.2022 00:00:00, and so on. The timestamps used in this example are called execution_date, the corresponding DAG instances — DAG Runs, and the Task entities associated with specific DAG Runs — Task Instances.

This concept is very important for maintaining the idempotence: start or restart of the Task with the same dataset for some date in the past allows you to reproduce the results obtained earlier. In addition, it is possible to run Tasks of one DAG for different timestamps simultaneously.



The concept of Task Instances and Dag Runs

Architecture

Airflow architecture is based on the following components:

- Web Server. Provides a convenient user interface that allows to run and monitor DAGs and Tasks, track their schedule, execution status, and so on.
- Metadata Database. A shared database that is used by Scheduler, Executor, and Web Server components to store the system metadata: global variables, data source connection settings, statuses of Task Instances and DAG Runs, etc. As this repository is based on the SQLAlchemy library, it requires the installation of a compatible database. For example, MySQL or PostgreSQL.
- Scheduler. A service responsible for planning in Airflow. It handles both triggering scheduled workflows and submitting Tasks to the Executor for running.
  The Scheduler analyzes the results of DAGs parsing and checks the existence of Tasks ready to run regularly (by default, once a minute). When the predefined conditions for Tasks running are met, the scheduler initializes Task Instances. To perform active Tasks, the scheduler uses the Executor specified in the settings.

  For certain Metadata Database versions (PostgreSQL 9.6+ and MySQL 8+), using of several schedulers is supported. It helps to provide the maximum fault tolerance.
- Executor. A mechanism that runs Task Instances. It works in conjunction with the Scheduler. In the default Airflow installation, the Executor runs everything inside the Scheduler (within the same process). But for production needs it is recommended to use Executors that push Tasks execution out to Workers.

Executor types

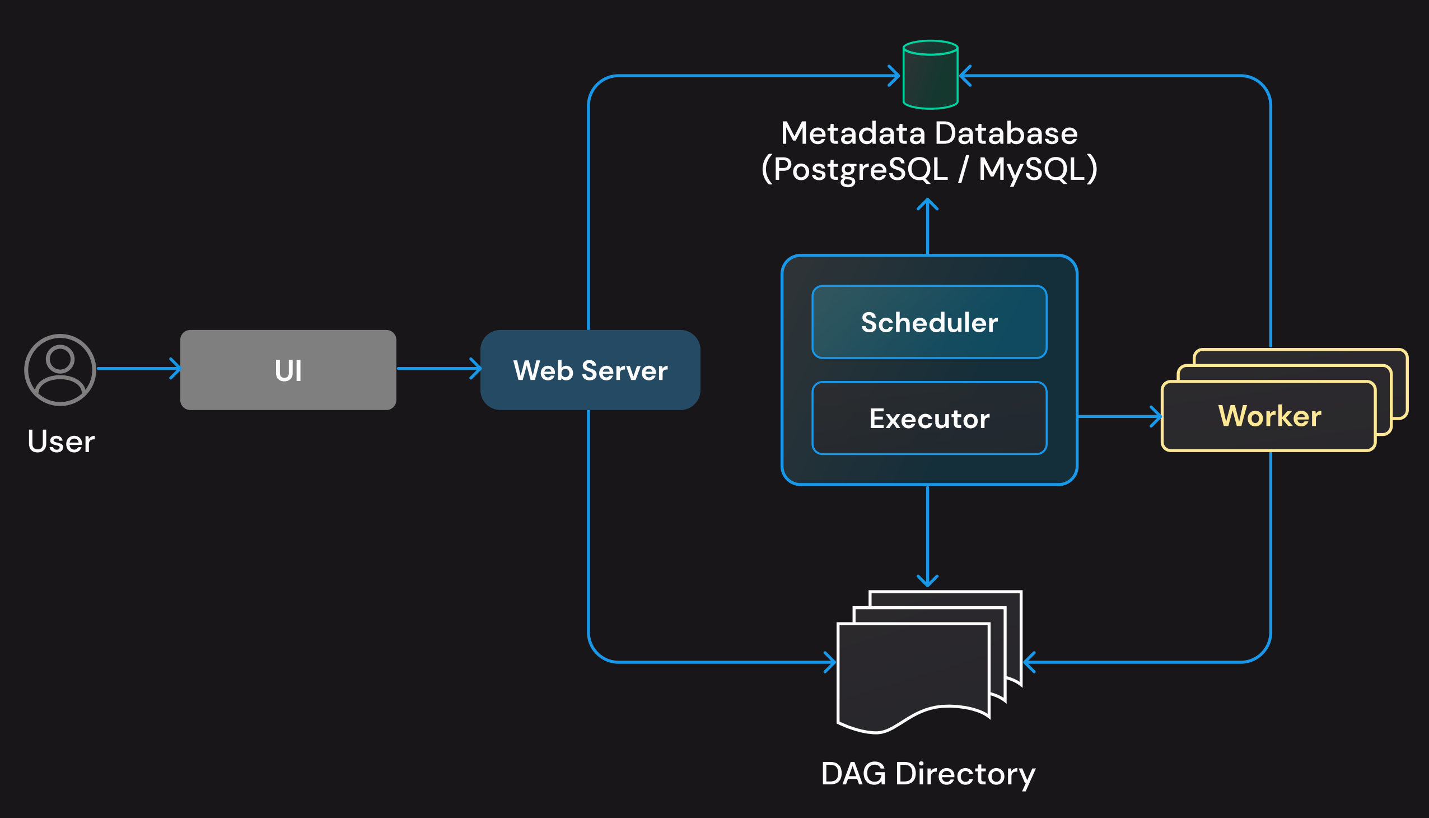
Type	Description
SequentialExecutor	Launches the Tasks sequentially and suspends the Scheduler for the duration of their execution. That is why this Executor is recommended only for testing — it is not suitable for a production environment
LocalExecutor	Starts a new child process for each Task, that allows to process multiple Tasks in parallel. It perfectly simulates a production environment in a test environment, but it is also not recommended for real use since the low fault tolerance: in case of a failure on the machine where Executor is running, the Task is not transferred to other nodes
CeleryExecutor	Allows to have multiple Workers running on different nodes. The executor uses Celery and requires additional installation of the message broker. For example, Redis or RabbitMQ. When the load increases, it is enough to connect an additional Worker. If any Worker fails, its work is transferred to the rest of the nodes
DaskExecutor	It is similar to CeleryExecutor, but instead of the Celery for parallel computing it uses the Dask library
KubernetesExecutor	Launches a new Worker on a separate pod in Kubernetes for each Task Instance
CeleryKubernetesExecutor	Allows to run CeleryExecutor and KubernetesExecutor simultaneously. The specific type is selected depending on the status of the Tasks queue
DebugExecutor	Used to run and debug pipelines from the IDE

- Worker. A separate process in which Tasks are executed. The location of the Worker is determined by the selected type of the Executor.
- DAG Directory. A folder of DAG files that can be read by the Scheduler, Executor, and any Worker that the Executor has.

IMPORTANT

All DAGs in the form of python-scripts should be located in the DAG Directory. Web UI is used only for monitoring DAGs, not for adding or editing.

The high-level architecture view of Airflow is shown below. Depending on the Executor type, this scheme can contain additional components: for example, the Message Queue between the Executor and its Workers for the CeleryExecutor. However, in such particular cases you can still think of an Executor and its Workers as a single logical component that handles the Tasks execution.



Airflow architecture

NOTE

For more details about Airflow concepts, see AirFlow documentation.

Flink overview

Konstantin Alpashkin

Contents

To Table of Contents

- Architecture
  - Flink Client
  - JobManager
  - TaskManager
- Concepts
- Flink History Server
  - History Server REST API

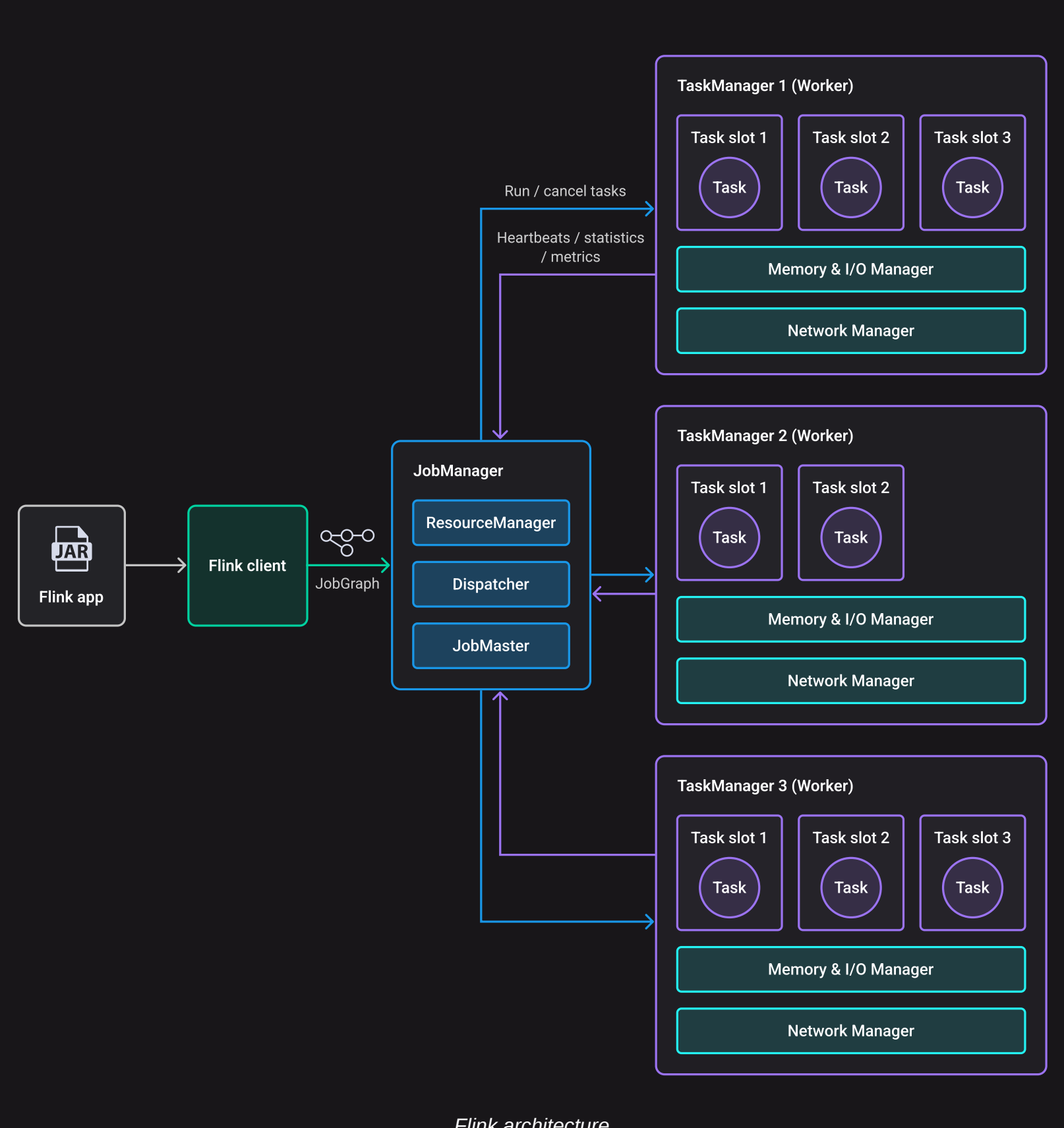
The Flink service provides an engine for stream and batch data processing. Being similar to Spark in some aspects, Flink offers stateful and stateless computations, an SQL abstraction layer, high availability support, and robust clustering capabilities. However, Flink was initially designed as a stream-first framework that supports bounded and unbounded data streams, making it versatile for various applications. Flink's applicability examples are: real-time analytics, event-driven applications, pattern/anomaly/fraud detection apps, high-load ETL pipelines, etc.

Flink provides the following capabilities:

- Unified API for stream and batch processing flows. Although Flink was primarily designed for stream processing, it can also handle batch processing tasks.
- Stateful computations. Flink supports stateful processing, remembering the data at various processing stages. Flink can guarantee exactly-once processing semantics, ensuring that data is processed once and only once, even in case of failures or restarts.
- Windowing and time-based operation. Flink has built-in support for various types of windows, which are a key tool for processing unbounded streams. The windows allow grouping and aggregating data over arbitrary time intervals. Flink also supports running aggregations like `SUM`, `AVG`, `COUNT` in scope of a window based on the event time or processing time.
- Fault tolerance and recovery. Flink provides powerful checkpointing capabilities to ensure fault tolerance. With checkpoints, the system periodically persists its state, being able to recover from a successful checkpoint in case of a failure. A similar savepoints mechanism allows triggering manual snapshots of Flink jobs, allowing you to pause, update, and restart jobs while preserving the state.
- SQL over streams. Flink provides an SQL abstraction over incoming data streams, allowing you to run SQL queries against a data stream as if it were a regular table. You can run operations like filters, joins, aggregations, and windowing on data streams directly in SQL.

Architecture

The high-level Flink architecture is shown below.



Flink architecture

The main diagram components are highlighted below.

Flink Client

Compiles Flink application code into a logical graph (also called a Dataflow graph or JobGraph) and submits it to JobManager. It is not a part of the Flink runtime. There are several implementations that can act as a client in a Flink cluster:

- link C .
- REST endpoint. Flink can receive JARs with tasks within a REST request payload.
- SQL client. Allows running Flink jobs using a built-in CLI interface for SQL.
- Java/Python client applications that programmatically submit new jobs using Flink API.

JobManager

The key component that coordinates the lifecycle of Flink applications. It manages task scheduling, monitors the state of tasks, coordinates checkpoints, handles recovery, and so on. JobManager is a single entity per cluster that directly controls one or more TaskManagers.

The JobManager process comprises the following components:

- Resource Manager. Responsible for resource provisioning in a Flink cluster. It manages task slots, which are the unit of resource scheduling in Flink.
- JobMaster. A per-job object responsible for managing the execution of a single Dataflow graph. Multiple jobs can run in parallel in a Flink cluster, each having its own JobMaster.
- Dispatcher. Creates a new JobMaster for each submitted job. Exposes REST endpoints to receive Flink application JARs and also provides the link web with information about job execution.

TaskManager

A JVM process, capable of running one or more subtasks in separate threads. A Flink cluster includes one or more TaskManagers. TaskManagers buffer data streams and exchange data between each other. The smallest unit of resource scheduling in a TaskManager is a task slot.

> TIP
>
> In ADH, you can submit Flink jobs to a YARN cluster, which automatically creates containers for Flink JobManagers/TaskManagers on hosts controlled by YARN. For more information about supported modes and usage examples, see un Flink on YAR .

Concepts

The following are major Flink-specific concepts frequently used in related articles. For more details, see Flink documentation.

- Bounded and unbounded streams

  A bounded data stream is a stream of data that has a well-defined, finite size. For example, reading a CSV file and processing its records with Flink DataStream API is an example of bounded processing. This mode assumes that the entire dataset can be ingested (possibly sorted or pre-processed) before applying any computations.

  An unbounded stream has a starting point but may never end, continuously delivering data records at arbitrary intervals. Examples of unbounded data streams: a Kafka topic, a web socket emitting JSON records, or a digital interface of a sensor.

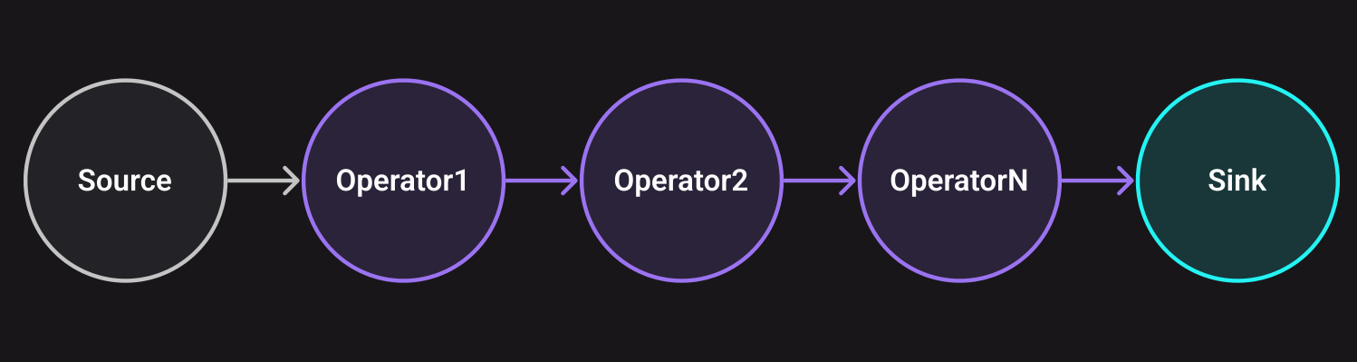
- State and stateful processing

  State refers to the data that Flink retains between individual processing stages (for example, between two aggregation operators). The examples of stateful processing: searching for patterns in the input data stream, aggregating events within a specific time frame (minute/hour/date), etc. Stateful processing assumes additional expenses on managing and storing the state data. However, it is the state data that allows complex computations like aggregations, deduplication, pattern matching, and window-based computations.

  In Flink, the state data is managed by the component called state backend. Choosing an appropriate state backend implementation that best fits a particular Flink application can help minimize state data access costs.

- Sources and sinks

  A typical Flink application dealing with data streams has a structure presented below. The following is a simplified representation since Flink can automatically create several source/sink/operator instances in JVM, depending on many factors, such as parallelism settings, enabled optimization mechanisms, etc.



Sources and sinks

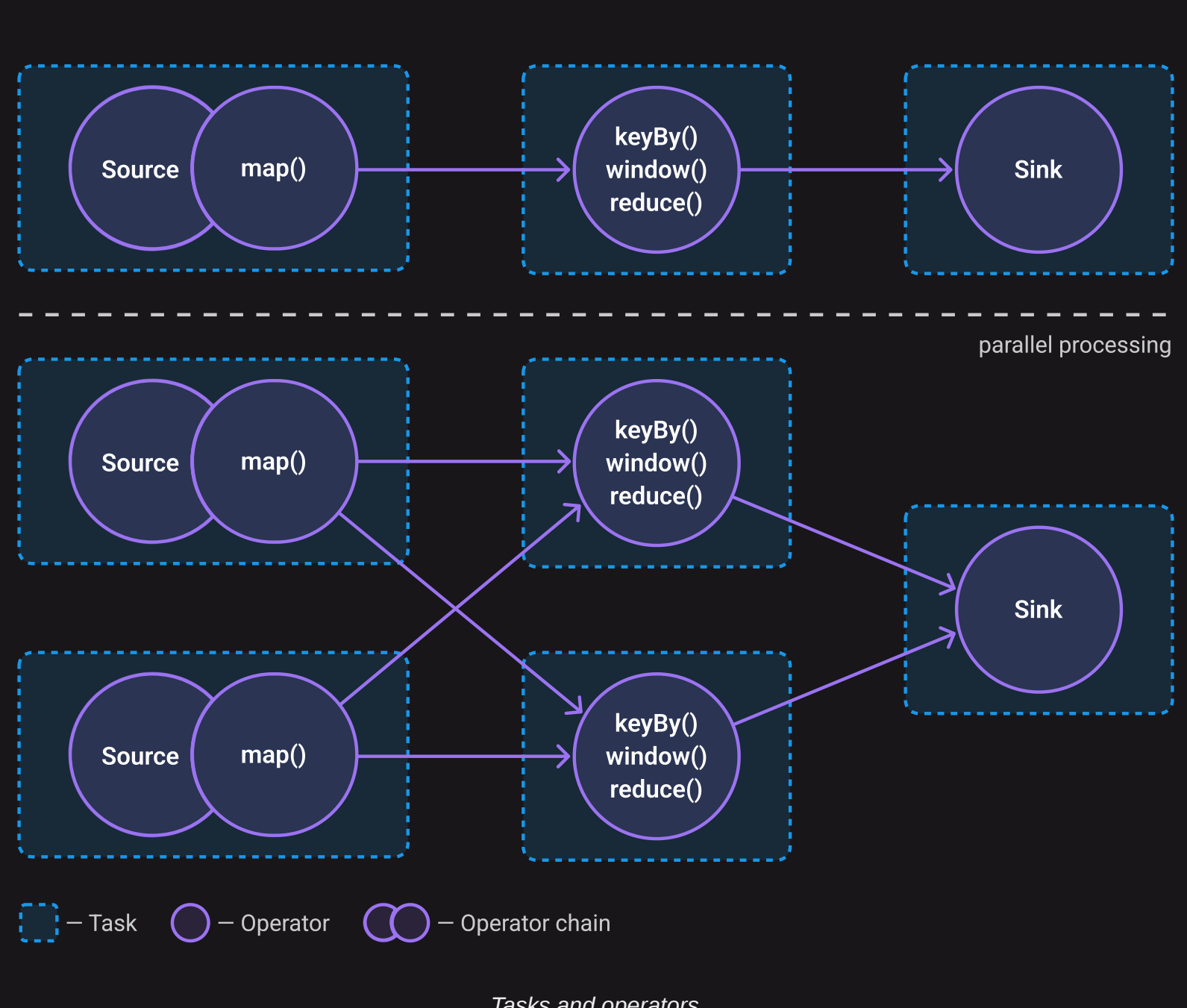
  The major flow components are as follows:

  - Source. The entry point of a Flink job. It reads data from an external system, such as a message queue, database, or socket, and creates a DataStream entity, which is further processed by operators. Flink comes with in-built source connectors for Kafka, JDBC, file systems, etc.
  - perator apply transformations on DataStreams.
  - Sink is the endpoint of a Flink application flow. Data sinks consume DataStreams and forward their data to files, sockets, queues, and other external systems.

- Tasks and operators

  Operators consume one or more DataStreams, apply transformations, and output new DataStream(s). Examples of transformation operators are: `map()`, `flatMap()`, `keyBy()`.

  A task is a basic unit of work. For each instance of an operator that runs in parallel, it takes one task to complete its job. For example, an operator with parallelism set to `3` will have each of its instances executed by three separate tasks.



Tasks and operators

Flink History Server

The Flink service comes with the Flink History Server component that provides a eb with statistics about completed Flink applications.

> NOTE
>
> The Flink History Server component is available starting ADH .0. .

The statistics about completed jobs are generated by a JobManager and are stored in a configurable location (defaults to */apps/flink/completed-jobs* in HDFS). The History Server scans this location periodically and renders information about completed jobs in the web UI. The archive directory locations and scan interval can be configured using Flink service configuration parameters.

History Server REST API

Apart from the web UI, Flink History Server exposes a REST API, which can be used to fetch information about completed jobs. The REST endpoints do not require authentication, accept only GET requests, and can be requested as follows:

```
GET http://<history_server_host>:<port>/<endpoint_path> HTTP/1.1
```

where:

- `<history_server_host>` is the name or IP-address of the host with the Flink History Server component.
- `<port>` is the port number to accept REST requests. Defaults to `8081` and can be changed using the `historyserver.web.port` parameter of the Flink service in ADCM.

The following endpoints are available:

GET [BASE_URL]/config

Returns general information about the Flink cluster.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/config'
```

Sample response:

```
{
    "refresh-interval": 10000,
    "timezone-name": "Coordinated Universal Time",
    "timezone-offset": 0,
    "flink-version": "1.20.1",
    "flink-revision": "ae77a60 @ 2025-04-28T10:41:27+02:00",
    "features": {
        "web-submit": false,
        "web-cancel": false,
        "web-rescale": false,
        "web-history": true
    }
}
```

GET [BASE_URL]/jobs/overview

Returns a list of submitted Flink jobs.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/jobs/overview'
```

Sample response:

```
{
    "jobs": [
        {
            "jid": "547cc23f3e85744bffbfcafe7c06471a",
            "name": "Flink Java Job at Sun Jun 15 23:25:44 UTC 2025",
            "start-time": 1750029945589,
            "end-time": 1750029946616,
            "duration": 1027,
            "state": "FINISHED",
            "last-modification": 1750029946616,
            "tasks": {
                "running": 0,
                "canceling": 0,
                "canceled": 0,
                "total": 3,
                "created": 0,
                "scheduled": 0,
                "deploying": 0,
                "reconciling": 0,
                "finished": 3,
                "initializing": 0,
                "failed": 0
            }
        }
    ]
}
```

GET [BASE_URL]/jobs/<job_id>

Returns details on a specific Flink job, including the job execution plan, vertices, subtasks, etc.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/jobs/547cc23f3e85744bffbfcafe7c06471a'
```

Sample response:

```
{
    "jid": "547cc23f3e85744bffbfcafe7c06471a",
    "name": "Flink Java Job at Sun Jun 15 23:25:44 UTC 2025",
    "isStoppable": false,
    "state": "FINISHED",
    "job-type": "BATCH",
    "start-time": 1750029945589,
    "end-time": 1750029946616,
    "duration": 1027,
    "maxParallelism": -1,
    "now": 1750029946692,
    "timestamps": {
        "FAILED": 0,
        "SUSPENDED": 0,
        "CANCELLING": 0,
        "INITIALIZING": 1750029945589,
        "FAILING": 0,
        "CANCELED": 0,
        "RUNNING": 1750029946033,
        "FINISHED": 1750029946616,
        "CREATED": 1750029945752,
        "RESTARTING": 0,
        "RECONCILING": 0
    },
    "vertices": [
        ...
    ],
    "status-counts": {
        "INITIALIZING": 0,
        "CANCELING": 0,
        "DEPLOYING": 0,
        "FAILED": 0,
        "SCHEDULED": 0,
        "RUNNING": 0,
        "RECONCILING": 0,
        "FINISHED": 3,
        "CREATED": 0,
        "CANCELED": 0
    },
    "plan": {
        ...
    }
}
```

GET [BASE_URL]/jobs/<job_id>/config

Returns an execution configuration for a given Flink job.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/jobs/547cc23f3e85744bffbfcafe7c06471a/config'
```

Sample response:

```
{
    "jid": "547cc23f3e85744bffbfcafe7c06471a",
    "name": "Flink Java Job at Sun Jun 15 23:25:44 UTC 2025",
    "execution-config": {
        "execution-mode": "PIPELINED",
        "restart-strategy": "Cluster level default restart strategy",
        "job-parallelism": 1,
        "object-reuse-mode": false,
        "user-config": {}
    }
}
```

GET [BASE_URL]/jobs/<job_id>/vertices/<vertex_id>

Returns details about a particular vertex of a Flink job.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.
internal:8082/jobs/547cc23f3e85744bffbfcafe7c06471a/vertices/33b1d60e4c0d4656355755296d6794e0'
```

Sample response:

```
{
    "id": "33b1d60e4c0d4656355755296d6794e0",
    "name": "CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.
CollectionInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:97)) -> Combine (SUM(1), at main(WordCount.java:100)",
    "parallelism": 1,
    "maxParallelism": 1,
    "now": 1750029946787,
    "subtasks": [
        {
            ...
        }
    ],
    "aggregated": {
        ...
    }
}
```

GET [BASE_URL]/jobs/<job_id>/exceptions

Returns the exceptions raised during the job execution.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/jobs/57a9afc1904b14f808c80378ee20ffeb/exceptions'
```

Sample response:

```
{
    "root-exception": null,
    "timestamp": null,
    "all-exceptions": [],
    "truncated": false,
    "exceptionHistory": {
        "entries": [],
        "truncated": false
    }
}
```

GET [BASE_URL]/jobs/<job_id>/accumulators

Returns a list of accumulator objects involved into the processing of the given the job.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/jobs/57a9afc1904b14f808c80378ee20ffeb/accumulators'
```

Sample response:

```
{
    "job-accumulators": [],
    "user-task-accumulators": [
        {
            "name": "85d49c256c7a7e0370ebba1e88e8f199",
            "type": "SerializedListAccumulator",
            "value": "[[B@7b294b0f, [B@2a90f4d2, [B@2a86b8be, ..."
        }
    ],
    "serialized-user-task-accumulators": {
        "85d49c256c7a7e0370ebba1e88e8f199": "rO0ABXNyACVvcmcuYXBhY2hlLmZsaW5rLnV..."
    }
}
```

GET [BASE_URL]/jobs/<job_id>/vertices/<vertex_id>/subtasktimes

Returns a list of Flink subtasks, which were created to complete the job.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.
internal:8082/jobs/547cc23f3e85744bffbfcafe7c06471a/vertices/33b1d60e4c0d4656355755296d6794e0/subtasktimes'
```

Sample response:

```
{
    "id": "33b1d60e4c0d4656355755296d6794e0",
    "name": "CHAIN DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.
CollectionInputFormat)) -> FlatMap (FlatMap at main(WordCount.java:97)) -> Combine (SUM(1), at main(WordCount.java:100)",
    "now": 1750029946723,
    "subtasks": [
        {
            "subtask": 0,
            "host": "ka-adh-3",
            "endpoint": "ka-adh-3.ru-central1.internal:38443",
            "duration": 565,
            "timestamps": {
                "INITIALIZING": 1750029946391,
                "CANCELING": 0,
                "DEPLOYING": 1750029946222,
                "FAILED": 0,
                "SCHEDULED": 1750029946037,
                "RUNNING": 1750029946393,
                "RECONCILING": 0,
                "FINISHED": 1750029946602,
                "CREATED": 1750029945835,
                "CANCELED": 0
            }
        }
    ]
}
```

GET [BASE_URL]/jobs/<job_id>/plan

Returns the execution plan for a given Flink job.

Sample request:

```
$ curl 'http://ka-adh-3.ru-central1.internal:8082/jobs/57a9afc1904b14f808c80378ee20ffeb/plan'
```

Sample response:

```
{
    "plan": {
        "jid": "57a9afc1904b14f808c80378ee20ffeb",
        "name": "Flink Java Job at Sun Jun 15 23:49:15 UTC 2025",
        "type": "BATCH",
        "nodes": [
            {
                "id": "5b6d423820643b538f78784054364b75",
                "parallelism": 1,
                "operator": "Data Sink",
                "operator_strategy": "(none)",
                "description": "DataSink (collect())",
                "inputs": [
                    {
                        "num": 0,
                        "id": "fa8699328ec101fd063912c9734556b7",
                        "ship_strategy": "Forward",
                        "exchange": "pipelined"
                    }
                ],
                "optimizer_properties": {
                    ...
                }
            },
            {
                "id": "fa8699328ec101fd063912c9734556b7",
                "parallelism": 1,
                "operator": "GroupReduce",
                "operator_strategy": "Sorted Group Reduce",
                "description": "Reduce (SUM(1), at main(WordCount.java:100)",
                "inputs": [
                    {
                        "num": 0,
                        "id": "9d782237f8c637c037444a774f60e6e7",
                        "ship_strategy": "Hash Partition on [0]",
                        "local_strategy": "Sort (combining) on [0:ASC]",
                        "exchange": "pipelined"
                    }
                ],
                "optimizer_properties": {
                    ...
                }
            },
            {
                "id": "9d782237f8c637c037444a774f60e6e7",
                "parallelism": 1,
                "operator": "Data Source -> FlatMap -> GroupCombine",
                "operator_strategy": "(none)<br/> -&gt; FlatMap<br/> -&gt; Sorted Combine",
                "description": "DataSource (at getDefaultTextLineDataSet(WordCountData.java:70) (org.apache.flink.api.java.io.
CollectionInputFormat))<br/> -&gt; FlatMap (FlatMap at main(WordCount.java:97))<br/> -&gt; Combine (SUM(1), at main(WordCount.
java:100)",
                "optimizer_properties": {
                    ...
                }
            }
        ]
    }
}
```

HBase architecture

Daria Barysheva

Contents

To Table of Contents

Components
- Region Server
- Master server
- ZooKeeper

Operations with data
- Preprocessing
- Write
- Read

Performance improvements
- Region split
- HFiles compaction

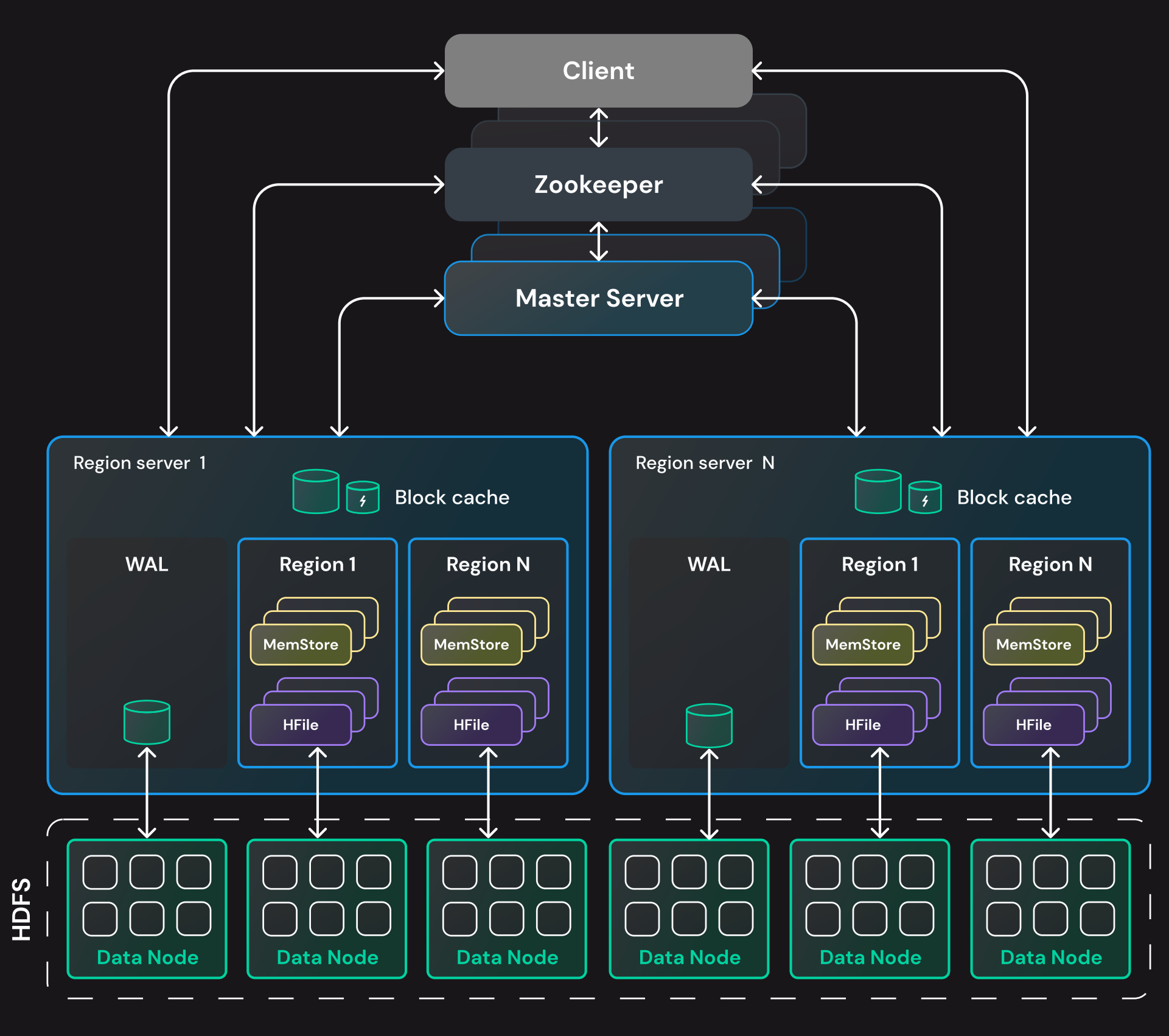
Data recovery

Components

HBase architecture is based on the following components:

- Region Server. It serves one or more Regions — ranges of rows, stored together. Each Region is served only by one Region Server. Region Servers are also called HRegionServers. A Region Server contains multiple components, some of them work on top of HDFS, using it as a persistent data storage.
- Master server. It is a main server responsible for managing an HBase cluster. It is similar to a NameNode in HDFS. A Master server manages distribution of Regions between Region Servers, maintains the registration of Regions, etc. It is also called HMaster. You can deploy several Master servers in your cluster: one active and one or more standby.
- ooKeep . It is a special service designed to manage configurations and synchronization of services. It is used to coordinate actions between HBase services.

The high-level architecture view of HBase is shown below.

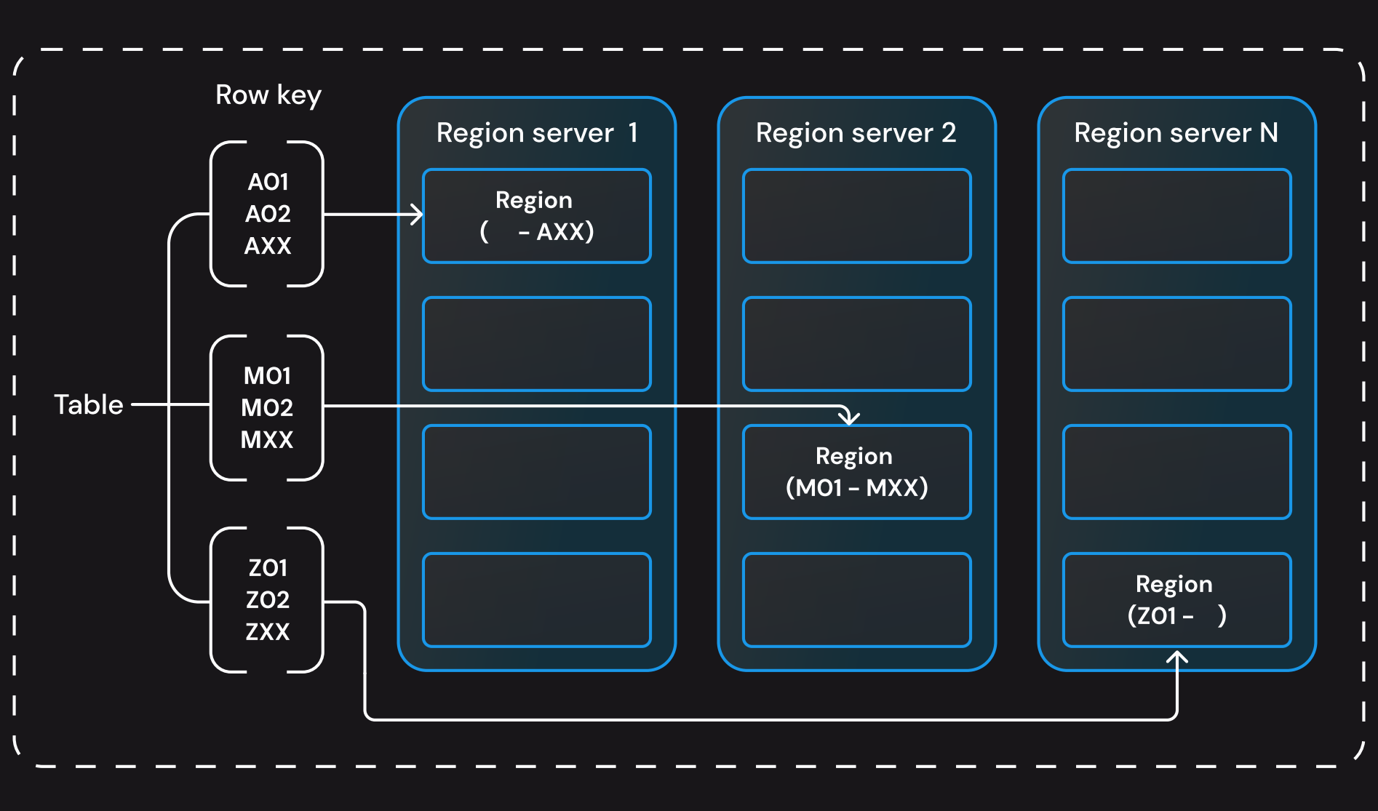


HBase architecture

Region Server

A Region Server maintains several Regions, running on top of HDFS. Each Region contains the rows, having row keys from the same range — assigned to that Region. Rows are sorted alphabetically by their keys. Each Region has a default size of 256 MB, each Region Server can include approximately 1000 Regions.

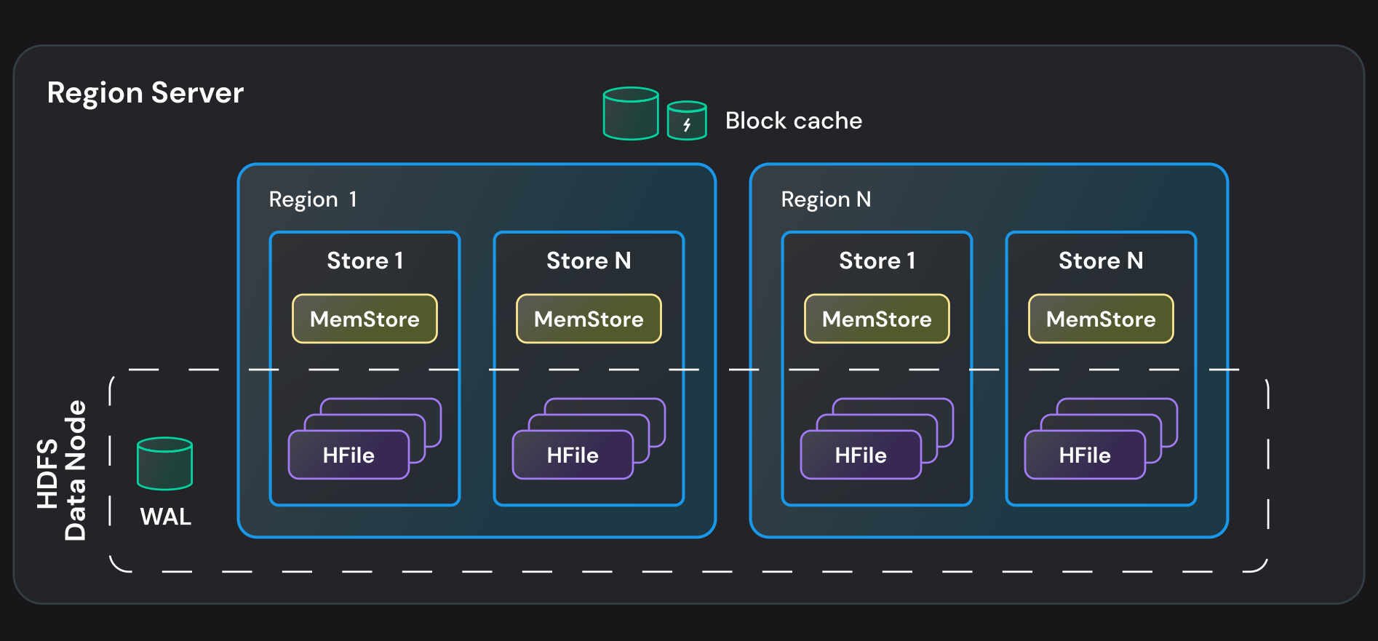
> NOTE
>
> Each Region is defined by the range of row keys. The first Region of the table contains the range, starting with an empty value. The last Region of the table contains the range, ending with an empty value.



The example of Row keys distribution between Regions

A Region Server is responsible for managing the set of its Regions and executing read/write operations on them. Each Region Server consists of the following components:

- HFiles — the main persistent data storage in HBase. These files are stored in HDFS in the special format. The data in each HFile is sorted by Row key. One pair <Region, Column family> corresponds to at least one HFile.
- MemStore — the write buffer, stored in the memory. Since the data stored in HFiles is in the sorted order, updating files for each write operation could be very expensive. Instead of it, new data is written into a special memory area, called MemStore, where it is accumulated for some time. When the MemStore size reaches a certain limit, the data is flushed to a new HFile. The data is sorted in lexicographical order before saving it to HFiles.
  One MemStore corresponds to one Column family. That is why there can be multiple MemStores in one Region, since each Region can contain multiple Column families. The combination of the MemStore and all HFiles, containing data values of the same Column family, is called the Store.
- BlockCache — the read cache that stores frequently used data in memory. It reduces the time of read operations. The data that is not used for a long time is removed from the BlockCache.
- Write Ahead Log (WAL) — a special log file, attached to every Region Server and stored in HDFS. It is used to prevent possible data loss due to MemStore and other failures. All incoming operations are written to this log before their actual implementation. So, WAL stores the new data values, that have not been saved to the permanent storage yet. This allows you to recover data after any failure. WAL is also called HLog.



Region Server components

Master server

A Master server is responsible for monitoring all Region Servers and maintaining the interface for all metadata changes in the HBase cluster. Its main functions are listed below:

- Assigns Regions to Region Servers on startup and re-assigns them during data recovery and load balancing processes.
- Coordinates Region Servers, similarly to how NameNodes manage DataNodes in HDFS.
- Performs DDL operations, providing the interface for creating, deleting, and updating tables.
- Monitors all the instances of Region Servers in the cluster and performs data recovery after their crash — with the help of ZooKeeper (see ata Recover).

ZooKeeper

ZooKeeper acts like a coordinator of other components work in HBase clusters, performing the following functions:

- Checks the availability of Region Servers by receiving regular heartbeats from them. If any Region Server fails to send a heartbeat, Master server gets the notification and runs the automatic ata recovery procedur .
- Provides standby Master servers with information about the status of the active Master server. If the active Master server fails to send a regular heartbeat to ZooKeeper, the inactive Master gets the corresponding notification and becomes active.
- Stores the location of the catalog table hbase:meta (previously called `.META.`), which keeps a list of all Regions in the system. This table stores data about Regions in the form of key/value pairs, where key — the combination <table name, Region start row key, Region ID>; and the value — the full information about this Region and the path to the Region Server, containing it (including host name and port number).

Operations with data

Preprocessing

Whenever a client application makes a write or read request to HBase, first of all it should get the address of the Region Server, which contains the Region with necessary data. That is why any data operation in HBase begins with the following steps:

1. The client retrieves the location of the catalog table hbase:meta from ZooKeeper and stores it in its own cache.
2. From the `hbase:meta` table, the client requests the location of the Region Server that contains the piece of data with defined row keys. The client also keeps this information in the cache.
3. The client requests necessary data from the detected Region Server.

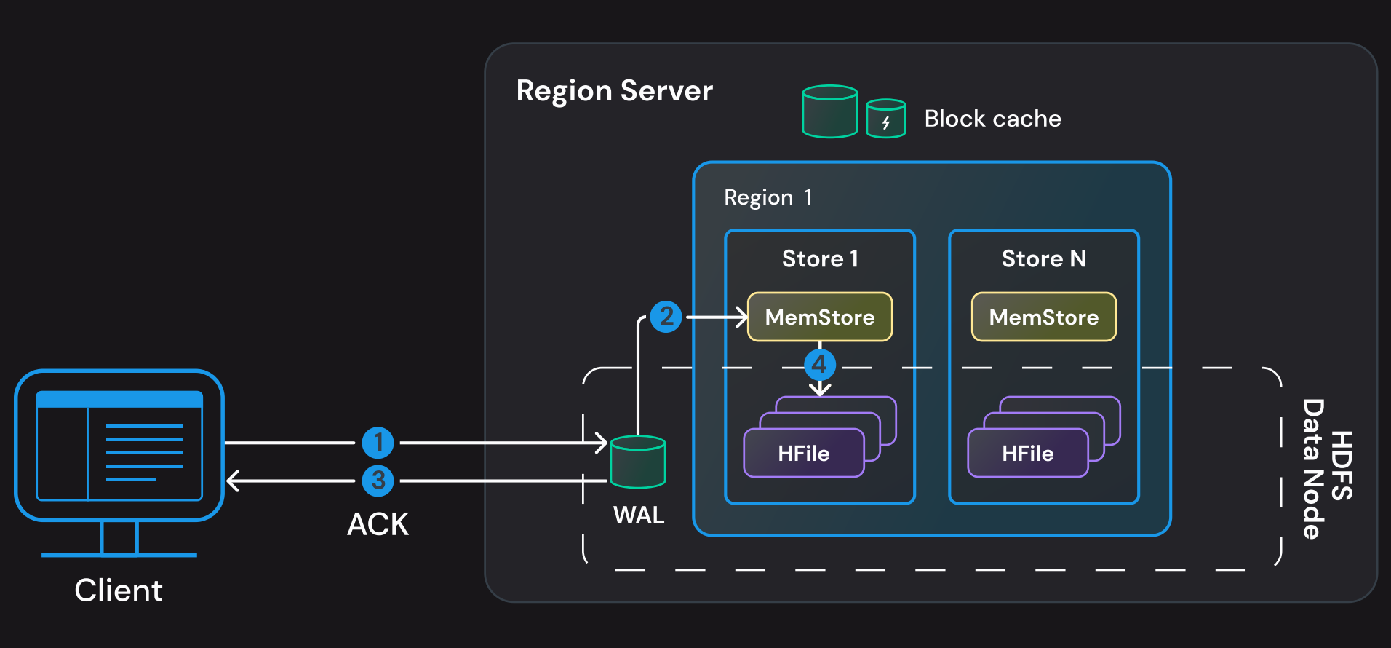
For future references, the client uses its own cache to retrieve the location of the `hbase:meta` table and the path to the Region Server, corresponding to the previously read row keys. The client does not refer to `hbase:meta` table again, until or unless there is a data miss because of the Region shifting or moving. In this case, the client requests the information from `hbase:meta` table again and updates its cache.

Write

The typical steps used for write operations in HBase after detection the right Region Server, are described below:

1. HBase writes data to WAL (Write Ahead Log) that is used to provide fault-tolerance.
2. The data is copied to the MemStore that serves as the RAM of the HDFS DataNode. The choice of MemStore depends on the Column family, present in the request.
3. Once the data is placed to the WAL and MemStore, the client receives the acknowledgment (ACK) — a confirmation that the task has completed.
4. Further, when the MemStore fills up, it flushes all its information to a new HFile in a sorted order.

> NOTE
>
> Client applications do not interact directly with underlying HFiles stored in HDFS.



Typical write sequence

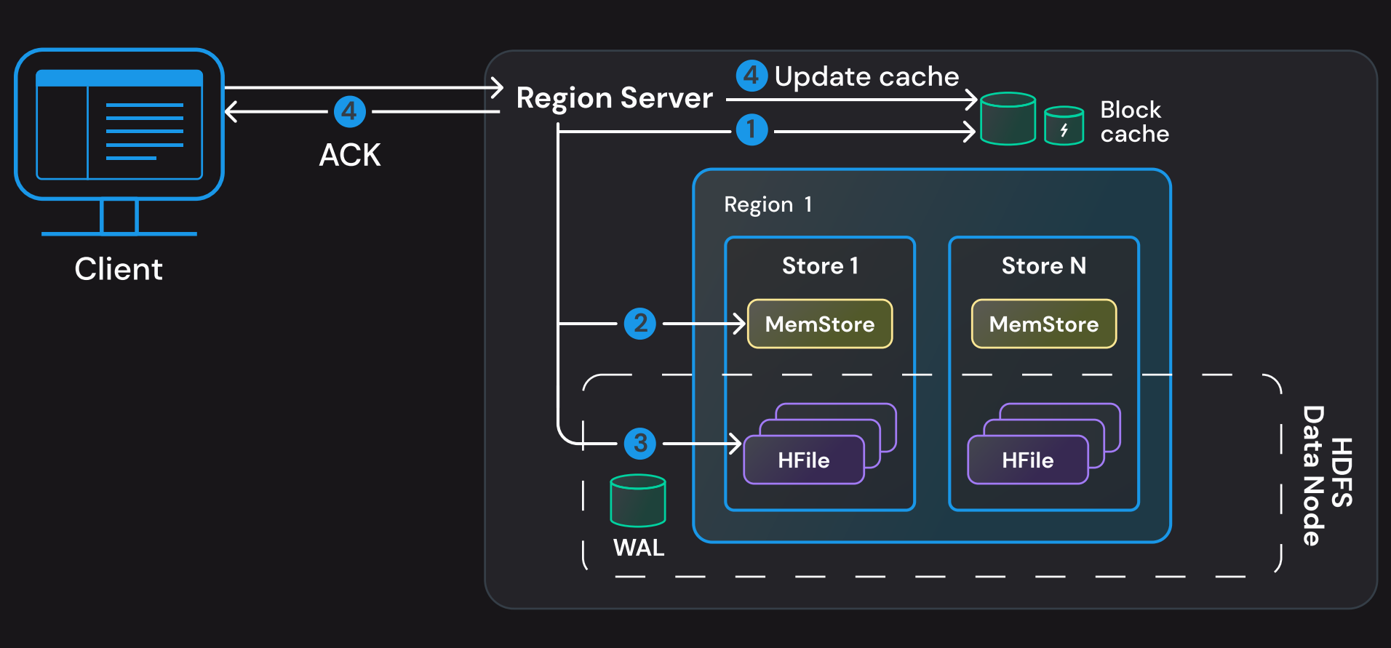
Read

The typical steps, used for read operations in HBase after the right Region Server has been detected, are described below:

1. The required data is searched in the BlockCache — an area, where all the recently read key/value pairs are stored.
2. If the required data is not found in the BlockCache, the search process moves to the MemStore that keeps the most recently written files, not dumped to HFiles yet.
3. If the required data is not found in the MemStore, the search process moves to HFiles, stored in HDFS.

> NOTE
>
> As each HFile contains a MemStore snapshot created at the point when it was flushed, data for the whole requested row can be stored across multiple HFiles. So, to compose the completed row, HBase reads all HFiles that can contain information for that row. Using Bloom filters can facilitate this process.

4. After the search is completed (no matter at what stage it returned the result) the required data is written to the BlockCache and returned to the client along with the acknowledgment (ACK) — the task completion confirmation.



Typical read sequence

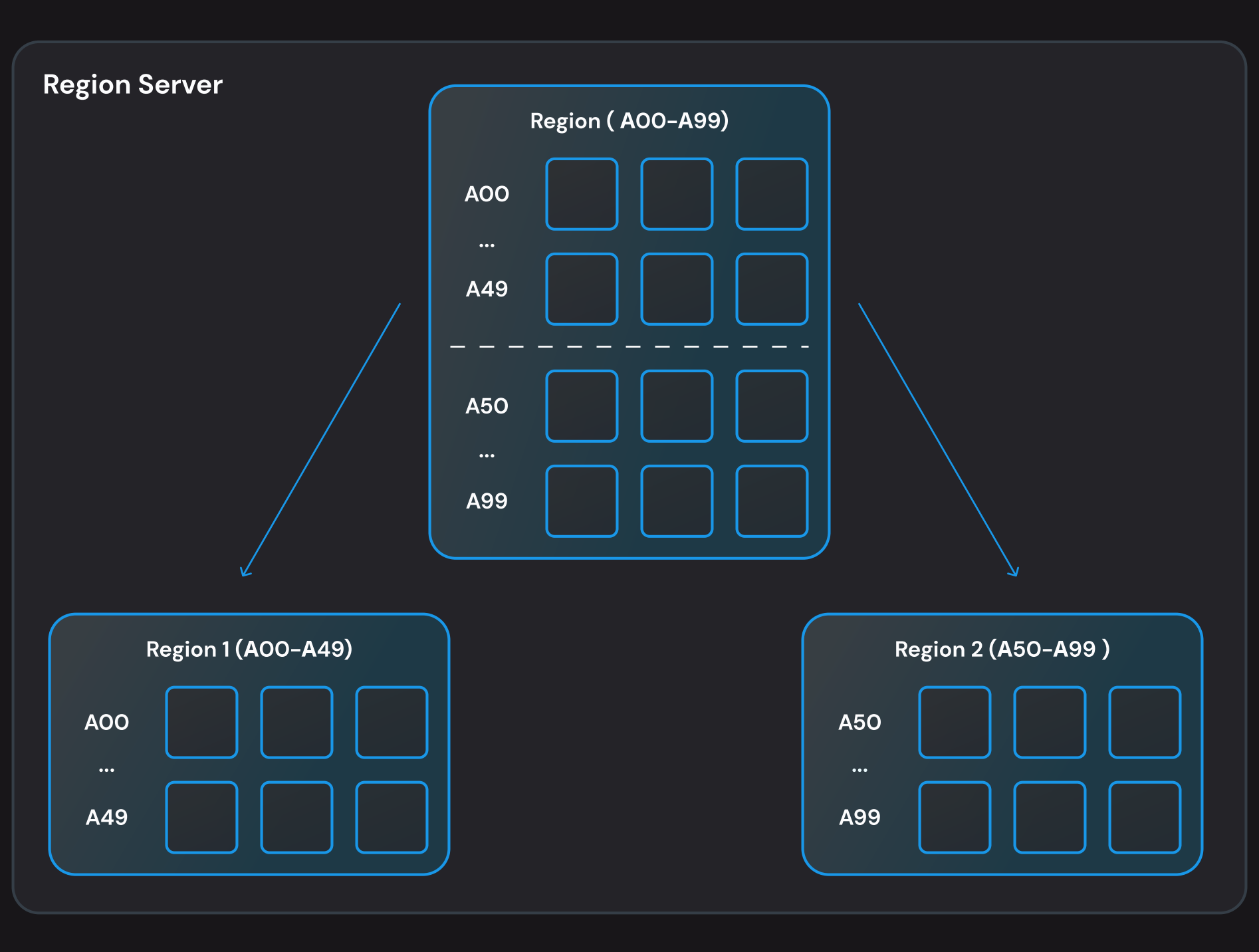
Performance improvements

Region split

When the amount of data in some Region reaches a specific critical size, HBase launches the Region split — a special operation that divides the source Region into two child Regions. Then, it is reported to the Master server.

Initially, new Regions are handled by the same Region Server, until the Master allocates them to a new one — for load balancing.

> TIP
>
> In order to avoid frequent splits of Regions in your database, you can set the boundaries of Regions and increase their maximum size.

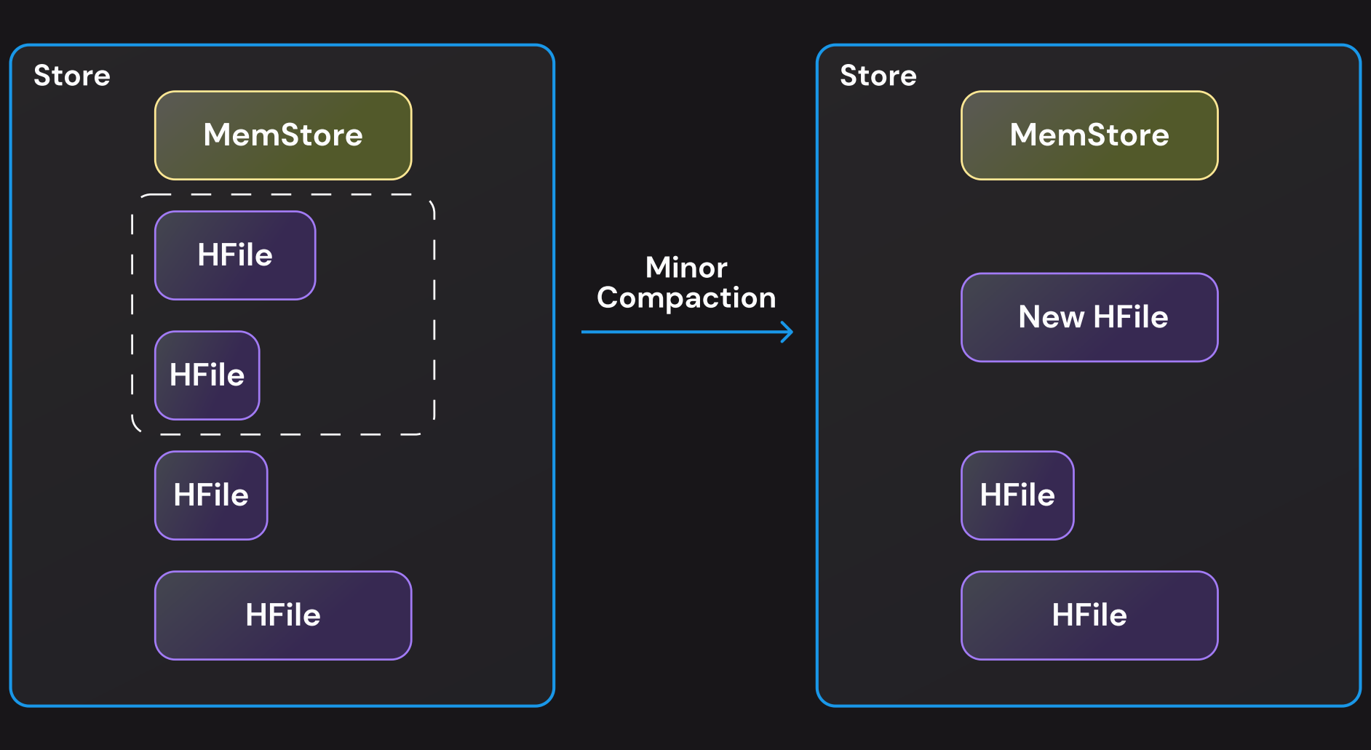


Region split

HFiles compaction

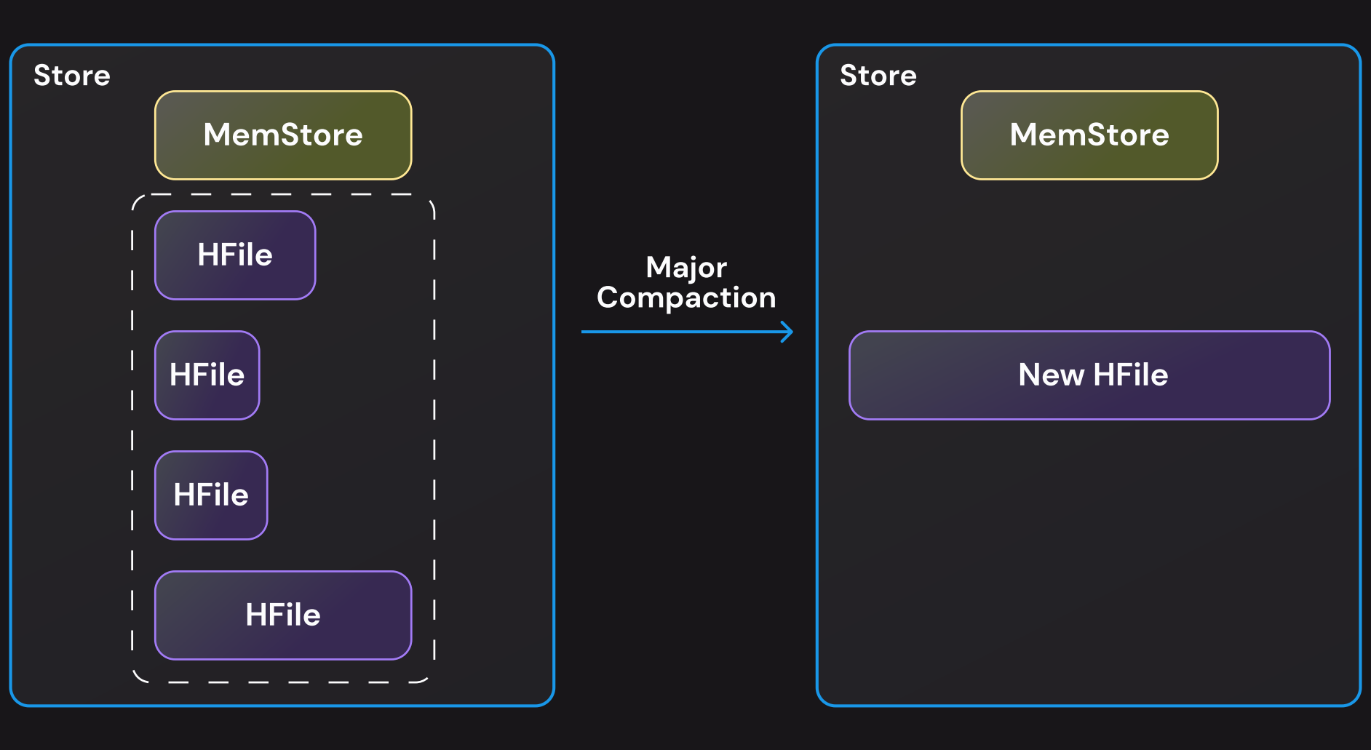
As data in one Region can be stored in several HFiles, HBase periodically merges them together to reduce the number of disk seeks needed for one read operation, and to speed up the work. This operation is called compaction. There are two types of compactions:

- Minor compaction — picks several smaller HFiles and rewrites them to bigger HFiles. This is helpful for storage space optimization. It starts automatically and runs in the background with a low priority, compared to other HBase operations. In Minor compaction files to compact are selected based on heuristic.



Minor compaction

- Major compaction — rewrites all HFiles of the Region Store to one bigger HFile. It also performs physical deletion of the expired data and the data, previously marked with the `tombstone` label. It increases the read performance. Major compaction is started manually or when certain condition are triggered (for example, by a timer). It has a high priority and can significantly slow down the cluster — as input-output disks and network traffic can be congested. So, this type of compaction should be scheduled during low peak load timings.



Major compaction

Data recovery

HBase supports automatic data recovery after different failures. The typical recovery process includes the following steps:

1. If any of the Region Servers fails, ZooKeeper notifies Master server about this.
2. The Master server distributes all the Regions of the crashed Region Server between active Region Servers. While doing this, the Master also distributes the WAL of the crashed Region Server stored in HDFS — to recover the data of crashed MemStores (which are now unavailable).
3. Each Region Server re-processes the WAL to rebuild the MemStore for each Column family from the failed Region Server. Since all data is written to the WAL in the chronological order, re-processing its records means making all necessary changes, that were stored in MemStores.
4. Regions continue to work on active Region Servers.

HBase data model

Daria Barysheva

Contents

To Table of Contents

Basic objects
Workflow overview
Example

HBase is a column-oriented database. Though it stores data in tables with rows and columns (as relational databases do), do not consider these tables as sets of rows with a fixed scheme. Instead of it, it is helpful to consider HBase tables as multi-dimensional maps.

Basic objects

First, consider the data model terminology used in HBase:

- **Table**. A set of multiple rows that can have one or more columns, belonged to the fixed column families. Each table scheme defines only column families.
- **Row**. Consists of a row key and columns with values, associated with them. Rows are sorted alphabetically by the row key, as they are stored.
- **Row key**. A unique row identifier, used as a primary key. Any access to HBase tables uses this key. You should think carefully about the design of your row keys.

TIP

As rows are sorted by the row key, the goal is to store the related rows near each other. But at the same time, keep an eye on the uniformity of the distribution — in order to prevent that some portions of row keys are accessed much more often than others.

- **Column**. Consists of a column family and a column qualifier, which are delimited by the colon character, for example: `cf:column1`, `cf:column2`, etc.
- **Column family**. A logical and physical grouping of columns and their values — often for performance reasons. Column families are fixed at table creation and can't be changed on the fly. Nevertheless, each given table row might not store anything in a given column family.
  A column family can have a set of storage properties, such as: whether its values should be cached in memory, how its data is compressed or its row keys are encoded, time to live (TTL), and maximum number of stored versions for values, etc.

CAUTION

It is not recommended to define more than three column families for each table.

- **Column qualifier**. A column identifier, needed to provide the index for a given piece of data in a column family. Though column families are fixed, column qualifiers are mutable: they can be changed on fly and differ greatly between rows of the same table.
- **Cell**. A combination of a row, a column family, and a column qualifier. It contains a data value and a timestamp that represents the current value version.
- **Timestamp**. A version identifier of a cell value. By default, it represents the time on the RegionServer when this data value was written to HBase. You can also specify a different timestamp value.
- **Value**. A piece of data, stored in HBase with a specific combination: <Table, Row Key, Column Family, Column Qualifier, Timestamp>.

The HBase data model can be remembered as the following key/value match: <Table, Row key, Column family, Column qualifier, Timestamp> → Value

Workflow overview

At the logical level the data in HBase is organized into tables, indexed by the primary row key. You can store an unlimited set of columns for each row key. Columns are organized into column groups, called column families. As a rule, each column family combines the columns with the same pattern of usage and storage.

Columns are used for storing pieces of data, called values. Each column is defined by column family name and its own unique name, called column qualifier. A column can store several versions of one value. Different value versions have different timestamps.

Suppose that there is a table with two column families and two rows identified by their row keys. For each combination of the row key and the column family, you can define different column qualifiers. You also can add new column qualifiers on the fly.

The approximate logical view of such table is shown below (where `ts` is an abbreviation of "timestamp").

Logical view

Row Key	Column Family 1		Column Family 2	
	Column Qualifier 1	Column Qualifier 2	Column Qualifier 3	Column Qualifier 4
Row Key 1	ts1:value1	ts3:value3 ts2:value2	ts6:value6	—
Row Key 2	ts5:value5 ts4:value4	—	—	ts8:value8 ts7:value7

You can see sparse data in the table, shown above, as filling of columns for each row key is optional. You don't even have to fill each column family for your rows. If the column is missing, it does not cause overhead for storing empty values, because at the physical level the data are stored differently, than at the logical one.

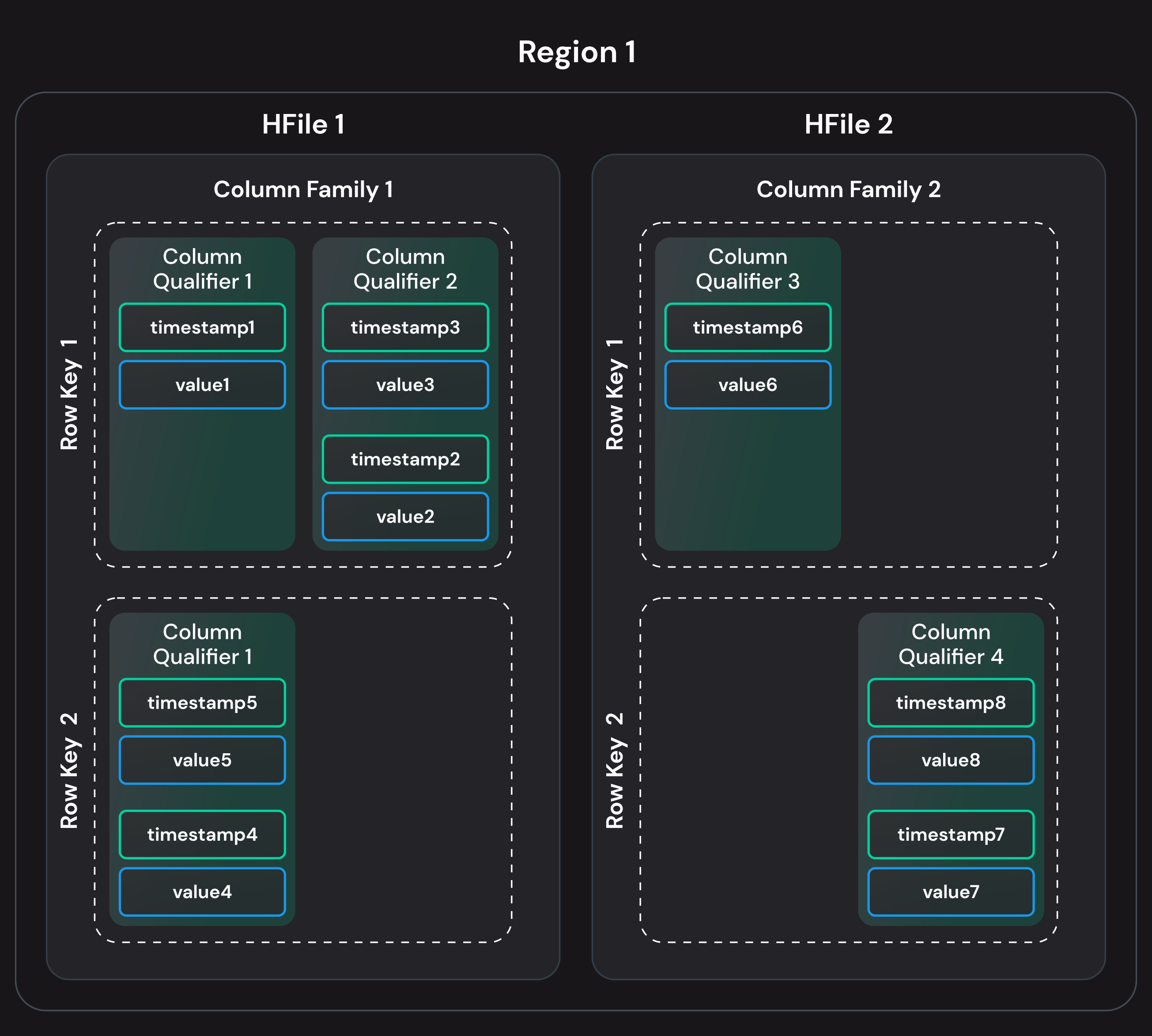
Physically all the data values in HBase are stored in the row key-sorted order. Blocks of nearby row keys form Regions described in the HBase architecture article. The data, corresponding to different column families, are stored separately — in different HFiles. Each column family uses at least one HFile. It allows, if necessary, to read data only from the desired column family. Columns that correspond to the same column family and the same row key are also stored as a list, sorted alphabetically.

Any column can be missing or present for each row key — empty values are never stored in the database.

When you delete a certain column value, it is not physically deleted immediately — it is marked with a special flag `tombstone`. The physical deletion of the data will occur later, while performing the Major compaction operation.

In addition to manual deletion of old values versions, you can configure automatic one. At the column family level, you can define such parameters as time to live (TTL) and the maximum number of stored versions. If the difference between the timestamp for a particular version and the current time is greater than TTL, this record is marked for deletion. If the number of versions for a certain column exceeds the maximum number of stored versions, the record is also marked for deletion.

The approximate physical view of our example table is shown below. You can see that all the data is stored in one Region, and different column families belong to different HFiles inside of this Region.



Physical view

Example

This section describes the HBase data model on the particular example.

Suppose there is a need to store text articles. Each article can have basic attributes, which are changed very rarely (author name, article header, creation datetime, etc.), and tags, which describe the article content and help to perform text search among all articles (arch, concepts, tutorials, ref, etc.). The tags are modified more often.

TIP

Since the data that corresponds to different column families is stored separately, try to distribute it according to the frequency of use.

In the given example, we could create the table `articles` with two column families: `basic` — for storing rarely modified data, and `tags` — for storing document tags. The following JSON illustrates the possible data model in this case.

Please, do not consider this code as illustration of how data are physically stored: this is only a mock-up used to understand the basic data objects of HBase. JSON is more suitable for describing mappings, used in HBase, than a regular two-dimensional table.

```
{
    "article1": { ①
        "basic": { ②
            "author": { ③
                1637054560096: "Test author" ④
            },
            "header": {
                1637056832082: "Test article. Version 3",
                1637055836875: "Test article. Version 2",
                1637054560118: "Test article"
            }
        },
        "tags": {
            "arch": {
                1637054560141: true
            },
            "concepts": {
                1637054560160: true
            },
            "tutorials": {
                1637054564066: true
            }
        }
    },
    "article2": {
        "basic": {
            "author": {
                1637054576501: "Test author2"
            },
            "header": {
                1637054576516: "Test article2"
            }
        },
        "tags": {
            "ref": {
                1637054577512: true
            }
        }
    }
}
```

① The top level of key/value pairs describes table rows: `article1` and `article2` are row keys, unique for each row.

② The second level describes column families, which are fixed for each table row: `basic` and `tags`.

③ The third level describes column qualifiers: `author` and `header` — inside of the column family `basic`; `arch`, `concepts`, `tutorials`, and `ref` — inside of the column family `tags`. Column qualifiers are not fixed, they can be added on fly. Their set can differ from one row to another.

④ The bottom level describes cells — pairs of timestamps and values. Thanks to timestamps, versioning of values is supported.

So, to get a specific data value stored in HBase, you should know the following path: *Table → Row Key → Column Family → Column Qualifier → Timestamp*. Without defining timestamps, you get the most recent data. For example, if you try to get the value of the row with the key `article1` and the column `basic:header`, you get the most recent value `Test article. Version 3`. But if you additionally define the timestamp `1637054560118` — you get the value `Test article`.

```
hbase(main):005:0> get 'articles', 'article1', {COLUMN => 'basic:header', TIMESTAMP => 1637054560118}
COLUMN                CELL
 basic:header         timestamp=1637054560118, value=Test article
1 row(s)
Took 0.0202 seconds
hbase(main):006:0> get 'articles', 'article1', 'basic:header'
COLUMN                CELL
 basic:header         timestamp=1637056832082, value=Test article. Version 3
1 row(s)
Took 0.0054 seconds
```

The main commands, based on the data model showed in this example, are described in the Quick start with HBase shell article.

HDFS architecture

Albert Bagdasaryan

Contents

To Table of Contents

- Components
- Architecture
- Operations with files
- Replication
  - Principles
  - How it works

HDFS (Hadoop Distributed File System) is a highly fault-tolerant distributed file system designed for deployment on low-cost hardware. It provides high-speed access to application data and is best suited for applications with large datasets.

All files and directories in HDFS are represented on the NameNode by inodes that contain various attributes such as permissions, modification timestamp, disk space quota, namespace quota, and access time.

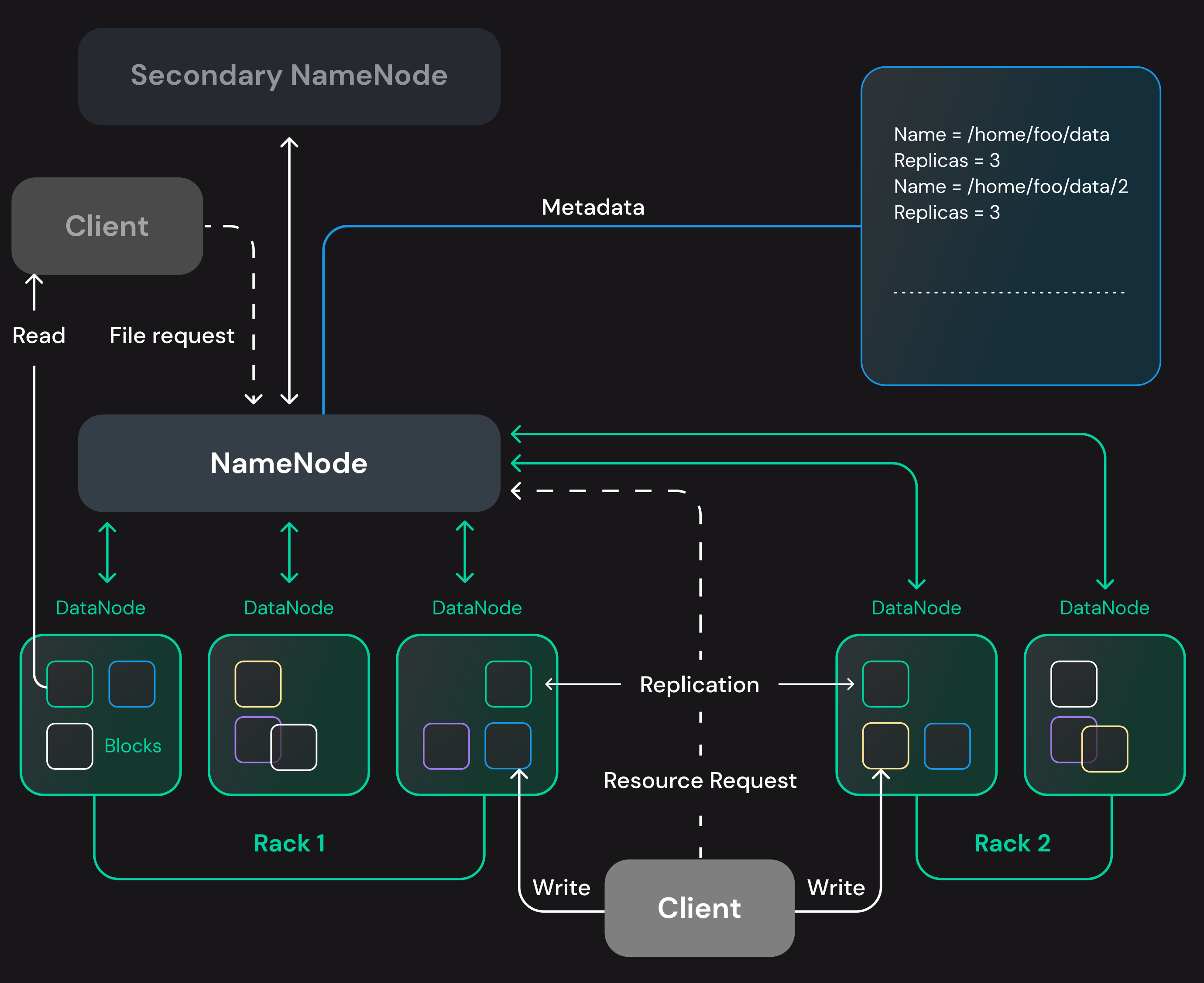
Components

HDFS is supported by the following components:

- **NameNode** acts as the master of the system. It maintains the HDFS namespace, that is, directories and files, and manages the blocks located on the DataNodes.
- **DataNodes** are NameNode subordinates deployed on most of the cluster machines for storing and processing application data. They are responsible for processing read and write requests from clients.
- **Secondary NameNode** periodically makes HDFS namespace checkpoints using the HDFS transaction log and clears that log. It makes the checkpoints available to the NameNode so that the latter can update its namespace image as needed. In the event of a NameNode failure, you can restart the NameNode using the latest saved HDFS checkpoint and transaction log.

Architecture

The NameNode is the primary server that manages the HDFS namespace and controls client access to files. The DataNodes manage the storage attached to the nodes they run on.



HDFS components

HDFS exposes a file system namespace and enables user data to be stored in files. An HDFS file contains one or more blocks stored in multiple DataNodes. The NameNode performs file system namespace operations including opening, closing, and renaming files and directories. The NameNode also controls the mapping of file blocks to the DataNodes.

The DataNodes process read and write requests from HDFS clients. In addition, they perform block creation, deletion, and replication when the NameNode instructs them to do so.

HDFS supports a traditional hierarchical file organization. An application or a user can create directories and then store files in them. The HDFS namespace hierarchy is similar to most other file systems, meaning a user can create, remove, rename, or move files.

The NameNode records any change to the file system namespace or its properties. An application can request the number of file replicas that HDFS should maintain. The NameNode organizes the required number of copies of the file's blocks. This number is called the replication factor of that file.

Operations with files

Unlike local file systems, HDFS does not allow you to modify a file. Files in HDFS can be written only once, and only one process writes to the file at a time. Since HDFS is used for big data, this file system focuses on large file sizes (>10 GB). The files consist of larger blocks than in conventional file systems.

An HDFS block is a BLOB on the underlying local file system with a default size of 128 MB. You can extend the block size up to 256 MB.

IMPORTANT

The files compressed with some codecs (for example, gzip, zlib) are not divided into blocks (**non-splittable**). It means that each file will be processed by a single Mapper. If you want your compressed file to be processed by several Mappers in parallel, use **splittable** formats, e.g. bzip2.

In Hadoop, you can do the following operations on files:

- write
- read
- delete
- replicate

The entire process of creating (writing) an HDFS file is as follows:

1. The client splits its data into parts with the block size.
2. The client connects the NameNode and requests the write procedure with a specific replication factor.
3. In response, the NameNode returns a list of DataNodes to record replicas of the first block.
4. The client goes to the first DataNode in the list and writes the first block. If the connection fails, the client takes the second DataNode from the list and so on.
5. The first DataNode writes the block and passes the block replica to the second DataNode in the list. That node writes the block and sends its replica to the next node in the list. This continues until the entire list is covered.
6. After completion of the block recording, the DataNodes send success messages to the client in reverse direction.
7. As soon as the client receives the first confirmation of the successful block writing, it notifies the NameNode about the block recording, then receives a list of DataNodes to write the second block to, and so on.

The client proceeds to write the next block if it manages to successfully write the previous block to at least one node, that is, the replication will work in background according to the **eventual** principle, spreading the blocks further to DataNodes and achieving the required replication level.

Replication

HDFS provides a reliable way to store large amounts of data in a distributed environment. It replicates file blocks to provide fault tolerance at the block level. Each block has multiple copies in HDFS.

Principles

HDFS splits a large file into several blocks and stores each block in 3 different DataNodes (the default replication factor is 3). For example, if you store a 128 MB file in HDFS, you will use 384 MB (3 * 128 MB) because the file will consist of only one block, and its three replicas will be on different DataNodes.

Notice that a DataNode cannot contain more than one replica of a block. Typically, two copies are on the same rack and one copy is off the rack. It is advised to set the default replication factor to at least 3 so that even if something happens to the rack, one copy is always safe.

NOTE

- The topology (racking) is configured manually. For more information, see Rack awareness article.
- You can change the replication factor for any file in HDFS.

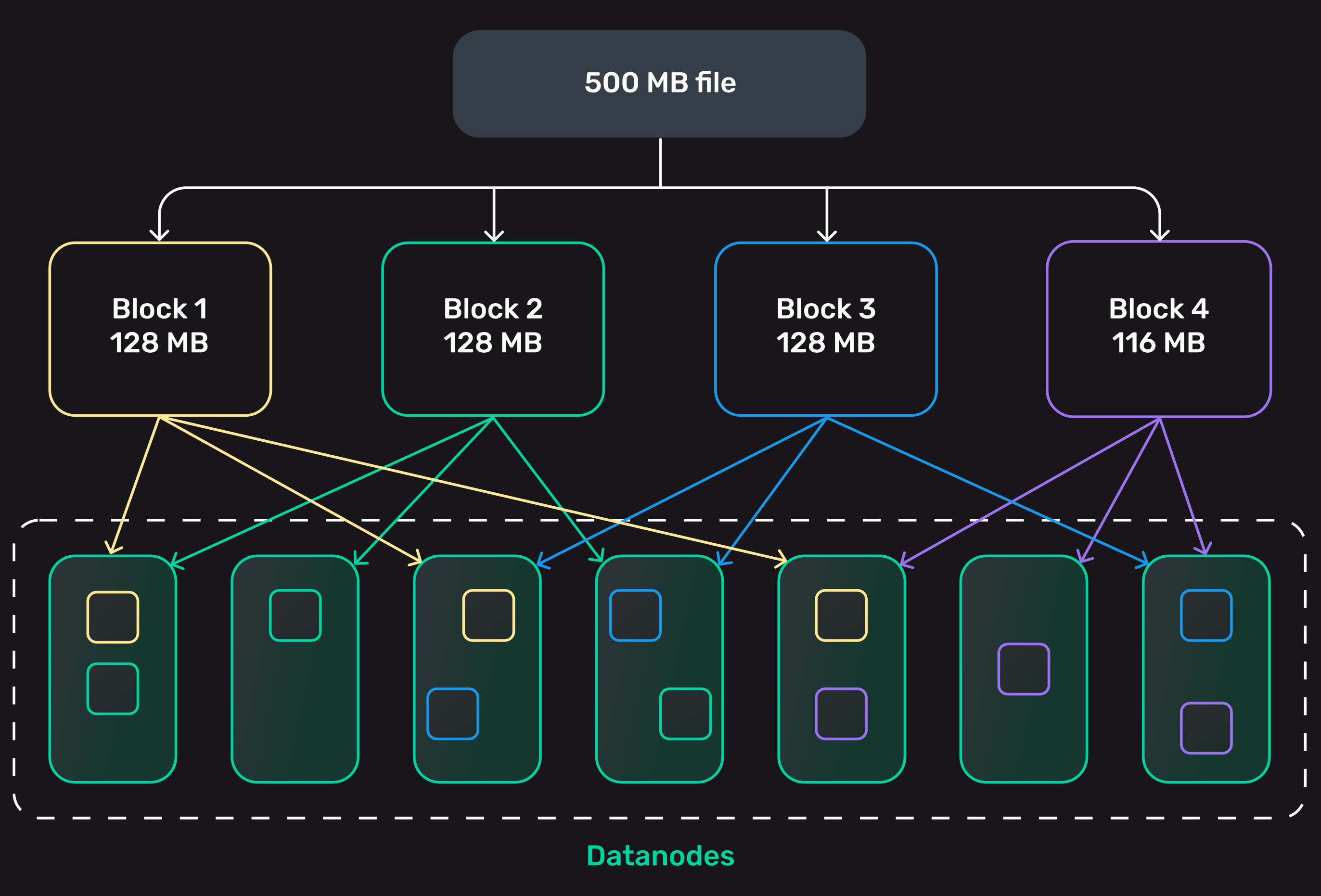
There are the following advantages of splitting HDFS files into blocks:

- Very large files fit better into smaller disks.
- There is less wasted space left on each disk because many 128 MB blocks can be stored on a disk.
- It optimizes file transfer by distributing the load across multiple machines. For example, if a file consists of 10 blocks stored on 10 DataNodes and a client reads the file, the load is distributed across 10 machines.

How it works

The default replication factor is 3 and the block size is 128 MB. When a client saves a file to HDFS, it first splits the file into 128 MB chunks and then writes them to HDFS blocks. Except for the last block, all other blocks will be 128 MB in size. The last block can be less than or equal to 128 MB depending on the size of the file. The default block size is configurable.

For example, we want to store 500 MB of data in HDFS. This data will get split into 3 blocks of 128 MB each, and the last block will be 116 MB.



Distribution of HDFS blocks

All blocks are distributed among DataNodes in accordance with the replication factor. Each block replica is stored in one unique DataNode and cannot be found twice in the same DataNode.

The process of getting an HDFS file is similar to downloading a movie through a torrent. The movie file gets broken down into multiple pieces and these pieces are downloaded from multiple machines in parallel. It helps to download the file faster.

You can set the default replication factor for the entire file system as well as for each file and directory individually. For unimportant files, you can reduce the replication factor, and for very important files, set a higher replication factor.

Whenever a DataNode goes down or fails, the NameNode instructs the DataNodes that have copies of lost blocks to start replicating the blocks to other DataNodes, so that each file again reaches the replication factor assigned to it.

Scalability

Albert Bagdasaryan

To Table of Contents

A Hyperwave cluster is easy to scale by adding new servers or whole server racks to the cluster and increasing the memory in the NameNodes to deal with the rising load. This will generate a lot of "rebalancing traffic" at first, but will deliver extra storage and computation power. Because the NameNodes do matter, we recommend that you pay the premiums for those machines.

Use the following guidelines to scale your existing Hyperwave cluster:

1. Ensure there is potential free space in the data center near the Hyperwave cluster. This space should be able to accommodate the power budget for more racks.
2. Plan the network to cope with more servers.
3. It might be possible to add more disks and RAM to the existing servers — and extra CPUs if the servers have spare sockets. This can expand an existing cluster without adding more racks or making network changes.
4. A hardware upgrade in a live cluster can take considerable time and effort. We recommend that you plan the expansion one server at a time.
5. CPU parts do not remain on the vendors price list forever. If you do plan to add the second CPU, consult with your reseller on when they will cut the price of CPUs that are your existing parts and buy these parts when available. This typically takes at least 18 months.
6. You are likely to need more memory in the master servers.

TEZ overview

Konstantin Alpashkin

Contents

To Table of Contents

Tez UI

The ADH Tez service is based on the Apache TEZ project. You can use this service to create high-performance batch and interactive data processing applications coordinated by YARN. Tez models data processing as a Directed Acyclic Graph (DAG), where vertices represent application logic and edges represent data movement. In fact, any arbitrary DAG can be executed directly in Tez.

In ADH, Tez is embedded in the Hive service. If you install Hive, the Tez service will be configured by default as an engine for executing SQL queries.

You can find all service settings in the Hive section of ADCM. Tez XML parameters section is located on the Hive configuration page. All the default parameters are presented on the Hive configuration parameters page. If you need to add any additional parameter, use the advanced mode.

Tez UI

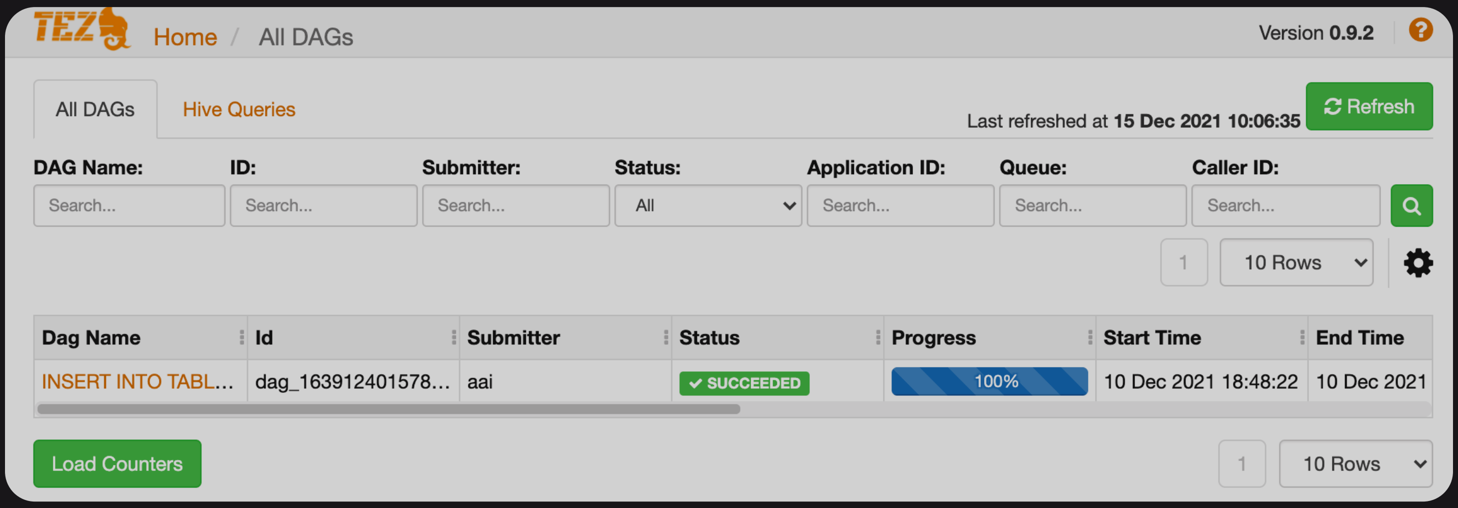
A link to the Tez UI is located on the Hive page in ADCM. Typically, it includes a fully-qualified domain name (FQDN) and a port number that defaults to `9999`. Tez provides interactions with YARN Application Timeline Server and reflects current and historical appearance of Tez applications in the Tez UI.

The Tez UI link typically has the following structure: *http://FQDN:9999*.

When you use an external host to reach out to the Tez UI, there might be an issue with DNS resolution and connectivity with YARN Application Timeline Server. The solution is to explicitly specify external IP addresses of external hosts in the *hosts* file. Regardless of the OS on your external host machine, the *hosts* file is typically located in the *~/etc/* directory. Add all the FQDN-IP matches of your cluster to this file. A typical hosts file has the following view:

```
127.0.0.1       localhost
::1         localhost
172.31.1.234 adh1.us-west-2.compute.internal
172.31.3.37 adh2.us-west-2.compute.internal
172.31.2.65 adh3.us-west-2.compute.internal
```

After making these changes, you can open the Tez UI on your external host.



Dag Name	Id	Submitter	Status	Progress	Start Time	End Time
INSERT INTO TABL…	dag_163912401578…	aai	✓ SUCCEEDED	100%	10 Dec 2021 18:48:22	10 Dec 2021

Tez UI

Iceberg tables

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
  - Differences from Hive tables
  - Features
- Architecture
  - Iceberg catalog
  - The metadata layer
  - The data layer
  - Query lifecycle
- Use cases
  - Branching
  - Tagging
  - Time travel
  - Rollback
  - Schema evolution

Overview

Apache Iceberg is an open-source table format for data lakes that enables ACID transactions, time travel, schema evolution, partition evolution, and other features.

It acts as an abstraction layer between a query engine and the data. Iceberg tables work similar to SQL tables and can be integrated with compute engines like Spark, Hive, Impala, and Trino.

Iceberg solves issues with traditional table formats, like the Hive table format, including data consistency, performance issues, the lack of schema evolution, and consistent concurrent writes in parallel.

Another important function of Iceberg tables is reliability. It allows tracking changes in a dataset and rolling back to the desired database state.

> NOTE
>
> In the ADCM paradigm, Iceberg is not a service. It's a set of parameters for services like Hive, Spark, and Impala. These parameters can be configured to alter the way these services interact with data. By default, it's enabled.

For more information about Iceberg tables, refer to the Iceberg tables specification.

Differences from Hive tables

Traditionally, the Hive tables format has been utilized for data lakes. This format monitors data at the directory level, which can lead to performance issues when numerous file list operations are required. Additionally, there's a risk of data loss when executing file list operations on an object store like S3, which is only eventually consistent.

In contrast, Iceberg maintains a comprehensive list of all files within a table by using a persistent tree structure. Any changes to a table are executed via an atomic object/file-level commit, which subsequently updates the path to a new metadata file. This metadata file contains the locations of all individual data files.

Features

The main features of Iceberg tables are:

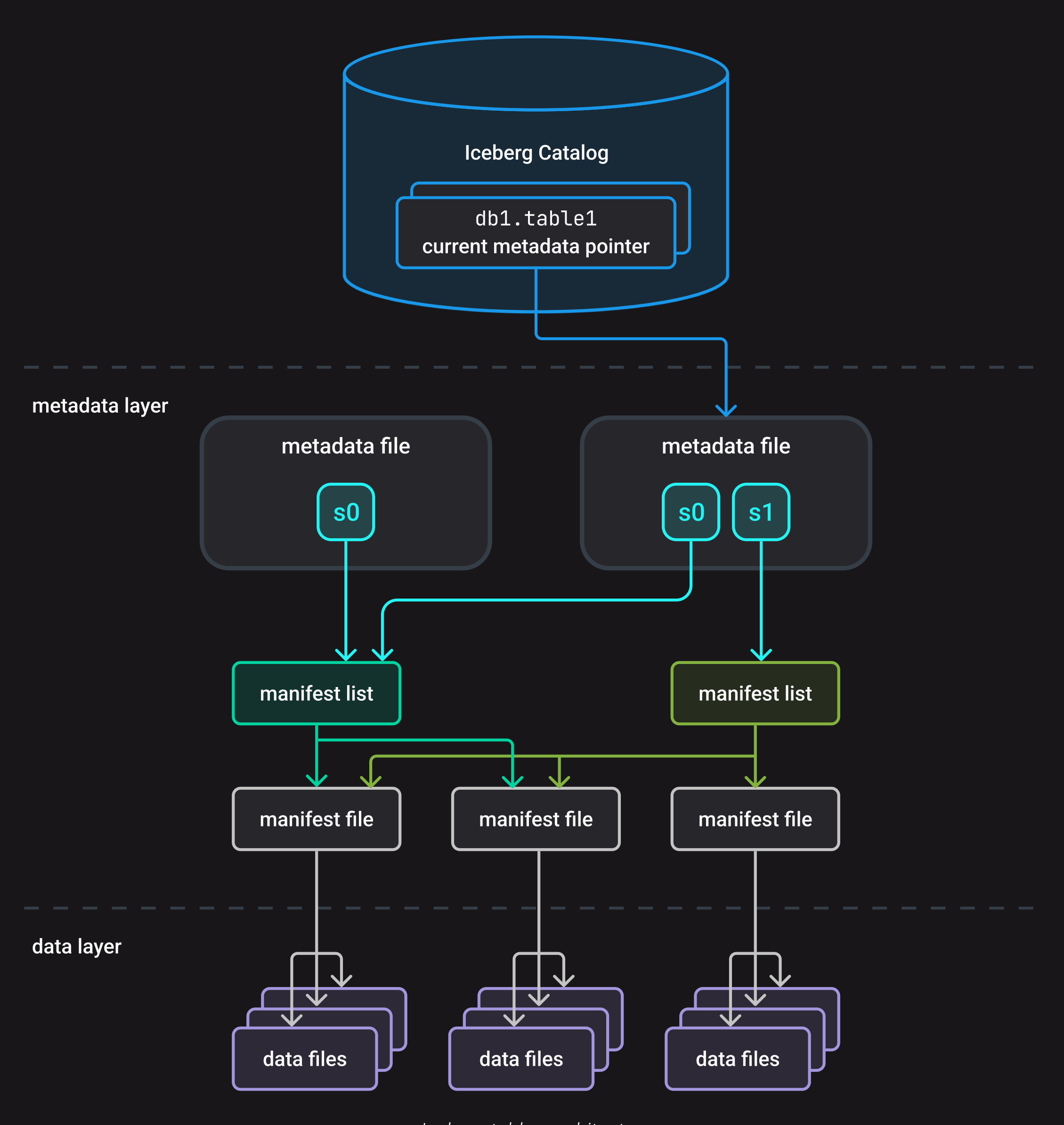
- **Expressive SQL**
  Iceberg supports a wide range of SQL operations, enabling tasks like row updates, data merging, and targeted deletes. It allows for data file rewrites and supports such queries as, for example, `INSERT INTO`, `MERGE INTO`, and `INSERT OVERWRITE`.
- **Schema evolution**
  Changes to the Iceberg schema only affect the metadata, not the actual data files. This supports seamless modifications to the data structure, such as adding, dropping, updating, reordering, and renaming columns.
- **Partition evolution**
  Unlike standard partitioning, Iceberg tables allow the alteration of the data's partitioning schema without a need for a rewrite of all previously partitioned data.
- **Snapshots**
  A snapshot represents the table's state at a particular moment. Iceberg maintains a log of past snapshots, enabling time-travel queries.
- **Time travel and rollback**
  Iceberg's time travel functionality enables data querying at a specific point in time. The rollback feature lets users restore tables to their prior state.
- **Transactional consistency**
  Iceberg supports ACID transactions, ensuring safe concurrent writes in the cluster and preventing read operations from being impacted by writes. Whenever changes occur, Iceberg generates a new, immutable version of the table's data files and metadata.
- **Faster querying**
  Iceberg provides several features aimed at boosting query speed and efficiency. These include the possibility of incremental data processing, fast scan planning, filtering metadata files, and the ability to filter out data files that don't contain matching data.

> NOTE
>
> The available Iceberg features depend on the service you are using and its version. To find out more information about the available Iceberg features in a specific service, please refer to the documentation for that service.

Architecture

There are three main layers in the Iceberg table architecture:

- Iceberg catalog layer
- The metadata layer
- The data layer



Iceberg tables architecture

Iceberg catalog

In Iceberg, a group of tables is usually grouped into namespaces. An Iceberg catalog keeps track of those namespaces and tables, along with references or pointers to the corresponding metadata files.

The catalog is an add-on to an existing metadata service, such as HDFS, Hive Metastore, or a relational database (JDBC). ADH uses Hive Metastore as the Iceberg catalog implementation.

The metadata layer

Apache Iceberg uses three types of metadata files to maintain a file location map:

- **Manifest files** — each manifest file tracks a subset of the files in a table for a single snapshot. These files track the individual files, their partitions, and column metrics.
- **Manifest lists** — define a snapshot of the table and list all the manifest files that make up that snapshot. Additionally, they store metadata for every manifest file in a given snapshot.
- **Metadata files** — define the table and track manifest lists, current and previous snapshots, the table and partition schemas.

The data layer

Iceberg tracks each data file in a table and saves information about its partitions and columns.

There are two categories of files in the data layer:

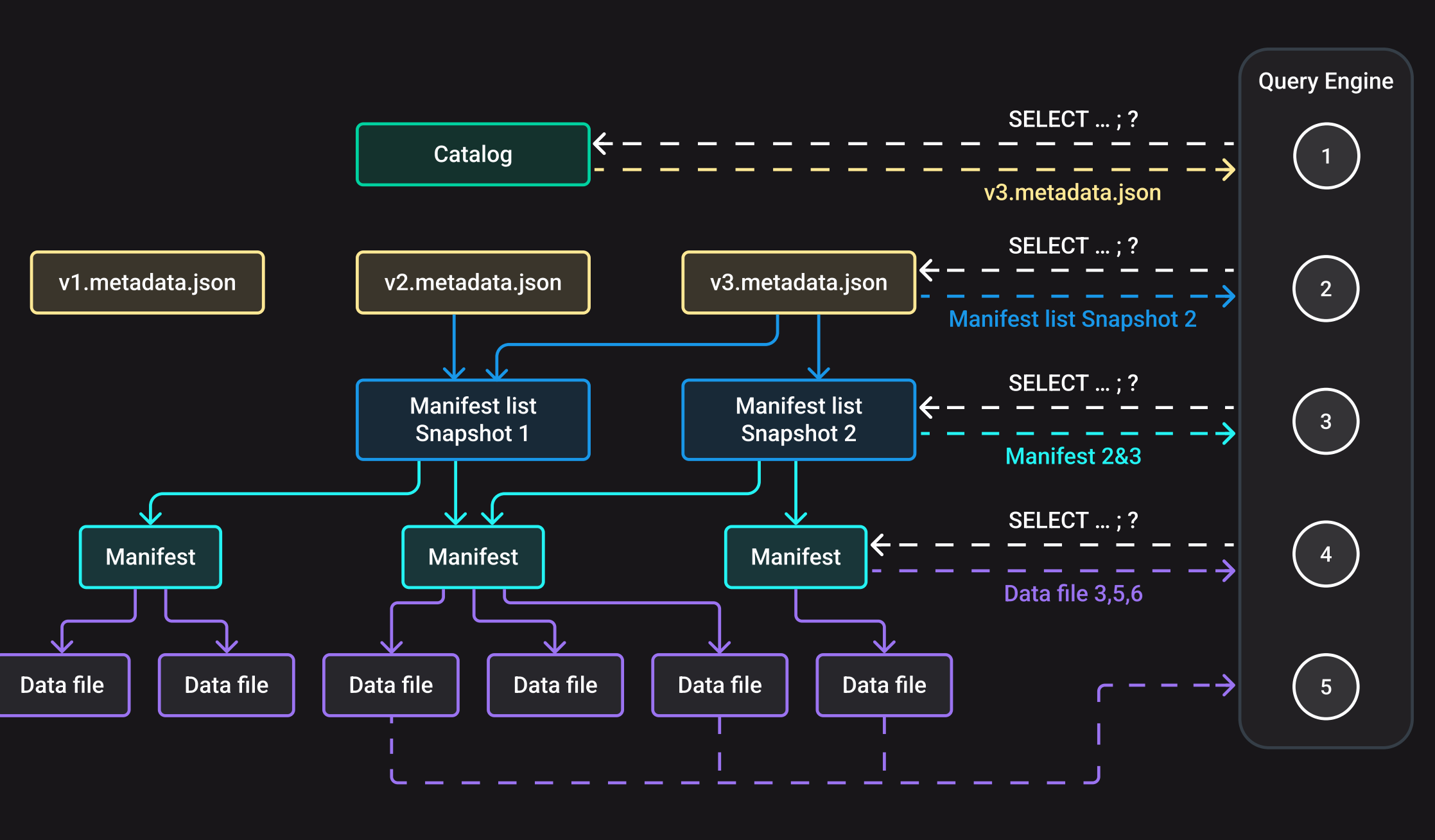
- **Data files** — actual data in such file formats as Parquet, Avro, or ORC.
- **Delete files** — the records about the files that have been marked as deleted.

Query lifecycle

To illustrate how a simple query operates within a cluster utilizing Iceberg tables, we can break down the lifecycle into the following steps:

1. When a query is submitted, the engine needs data from the Iceberg catalog and sends a request to retrieve the file path of the table's current metadata file.
2. The engine gets the following information from the metadata file:
   - The table's schema.
   - The partitioning of the table. It will help to determine how the data is organized so it can eliminate files that are not relevant to the query.
   - The current snapshot's manifest list. It will help to determine what files to scan.
3. Next, the engine retrieves a list of manifest files from the manifest list. Each manifest file contains a list of data files that comprise the data in the selected snapshot. Key details include the `partition-spec-id`, which indicates the partition schema used when writing the snapshot, and the partition field summaries (`PartitionFieldSummary`), which can help to determine whether a manifest can be skipped.
4. The engine then verifies in parallel each of the selected manifest files, reading the metadata. This metadata includes the `schema-id` and `partition-id` values for each data file and the type of content (checks if it's a data file or a delete file) to determine whether a data file is relevant to the query.
5. The engine scans the relevant files and returns the results.

The process of running a query on an Iceberg table is considerably quicker than on other table formats, as the engine can disregard irrelevant files, resulting in a much smaller portion of the table that needs to be scanned.



Query lifecycle inside Iceberg tables

Use cases

Below are examples of SQL requests that illustrate some of the Iceberg table features.

Branching

The branching is similar to version control systems like Git. You can create a local branch, commit necessary changes, and merge to the main branch.

An example statement for creating a branch:

```sql
ALTER TABLE test.db.table CREATE BRANCH 'task_1';
```

This syntax creates a branch with the same state as the current table, but you can create a branch of a specific table state using the `FOR` clause.

An example statement for adding new values to the created branch:

```sql
INSERT INTO test.db.table.task_1 VALUES (1, 'a'), (2, 'b');
```

An example statement for reading from a specific branch:

```sql
SELECT column1, column2 FROM test.db.table.task_1;
```

For more information on branching, see the Branching and Tagging article.

Tagging

In Iceberg, you can mark a snapshot to return to it later. You can create, query, or delete a tag.

An example of syntax for creating a tag:

```sql
ALTER TABLE test.db.table CREATE TAG `test` RETAIN 10 DAYS;
```

Time travel

Using the time travel feature, you can make a request to a specific state of the table in time.

An example of syntax for querying a table at a specific time:

```sql
SELECT * FROM test.db.table TIMESTAMP AS OF '2007-09-03 04:21:00';
```

An example of syntax for querying a specific snapshot:

```sql
SELECT * FROM test.db.table VERSION AS OF 10726387402803;
```

An example of syntax for querying the head of a specific branch:

```sql
SELECT * FROM test.db.table VERSION AS OF 'task_1';
```

Read more about the time travel feature in the Time travel section.

Rollback

Using the rollback feature, you can return a table to a previous state. The state can be specified either as a snapshot ID or a timestamp.

An example of syntax for rolling back to a snapshot:

```sql
ROLLBACK TABLE test.db.table TO SNAPSHOT '2894821635228246189';
```

Schema evolution

With Iceberg tables, you can alter a table structure by adding, dropping, renaming, or reordering columns.

Since Iceberg only updates metadata when table schema is altered, the data doesn't change when you make a schema update.

An example of syntax for changing a column name:

```sql
ALTER TABLE test.db.table CHANGE COLUMN names employee_names STRING;
```

Impala architecture

Eugenia Kuzina

Contents

To Table of Contents

- Overview
- Physical representation
- Components of the Impala service in ADH
  - Impala Daemon
  - Impala Statestore
  - Impala Catalog Service
  - Impala Client
- Impala role in ADH
  - Impala and Hive
  - Impala Metadata and Metastore
  - Impala and HDFS
  - Impala and HBase

Overview

Impala provides fast, interactive SQL queries on data stored in HDFS, HBase, or S3. In addition to the unified storage platform, Impala also uses the same metadata, SQL syntax (Hive SQL), and JDBC driver as Apache Hive. It makes Impala a unified platform for real-time or batch-oriented queries.

Impala is targeted for integration with standard business intelligence environments and supports the most relevant industry standards: clients can connect via JDBC, the authentication is accomplished with Kerberos or LDAP.

Impala has the following advantages:

- The SQL interface that data scientists and analysts already know.
- Ability to query high volumes of data.
- Distributed queries in a clustered environment for easy scaling and the use of cost-effective commodity hardware.
- Ability to share data files between different components without copying or export/import operations. For example, you can write and transform data with Hive, and query with Impala.

The Impala-based solution consists of the following parts:

- **Clients** — Hue, JDBC clients, and the Impala Client can interact with Impala. These interfaces are used to issue queries or complete administrative tasks, for example, connecting to Impala.
- **Hive Metastore** — stores information about the data available to Impala. For example, Metastore lets Impala know what databases are available and what the structure of those databases is. If you create, drop, and alter objects, and load data into tables through Impala SQL statements, the relevant metadata are changed automatically.
- **Impala** — processes that run on DataNodes, coordinate and execute queries. Each instance of Impala can receive, plan, and coordinate queries from Impala clients. Queries are distributed among Impala nodes, and these nodes then act as workers, executing parallel query fragments.
- **HBase and HDFS** — the storage for data that can be queried.

User applications send SQL queries to Impala through JDBC. The user application may connect to any Impala daemon in the cluster. This daemon becomes the coordinator for the query. Impala parses the query and analyzes it to determine what tasks need to be performed by Impala instances across the cluster. The execution is planned to achieve optimal efficiency. Local Impala instances access HDFS and HBase services to get data. Each Impala daemon returns data to the coordinator that aggregates the results and sends them to the client.

Physical representation

By default, Impala stores data files of an unpartitioned table in the root directory. However, all data files that are located in any directory below the root are part of the table data set. This is a common approach to working with unpartitioned tables, and Apache Hive also uses this technique.

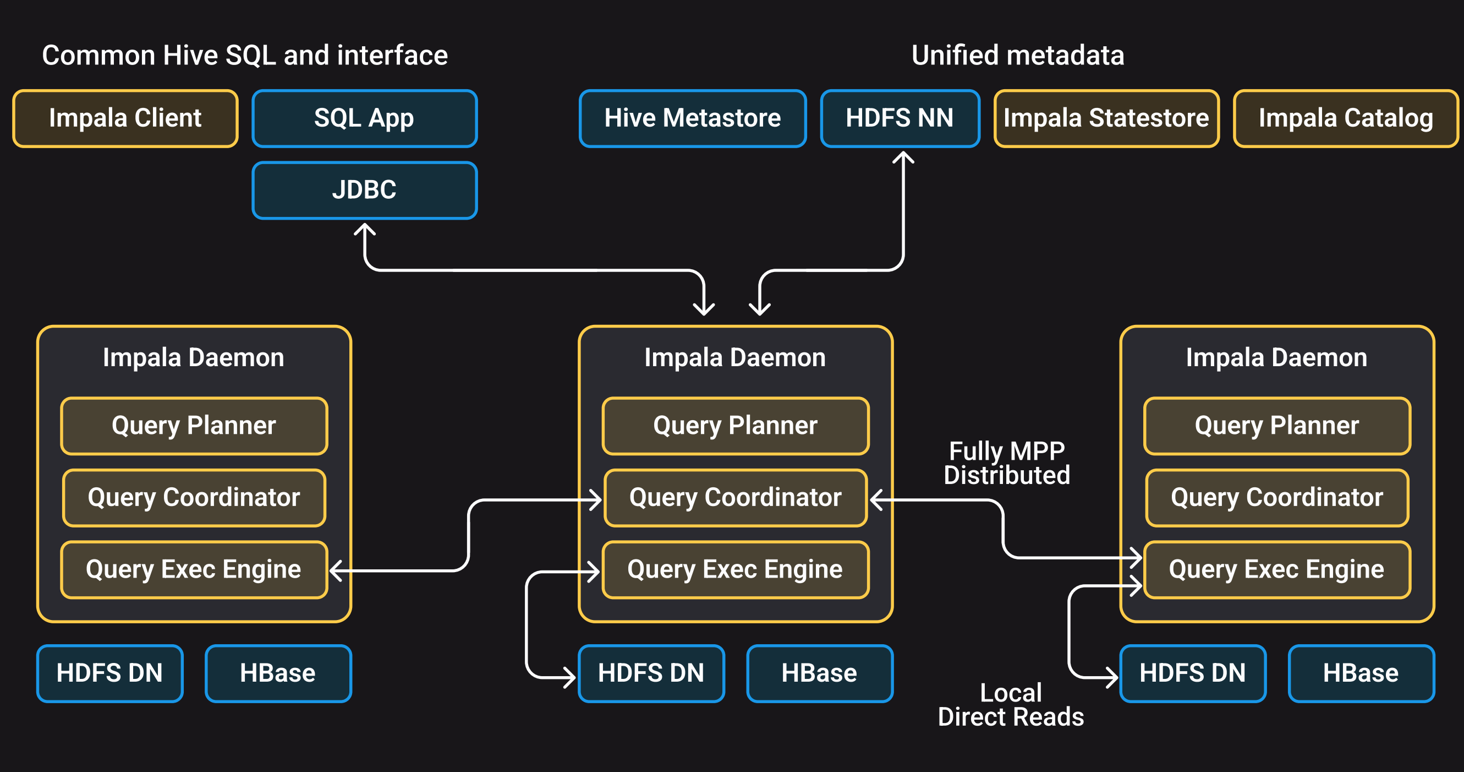
Data files of a partitioned table are placed in subdirectories. The path reflects the partition column values. For example, for day `17`, month `2` of table `T`, all data files will be located in the `<root>/day=17/month=2/` directory. Note that this form of partitioning does not imply a co-location of individual partition data. The blocks of the partition's data files are distributed across HDFS DataNodes.

Impala also gives the user great flexibility when choosing file formats. It supports the most popular file formats: Avro, RC, Sequence, plain text, and Parquet. These formats can be combined with different compression algorithms, such as snappy, gzip, and bz2. You can specify the storage format in the `CREATE TABLE` or `ALTER TABLE` statements. It is also possible to select a separate format for each table partition.

In most use cases, the Apache Parquet is recommended as an up-to-date, open-source columnar file format offering both high compression and high scan efficiency. Most Hadoop-based frameworks including Hive, Pig, MapReduce, and Cascading can process Parquet.

Components of the Impala service in ADH

Impala is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes that run on specific hosts within your cluster.



Impala architecture

Impala Daemon

The core Impala component is the ***Impala Daemon***. It is represented by the `impalad` process. The Impala daemon receives a client request, becomes a coordinator, parses a query request, splits it into different tasks, and distributes them to other `impalad` node processes. After each `impalad` worker node process receives the request, it starts to execute a local query (for example, querying the HDFS DataNode or an HBase region server), and returns the query results to the central coordinator. The coordinator collects query results from other `impalad` processes, combines them, and sends the result to the client.

The Impala daemon consists of three modules:

- **Query planner** — receives query requests from SQL APP, JDBC, and other clients, and converts the query into many sub-queries (creates an execution plan).
- **Query coordinator** — distributes these sub-queries among the Impala nodes.
- **Query executor** — is responsible for the execution of the sub-queries and returning the sub-query results to the coordinator.

Impala daemons can be deployed in one of the following ways:

- HDFS and Impala are co-located, and an Impala daemon runs on each host that is a DataNode.
- Impala is deployed separately in a compute cluster and reads remotely from HDFS, S3, etc.

The Impala daemons are in constant communication with Statestore to confirm which daemons are healthy and can accept new queries.

They also receive broadcast messages from the Impala Catalog Service whenever any Impala daemon in the cluster creates, alters, or drops any type of object, or when Impala processes an `INSERT` or `LOAD DATA` statement.

You can control which hosts operate as query coordinators and which work as query executors to improve the scalability of highly concurrent workloads on large clusters. For more information, see How to configure Impala with dedicated coordinators.

Impala Statestore

The ***Impala Statestore*** component collects the health status of all Impala daemons and continuously forwards results to all `impalad` process nodes. Statestore is represented by the `statestored` daemon process. The cluster only needs this process on one host. If an Impala daemon goes offline due to a hardware failure, network error, software issue, or other reasons, the Statestore informs all other Impala daemons, and new queries are not sent to the unreachable Impala daemon.

Since the Statestore helps when something goes wrong and broadcasts the metadata to the coordinators, this is not always critical to the normal operation of the Impala cluster. If the Statestore becomes unavailable, the Impala daemons continue to run and distribute work among themselves as they normally do when working with data known to Impala. The cluster just becomes less reliable if some Impala daemons fail. The metadata becomes also less consistent since it changes when the Statestore is offline. When the Statestore becomes available, it reestablishes the communication with the Impala daemons and resumes monitoring and broadcasting functions.

If you execute a DDL statement while the Statestore is down, the queries that access the new DDL-created object will fail.

Impala Catalog Service

The ***Impala Catalog Service*** component sends broadcast messages with metadata changes made by Impala SQL statements to all the Impala daemons in a cluster. For example, when you create a table, load data, or perform other table-changing or data-changing operation through Impala. It is represented by the `catalogd` daemon process.

You only need this process on one host in a cluster. Since the metadata changes are passed through the Statestore daemon, it makes sense to install ***Impala Catalog Service*** and ***Impala Statestore*** components on the same host.

You can use the ***load_catalog_in_background*** parameter to determine when the metadata of a table should be loaded. The ***load_catalog_in_background*** checkbox is located on the ***CLUSTERS*** → ***ADH cluster*** → ***Services*** → ***Impala*** → ***Components*** → ***Impala Catalog Service*** → ***Configuration*** tab.

If it is set to `false`, the table metadata is loaded when it is referenced for the first time. In this case, the first run of a query can be slower than subsequent runs.

If ***load_catalog_in_background*** is set to `true`, the Catalog Service loads metadata for a table even if that metadata is not required for any query. So metadata can be already loaded when the first requested query is executed. However, it is not recommended to set this option to `true`. Background load can interfere with query-specific metadata loading and cause occasional long-running queries that are difficult to diagnose. Impala also may load metadata for tables that are never used, increasing catalog size and memory usage for both the Catalog Service and Impala daemon.

Impala Client

The ***Impala Client*** component is represented by impala-shell that is a command-line interface for issuing queries to the Impala daemon. You can install this on one or more hosts anywhere on your network. It can connect remotely to any instance of the Impala daemon.

Impala role in ADH

A major Impala goal is to make SQL-on-Hadoop operations fast and efficient. Impala can interchange data with other Hadoop components, as both a consumer and a producer, so it can fit flexibly into your ETL and ELT pipelines.

Impala and Hive

Where practical, Impala uses existing Apache Hive infrastructure that many Hadoop users already have in place to perform long-running, batch-oriented SQL queries. Impala stores table definitions in MySQL or PostgreSQL database known as Metastore, the same database where Hive keeps this type of data. Thus, Impala can access tables defined or loaded by Hive, if all columns use Impala-supported data types, file formats, and compression codecs.

The initial focus on query features and performance means that Impala can read more types of data with the `SELECT` statement than it can write with the `INSERT` statement. To query data using the Avro, RCFile, or SequenceFile file formats, you should load the data using Hive.

The Impala query optimizer can also use table statistics and column statistics. Originally, you gathered this information with the `ANALYZE TABLE` statement in Hive. In Impala, use the `COMPUTE STATS` statement instead. `COMPUTE STATS` requires less setup, is more reliable, and does not require switching back and forth between impala-shell and the Hive shell.

Impala Metadata and Metastore

As described above, Impala maintains information about table definitions in a central database known as Metastore. Impala also tracks metadata for the low-level characteristics of data files — the physical locations of blocks within HDFS.

For tables with a large volume of data and/or many partitions, retrieving all the metadata can be time-consuming. Each Impala node caches this metadata to reuse for future queries on the same table.

If the table definition or the data in the table is updated, all Impala daemons in the cluster must receive the latest metadata, and replace the obsolete cached metadata, before issuing a query on that table. The metadata update is automatically coordinated through the ***Impala Catalog Service*** component, for all DDL and DML statements issued through Impala. For DDL and DML issued through Hive, or changes made manually on files in HDFS, you need to use the `REFRESH` statement (when new data files are added to existing tables) or the `INVALIDATE METADATA` statement (for new tables, after dropping a table, performing an HDFS rebalance operation, or deleting data files). `INVALIDATE METADATA` retrieves metadata for all the tables tracked by Metastore. If you know which tables have been changed outside of Impala, you can run `REFRESH <table_name>` for each affected table to get the latest metadata for those tables.

Impala and HDFS

Impala uses HDFS as its primary data storage. Impala relies on the redundancy provided by HDFS to protect against hardware or network failures on individual nodes. Impala table data is represented as data files in HDFS using standard HDFS file formats and compression codecs. When data files are present in the directory for a new table, Impala reads them all, regardless of file names. New data is added to files with names controlled by Impala.

Impala and HBase

HBase is an alternative to HDFS as a storage for Impala data. It is a database system built on top of HDFS, without built-in SQL support. To query the contents of the HBase tables through Impala, define tables in Impala and map them to equivalent tables in HBase. This way you can also execute JOIN queries including both Impala and HBase tables.

Kyuubi architecture

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Architecture

High availability

Overview

Apache Kyuubi is a distributed and multi-tenant JDBC interface for large-scale data processing and analytics. It's built on top of the Apache Spark engine and is able to run any custom JDBC-compliant tool (such as Hive, Flink, etc.).

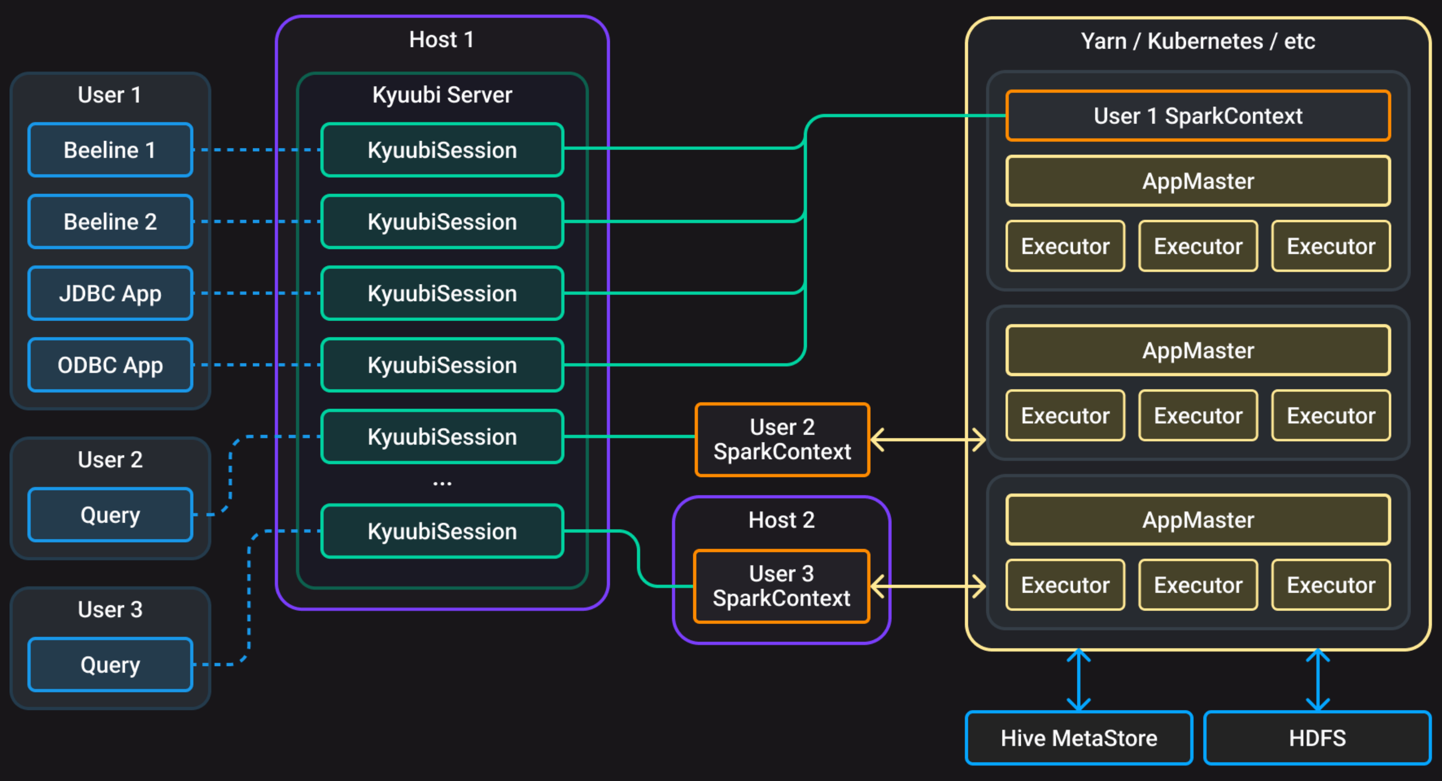
It's designed with purpose to create an SQL gateway for operations on data warehouses and data lakes. Kyuubi implements the Hive Service RPC module, the data access interface used in HiveServer2. For more information about what distinguishes Kyuubi from them, read the Kyuubi vs HiveServer vs Spark Thrift Server article.

Kyuubi's main features:

- **Multi-tenancy** — end-to-end multi-tenancy support for resource acquisition and data access through a unified authentication and authorization layer;
- **High availability** — load balancing through ZooKeeper;
- **Multiple workloads** — supports multiple different workloads with one platform, one data copy and one SQL interface.

Architecture

The main Kyuubi component is **Kyuubi Server** — a daemon process that handles concurrent connections and execution requests from clients and manages their execution.



Host 1
Kyuubi Server
User 1
Beeline 1
Beeline 2
JDBC App
ODBC App
User 2
Query
User 3
Query
KyuubiSession
KyuubiSession
KyuubiSession
KyuubiSession
KyuubiSession
...
KyuubiSession
User 2 SparkContext
Host 2
User 3 SparkContext
Yarn / Kubernetes / etc
User 1 SparkContext
AppMaster
Executor
Executor
Executor
AppMaster
Executor
Executor
Executor
AppMaster
Executor
Executor
Executor
Hive MetaStore
HDFS

Kyuubi architecture

When a user requests access to data via one or several clients, the server creates a **Kyuubi session** for that user. All of the user's requests are bound to the corresponding session.

For each session, Kyuubi creates a dedicated **SparkContext** for processing user requests: to compile, optimize, and execute SQL statements.

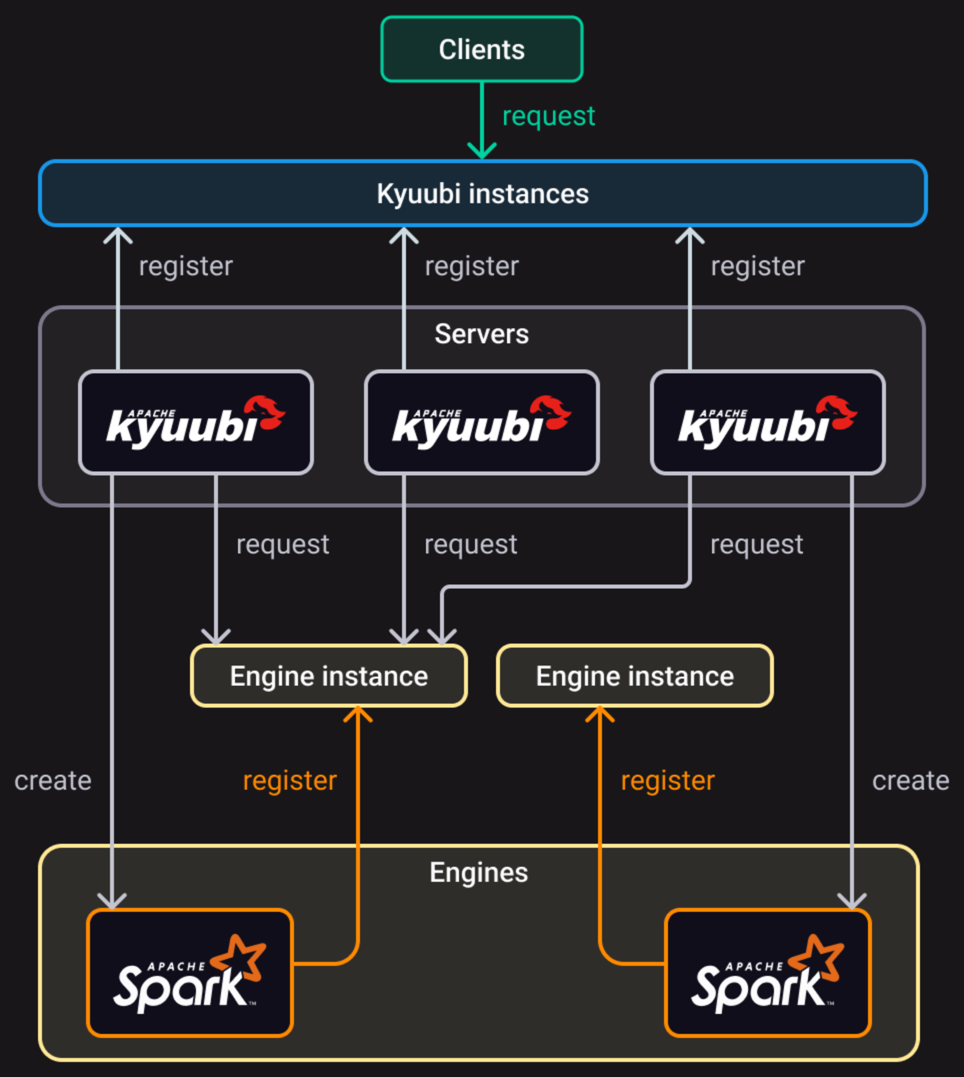
SparkContexts can be created locally, in client deploy mode (`--deploy-mode client`) by this service instance, or in YARN or Kubernetes clusters in cluster deploy mode (`--deploy-mode cluster`). They can manage their lifecycle, cache and recycle themselves, and are not affected by failover on the Kyuubi server.

If there are no active sessions for a SparkContext, Kyuubi frees its resources.

High availability

Kyuubi provides high availability (HA) and load balancing capabilities based on ZooKeeper.

In HA mode, SparkConexts can be created by several Kyuubi instances and shared between all servers. Kyuubi instances registered to the same namespace provide the ability to load balance each other.



Clients
request
Kyuubi instances
register
register
register
Servers
kyuubi
kyuubi
kyuubi
request
request
request
Engine instance
Engine instance
create
register
register
create
Engines
Spark
Spark

Kyuubi HA mode

When a client makes a request, it can choose between multiple servers.

The selected Kyuubi instance picks an available SparkContext. If none is available, the server creates a new one.

If the same user requests a new connection, it will be created on the same or another Kyuubi instance, but SparkContext will be reused.

When another user makes a request, the whole process is repeated. When a client chooses a Kyuubi instance, the namespaces used to store the address of the engine instances are isolated based on the user (by default), and different users cannot access other's instances across the namespace.

Kyuubi vs HiveServer vs Spark Thrift Server

Elena Kostyuchenko

To Table of Contents

Kyuubi, Spark Thrift Server, and HiveServer share the same interfaces and protocols, making their usage similar for users. The main advantage of Kyuubi and Spark Thrift Server over HiveServer is improved performance.

In the table below, you can see how Kyuubi compares to HiveServer and Spark Thrift Server in performance and feature availability.

Feature	HiveServer	Spark ThriftServer	Kyuubi
Interface	HiveJDBC	HiveJDBC	HiveJDBC
SQL syntax	Hive QL	Spark SQL	Spark SQL
SQL optimizer	Hive Optimizer	Spark SQL Catalyst	Spark SQL Catalyst
SQL parsing & planning	Server side	Server side	Engine side
UDF loading	Server side	Server side	Engine side
Job submission	A query is split into multiple Spark applications, called `RemoteDriver`	Within the server	Via Kyuubi engines with different isolation policies
Spark compatibility	Single version-specific	Built-in	Multi-version
Catalog management	HMS	HMS	HMS
High availability	Yes	No	Yes
Multitenancy	Yes	No	Yes
Permission control	SQL standard, fine-grained	No	SQL standard, fine-grained
Performance	Fair	Good	Good
Client concurrency	High	Low	High
Queuing	For queries	No	For engines
Resource setting	For queries	For pools	For engines
Resource management	YARN	Pools	YARN, Kubernetes, etc.
Resource occupancy time	Within a query	Permanent	Several modes supported depending on a share level, including dynamic resource allocation, available for all

Share levels of Kyuubi engines

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Share levels
  - CONNECTION
  - USER
  - GROUP
  - SERVER
  - Subdomain
- Configuration

Overview

Kyuubi share levels feature enables configuring conditions for sharing engines between users. This feature allows for more flexibility in cluster workload management: more isolated engines provide better engine stability and higher shareability of the engines means more frugal cluster resources management.

The current supported share levels are:

- CONNECTION
- USER
- GROUP
- SERVER

The default share level is USER. Beyond the main four share levels, you can configure the subdomain setting to create several engines.

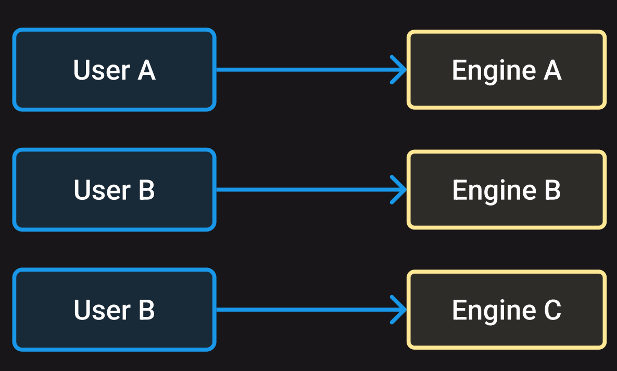
All supported share levels can be used together in a single Kyuubi server or cluster. The share levels work the same way for both single-node configurations and in clusters with high availability (HA).

Share levels comparison

Share level	Number of engines	Scenario	Isolation degree	Shareability
CONNECTION	One engine per session	Large-scale ad hoc ETL	High	Low
USER	One engine per user	Small-scale ad hoc ETL	Medium	Medium
GROUP	One engine per primary group	Small-scale ad hoc ETL	Low	High
SERVER	One engine per cluster	Admin	Highest if secured, lowest if unsecured	Admin only if secured

Share levels

CONNECTION

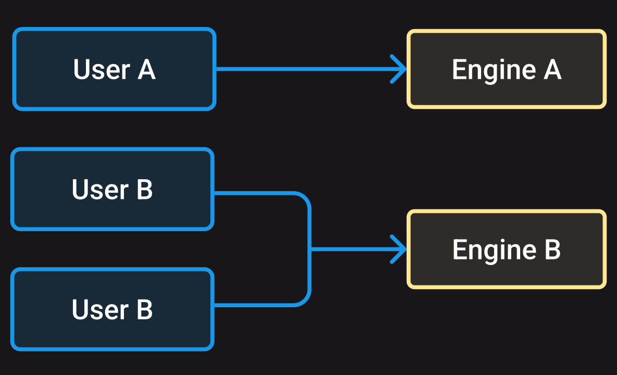


Client requests to engines in the CONNECTION share level

The CONNECTION share level creates an isolated engine, available only to one session. Within that session, a user can send multiple operation requests, including metadata calls or queries.

Once the session that created the engine is closed, the engine shuts down as well.

USER



Client requests to engines in the USER share level

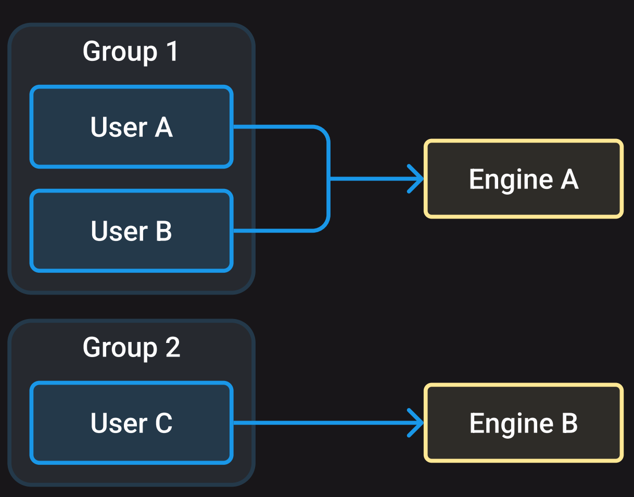
The USER share level isolates the engine by user. Each engine has a specific user as the owner of that engine. Any requests sent from one or multiple clients are redirected to that user's engine.

If the user has several sessions open, they will belong to the one same SparkContext, but each session will have its own SparkSession instance, which contains separate session state, including temporary views, SQL config, UDFs, etc.

To make a singleton SparkSession instance, which would be shared across sessions, set the value of the `kyuubi.engine.single.spark.session` parameter to `true` manually or using ADCM.

The engine with the USER share level shuts down after all sessions connected to it are closed and its time-to-live (TTL) has expired.

GROUP



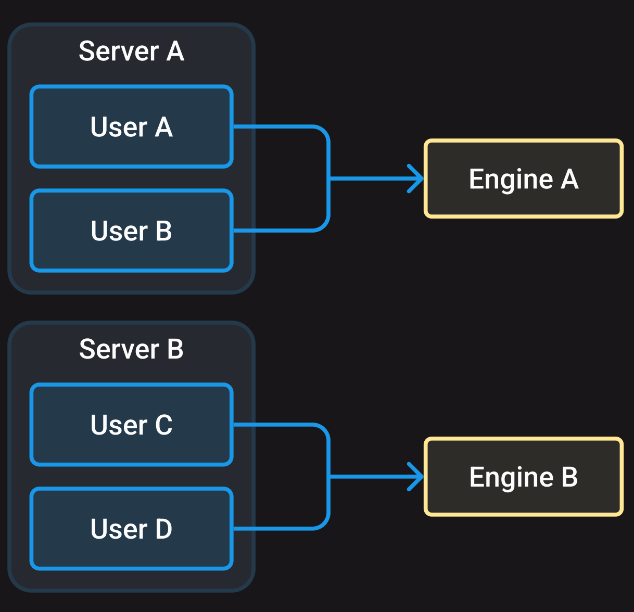
Client requests to engines in the GROUP share level

In the GROUP share level, an engine is shared between users from the same group. All sessions of users in the same group will belong to the group engine.

A group engine has the group name as the name of its owner. The grouping of users follows the Hadoop GroupsMapping. If the primary group is not found, it falls back to the USER level.

The engine with the GROUP share level shuts down after all sessions of all users in the group are closed and its time-to-live (TTL) has expired.

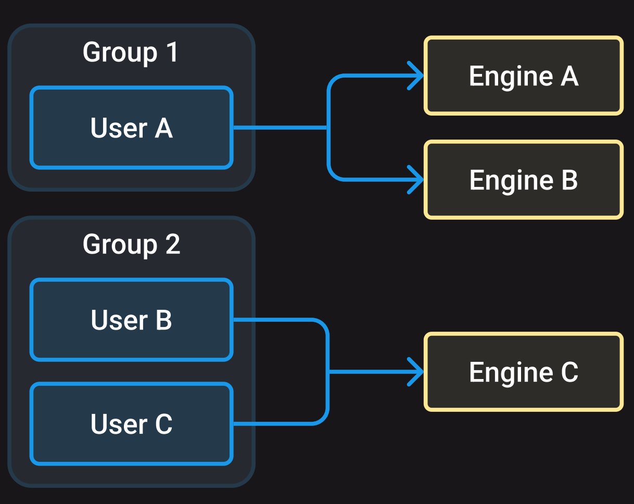
SERVER



Client requests to engines in the SERVER share level

A single engine is created for each single Kyuubi instance.

Subdomain



Example subdomain configuration for a GROUP share level

Using the `kyuubi.engine.share.level.subdomain` property, you can configure additional isolation policies for USER, GROUP, or SERVER share levels. The subdomain configuration enables creating multiple engines for a single user, group, or server.

If several engines are available, you can choose which engine to use in the JDBC connection URL.

Configuration

Share levels properties

Property	Value	Default	Description
kyuubi.engine.share.level (kyuubi.session.engine.share.level)	USER, CONNECTION, GROUP, SERVER	USER	The share level type. It can be set both in the server configuration file or in a connection URL. The latter has a higher priority
kyuubi.session.engine.idle.timeout	A number with a time units indicator	PT30 (30 min)	Time to live since an engine becomes idle. It can be set both in the server configuration file or in a connection URL. The latter has a higher priority
kyuubi.engine.share.level.subdomain (kyuubi.engine.share.level.sub.domain)	A valid ZooKeeper node name	—	Additionall engine isolation by znode. It can be set both in the server configuration file or in a connection URL. The latter has a higher priority

MapReduce architecture

Albert Bagdasaryan

Contents

To Table of Contents

Operations

Components

MapReduce algorithm

The computing core of the Hyperwave cluster is a Java-based framework MapReduce tightly bound with YARN. MapReduce provides Java classes, interfaces, and runtime environment for building and running custom applications. Through these components, a MapReduce application interacts with the YARN Resource Manager to perform its tasks on the cluster nodes where the YARN Node Manager is installed. HDFS DataNodes and YARN Node Managers are installed on the same cluster nodes. In this way, Hadoop binds together its storage and computing resources. In some cases, the names Node Manager and DataNode can be used interchangeably because they are installed on the same physical nodes.

Operations

MapReduce uses the following operations to process input data by an application request:

1. **Split**. The application defines the way to split the input files to records called **splits**. For example, if the application declares its input type as `TextInputFormat`, the MapReduce framework breaks the input data into lines.
2. **Map**. This operation is performed by the `map` method defined in the application `Mapper` class. The latter is used to create a map task replicated to all DataNodes where the input HDFS blocks are located. The MapReduce framework calls the `map` method for every split on the node. The task creates multiple key/value pairs and saves them to the local file system. The output file is partitioned by key. This is necessary for the next operation.
3. **Shuffle**. The system transfers data generated by the previous operation to those DataNodes where the next operation can be performed. For each reduce task, the shuffle service supplies all key/value pairs with a specific key.
4. **Reduce**. This operation is performed by the `reduce` method defined in the application `Reducer` class. The latter is used to create a reduce task replicated to several DataNodes and used to aggregate the data created by the map tasks. The framework calls the `reduce` method for each unique key with all of its paired data received by the node.

In the list above, a MapReduce application must define Java classes for two main stages:

- The **Map** stage is used to transform the input data into key/value pairs as defined in the application code.
- The **Reduce** stage is used to aggregate the result coming from the map stage to another key/value format and save the output to HDFS. When called, the reduce task creates a unique key/value pair, that is, it aggregates the input data.

If input data comes originally to the local file system, you need to copy it to HDFS that breaks files into HDFS blocks stored on different DataNodes. You can create a chain of MapReduce jobs where one job takes data from HDFS files created by another job. The job output data is stored in HDFS (for reliability). MapReduce and YARN are responsible for scheduling and monitoring tasks. When a task fails due to some issue in the cluster, YARN starts this task again.

Components

To implement the operations mentioned earlier, MapReduce uses the following components (most of them belongs to YARN):

- A Hadoop client is used to start a MapReduce application.
- The Resource Manager governs computing resources of the entire cluster.
- A Node Manager manages resources on a cluster node.
- The Application Master orchestrates the application lifecycle.
- A YARN container provides necessary computing resources to a map or reduce task running on the cluster node where that container exists.

MapReduce algorithm

You can start a MapReduce application as a job from a Hadoop client with a command similar to the following:

```
$ hadoop jar <jar_file_name> <class_name> <input> <output>
```

This command accepts the following arguments (this example presents not all possible arguments):

- `<jar_file_name>` is the path to the JAR file containing the application Java package.
- `<class_name>` is the name of the application main class.
- `<input>` is the path to the input file, directory, or a path template (for example, */path/to/folder/test**) in HDFS.
- `<output>` is the path to the output directory in HDFS.

MapReduce and YARN execute a MapReduce job (application) in the following order (for brevity, some operations are simplified):

1. MapReduce (as a YARN client) submits the application to the YARN Resource Manager.
2. The Resource Manager creates the MapReduce-specific Application Master (`MRAppMaster` for MapReduce applications) in a YARN container on one of the nodes managed by a Node Manager.
3. The Application Master requests the Resource Manager for computing resources on those DataNodes where the input blocks are stored. Then it requests the Node Manager on each of those nodes to create a container with a map task. The number of map tasks is the same as the number of blocks (excluding replicated blocks).
4. MapReduce prepares splits and runs the `map` method to process each split. The map process periodically sends a report to the Application Master. If a map process fails, the Application Master restarts the task.
5. The map tasks process the data and save the output to the local files system. HDFS is not used here to not overload the cluster with intermediate data.
6. After the map tasks complete their processes, the Application Master organizes containers with the reduce tasks.
7. The Application Master requests the shuffle service, which runs on the same nodes where the map tasks have stored their data, to transfer (shuffle) that data to the nodes where the reduce tasks are created.
8. The reduce tasks aggregate data in accordance with the application code and save the output to HDFS. Each task stores data in its own HDFS file.
9. The Application Master closes the job.

See Quick start with MapReduce for more information.

MapReduce overview

Albert Bagdasaryan

Contents

To Table of Contents

Map and Reduce

The MapReduce framework operates exclusively key/value pairs. Whether data is in a structured or unstructured format, the framework converts the incoming data into key/value pairs:

- Key is a reference to the input value.
- Value is a set of data to be processed.

Map and Reduce

The Map function produces a new list of key and value pairs:

- An output of Map is called intermediate output.
- Its type can be different from the input type.
- An output of Map is stored on the local disk from where it is shuffled to nodes performing the data reduce operations.

The Reduce function takes intermediate key/value pairs from the Map function and processes them. Usually reducers compute aggregation or summation operations:

- Input given to a reducer is generated by Map (intermediate output).
- The key/value pairs provided to reducers are sorted by key.

The Reduce function produces a final list of key/value pairs. It is stored in HDFS and its type can be different from the input type.

In a mapper, input data is processed by a user-defined function. The required business logic is mostly implemented at the mapper level, so that all heavy processing is done by the mapper in parallel. For this reason, the number of mappers is significantly greater than the number of reducers. Mappers generate output as intermediate data that goes as input to the reducers.

This intermediate result is processed by a user-defined function working at reducer. Then the final output is generated. This final output is stored in HDFS. The replication is done as usually.

For additional information, see Quick start with MapReduce.

Monitoring architecture in ADH

Konstantin Alpashkin

Contents

To Table of Contents

- Overview
- Architecture
  - Prometheus
  - Grafana
  - Node Exporter
  - Service exporters
  - Workflow details
- Usage recommendations

Overview

Monitoring is an ADH service designed to collect, store, and visualize cluster and service-specific metrics. It provides a single pane of glass to monitor the health, performance, and resource utilization of all nodes and cluster components.

The ADH Monitoring service is built on a popular open-source software stack:

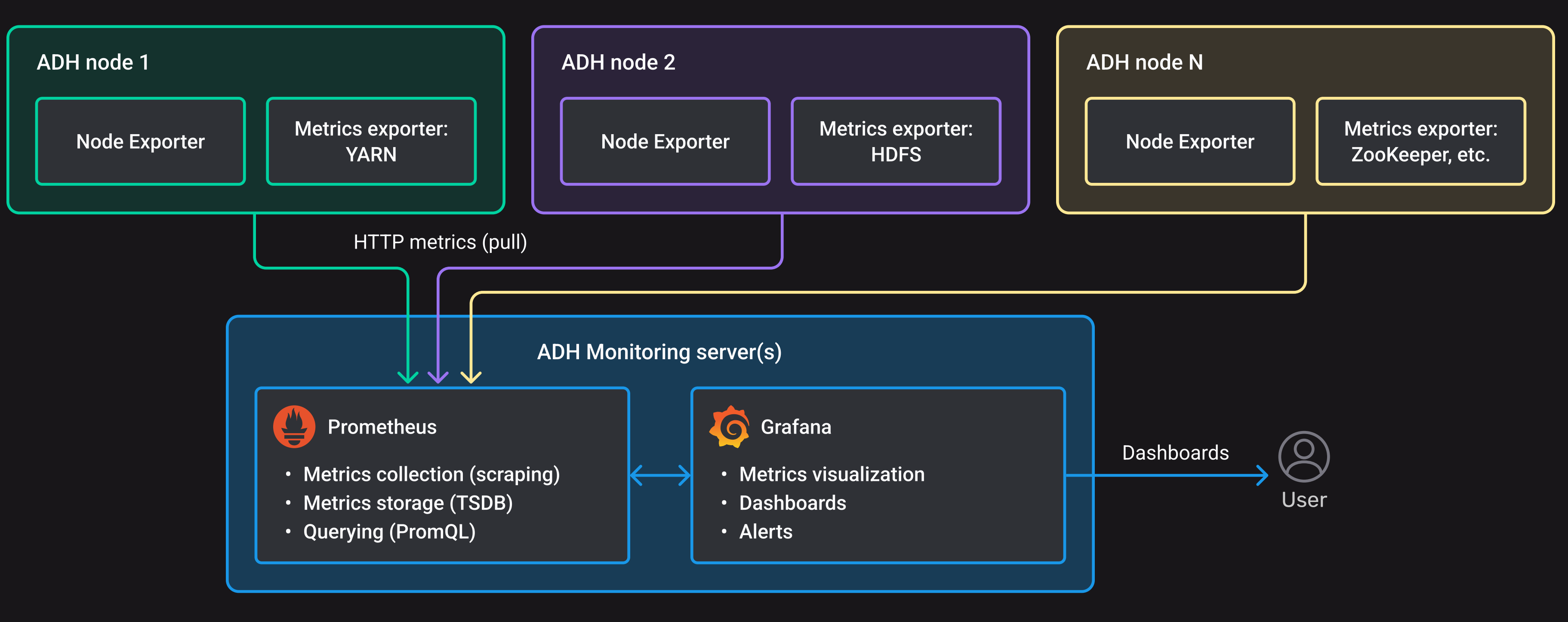
- Prometheus. A monitoring and alerting system that collects and stores metrics from applications and infrastructure.
- Grafana. A visualization and monitoring platform for visualizing metrics received from various data sources.
- Metric exporters. Lightweight agents responsible for sharing kernel-level and service-specific metrics with Prometheus.

Key principles of ADH Monitoring:

- **Centralized collection**. Prometheus acts as a central hub for collection and storage of metrics.
- **Decentralized data sources**. Metric exporters work as part of the corresponding ADH service (for example, there are exporters for HDFS, YARN, Zookeeper, etc.). The Monitoring server pulls metrics from the exporters.
- **Ready-made dashboards**. The service includes preconfigured informative Grafana dashboards for key ADH services.
- **Scalability**. A flexible architecture allows you to define new metrics to be reported and build custom dashboards without having to wait for a new release.

Architecture

The service architecture is a classic implementation of the Grafana/Prometheus stack that includes the following components deployed on one or more monitoring servers.



Monitoring architecture

> **NOTE**
>
> Although it is acceptable to install Monitoring components on ADH hosts along with other ADH services, in production, it is recommended to allocate a dedicated ADH host for Monitoring components.

The main components of the Monitoring service are described below.

Prometheus

Prometheus is the backbone of the Monitoring service, an open-source monitoring and notification system. It is responsible for collecting, storing, and processing metrics.

Key features:

- **Multidisciplinary data model**. Metrics are identified by a name and a set of key/value pairs (labels).
- **HTTP pull model**. Prometheus pulls metrics from target endpoints over HTTP.
- **Powerful PromQL query language**. Allows running complex aggregations and metrics analysis.
- **ADH configuration**. The Prometheus distribution that comes with the Monitoring service is preconfigured to collect metrics from main ADH services and Node Exporters.

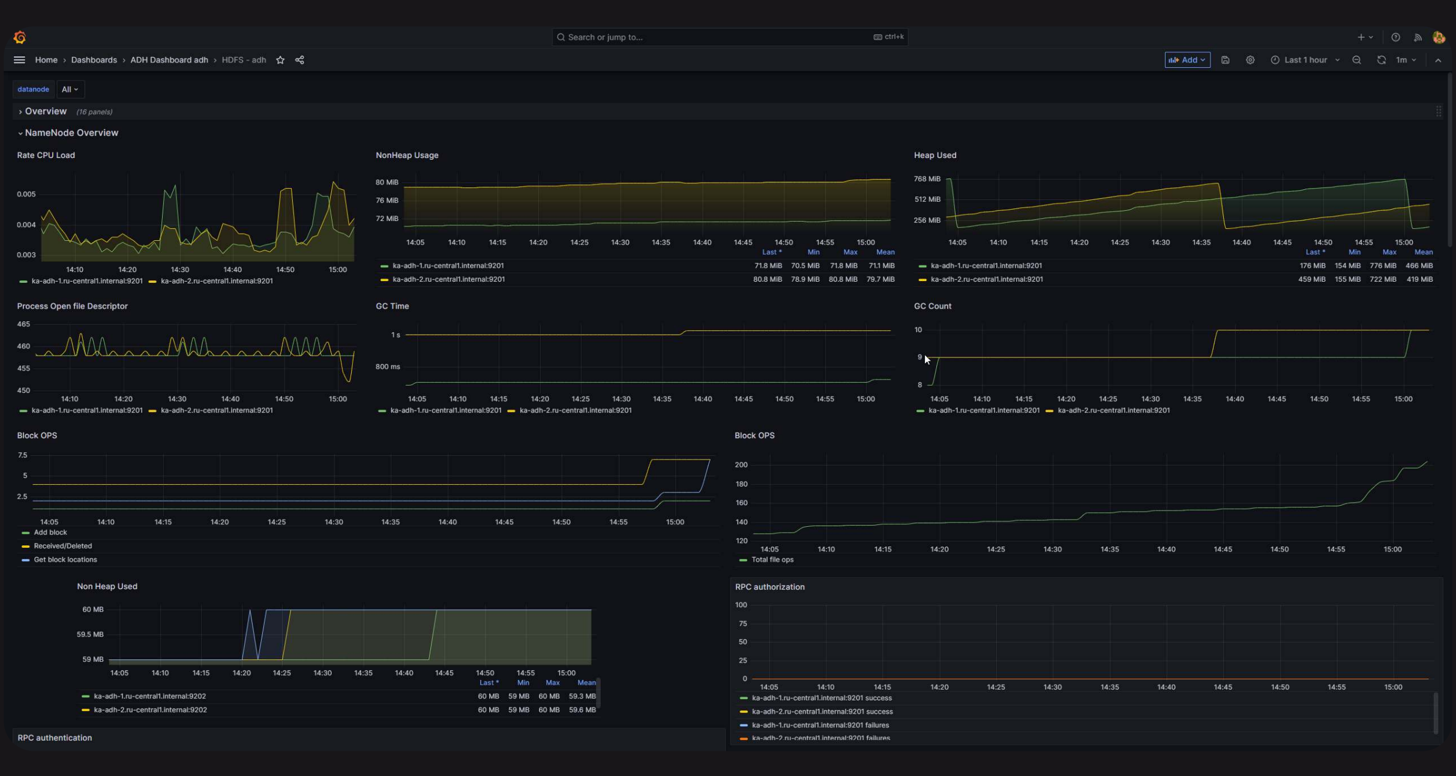
Grafana

Grafana is an open-source analytics and monitoring platform designed for data visualization. It is responsible for visualization of metrics stored in Prometheus through informative dashboards.

Key features:

- **Support for multiple data sources**. Grafana can work not only with Prometheus but also with other sources of metrics.
- **Rich visualization features**. It provides preconfigured graphs, charts, tables, heatmaps, etc.
- **Interactive dashboards**. Allows you to create dynamic dashboards with variables and switches.
- **Rich customization options**. You can edit provided dashboards and create new that suit your needs.
- **ADH configuration**. The Grafana distribution bundled with ADH includes a set of preconfigured dashboards for monitoring key services (HDFS, YARN, Zookeeper, etc.), as well as dashboards for monitoring the state of ADH hosts.

Grafana web UI is presented below. The up-to-date web UI link is available in ADCM (***Services*** → ***Monitoring*** → ***Info***).



Grafana web UI

Node Exporter

Node Exporter is the official Prometheus exporter for collecting hardware and operating system metrics. Node Exporters run on every host of an ADH cluster as tiny agents and report kernel-level metrics.

Key features:

- **Collecting OS metrics**. Node Exporter provides metrics on CPU usage, memory, disk I/O, disk space usage, network traffic, and many more.
- **API access**. Serves metrics to external systems through an HTTP endpoint.
- **Extensibility**. You can configure a Node Exporter to collect specific metrics by using pluggable collectors.
- **Deployment in ADH**. Node Exporters run as system daemons on each node of the ADH cluster, ensuring uniformity in the collection of host metrics.

Service exporters

A service exporter is an agent that works as part of an ADH service (HDFS, YARN, Hive, etc.) and is responsible for providing the metrics specific to that service in a Prometheus-compatible format. Some ADH services come with a JMX exporter, which shares metrics directly from the service JVM via JMX. Other services share metrics through a fixed HTTP endpoint.

Unlike Node Exporters, service exporters are not part of the Monitoring service, but are critical to its operation. Service exporters are installed and configured automatically when an ADH service is added to a cluster. For example, when installing HDFS NameNode/DataNode components, HDFS exporter is deployed and configured too.

> **NOTE**
>
> After adding a new ADH service, the Monitoring service needs to restart to get metrics from the new service.

Below is a list of preconfigured exporters for ADH services.

ADH service	Exporter type	Configuration location (ADCM)
HDFS	jmx-exporter	The ***Monitoring*** configuration group for DataNode, JournalNode, and NameNode service components
Ozone	Exposes metrics in the Prometheus format through a REST endpoint	***Services*** → ***Ozone*** → ***Primary Configuration*** → ***ozone-site.xml*** → ***hdds.prometheus.endpoint.enabled***
Impala	Exposes a custom HTTP endpoint (*/metrics_prometheus*) with metrics in the Prometheus format	—
Trino	jmx-exporter	The ***Monitoring*** configuration group for each component of the service
Hive	jmx-exporter	The ***Monitoring*** configuration group for Hive HiveServer2 and Hive Metastore components of the service
YARN	jmx-exporter	The ***Monitoring*** configuration group for Node Manager and Resource Manager components of the service
HUE	Exposes a custom HTTP endpoint with metrics in the Prometheus format	***Services*** → ***HUE*** → ***Components*** → ***HUE Server*** → ***Primary Configuration*** → ***hue.ini*** → ***desktop.enable_prometheus***
Zeppelin	Exposes a custom HTTP endpoint with metrics in the Prometheus format	***Services*** → ***Zeppelin*** → ***Primary Configuration*** → ***zeppelin-site.xml*** → ***zeppelin.metric.enable.prometheus***
HBase	jmx-exporter	The ***Monitoring*** configuration group for each component of the service
Solr	solr-exporter	All exporter settings are located on the ***Solr prometheus exporter*** component
Kyuubi	Exposes a custom HTTP endpoint with metrics in the Prometheus format	The following parameters are available in ***Services*** → ***Kyuubi*** → ***Components*** → ***Kyuubi Server*** → ***kyuubi-defaults.conf***: ▪ ***kyuubi.metrics.enabled*** ▪ ***kyuubi.metrics.reporters*** ▪ ***kyuubi.metrics.prometheus.port*** ▪ ***kyuubi.metrics.prometheus.path***
Zookeeper	Exposes a custom HTTP endpoint with metrics in the Prometheus format	The following parameters are available in ***Services*** → ***Zookeeper*** → ***Primary configuration*** → ***zoo.cfg***: ▪ ***metricsProvider.className*** ▪ ***metricsProvider.httpPort***

Workflow details

1. Node Exporters run on every physical or virtual node of an ADH cluster and collect OS metrics (CPU, RAM, disk and network I/O).
2. On every ADH host where ADH services are running, the corresponding service exporters are launched (for example, a JMX exporter for the HDFS DataNode component).
3. Prometheus, which is running on a dedicated host, periodically polls (scrapes) all metrics exporters by sending HTTP requests to their endpoints (for example, *http://<ADH_host_name>>:9100/metrics* for Node Exporter).
4. Prometheus receives metrics in a text format, encodes and stores them in a database on a local disk.
5. Grafana connects to the Prometheus API to receive and visualize metrics data.
6. Visualized metrics are presented to the end user in the form of informative dashboards in Grafana web UI.

Usage recommendations

For better stability and performance of the Monitoring service, consider the following recommendations:

- **Separation of roles**. To increase fault tolerance and improve the performance in large clusters, it is recommended to install the Monitoring service on dedicated ADH nodes that do not host main computing services.
- **Setting up alerts**. Use the built-in Prometheus Alertmanager to set up alert rules. You can integrate alerts with popular notification channels like email, Slack, etc. to quickly react to incidents. These components are not included in the ADH distribution and require a standalone configuration.
- **Storage Management**. Keep track of the amount of disk space allocated for Prometheus data storage. Set up retention policies to meet your requirements on the size of the metrics stored.
- **Customization of dashboards**. By using preconfigured Grafana dashboards bundled with ADH as a base, you can create custom dashboards to monitor specific parameters, which are critical for your workloads and business processes.
- **Track the Monitoring service health**. Do not forget to monitor the health and performance of the Monitoring service. For this, you can use metrics that Prometheus collects about itself.
- **Service configuration**. When changing the configuration of service exporters (adding new ADH services, changing service topologies, etc.), it is mandatory to restart the Monitoring service to apply the new configuration.

Ozone architecture

Elena Kostyuchenko

Contents

To Table of Contents

- Overview
- Concepts
- Components
- Data operations
  - Write operation
  - Read operation

Overview

Apache Ozone (O3) is a distributed key/value object storage that is optimized for working with both Hadoop services and S3 storages. It's highly scalable, can share the same cluster and security policies with HDFS, and is capable of effectively handling files no matter the size.

Ozone's main advantages:

- Unlike the HDFS file system, Ozone is designed to work effectively with a big number of small files.
- Integrates well with Hadoop services, such as Hive, Spark, MapReduce, Kerberos, and the Apache Ranger framework.
- Supports working with remote object storages via Amazon S3 API.
- Able to work with container environments such as Kubernetes and YARN.

Concepts

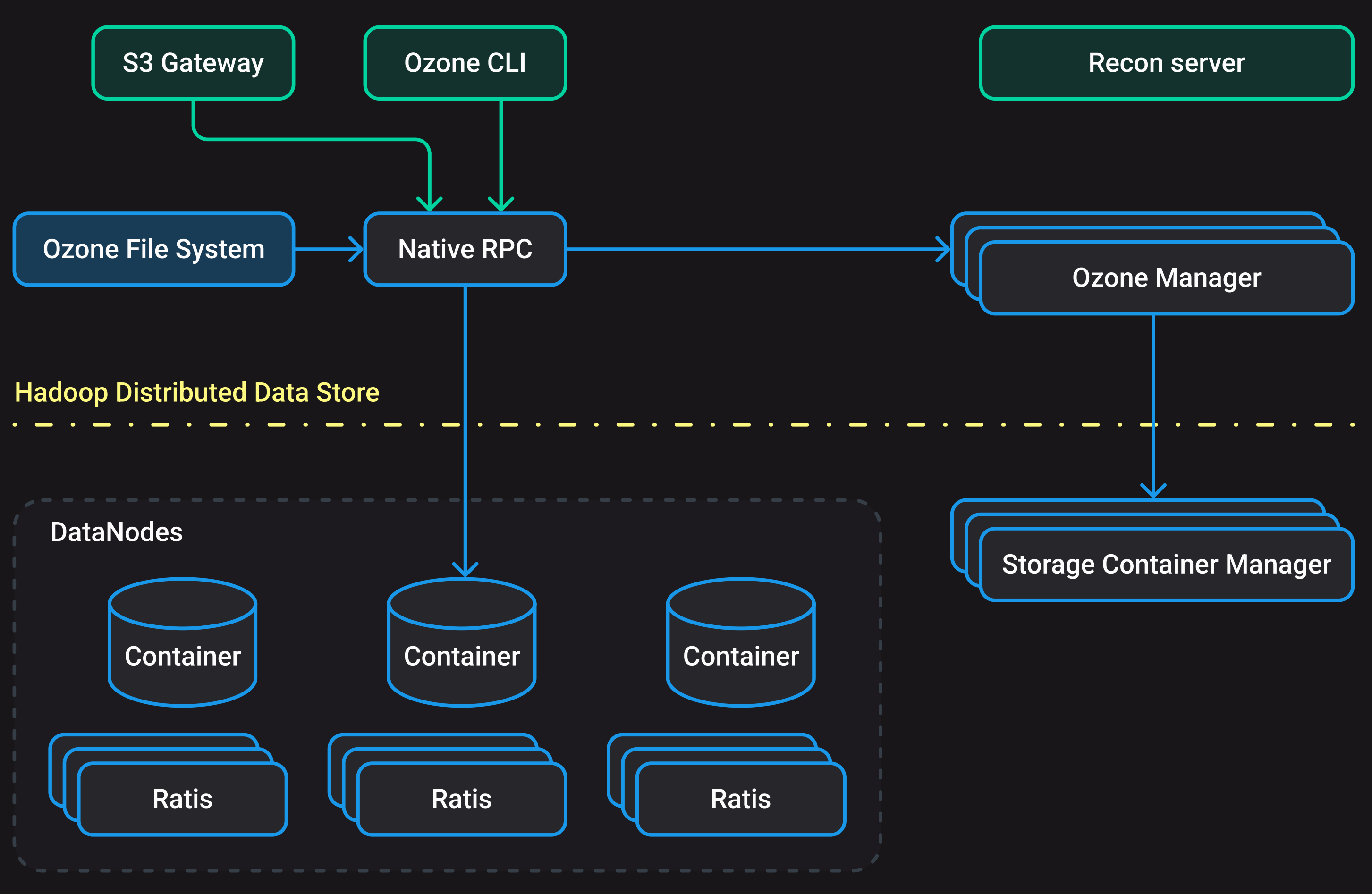
Ozone is based on the following components:

- **Blocks**
  Same as in HDFS, blocks are the basic unit of storage. Each file in the system is comprised of blocks. In Ozone, the default block size is 256 MB.
- **Containers**
  A container is a group of blocks that is replicated as a single unit.
- **Keys**
  A key is a marker that indicates which blocks belong to a particular file. The keys act as file IDs for all data operations by clients.
- **Buckets**
  A bucket is a directory in terms of Ozone. Buckets can contain any number of keys, but cannot contain other buckets.
- **Volume**
  A volume is a home directory of a user in Ozone, which can only be created by an administrator. Ozone namespace can contain any number of volumes. Once a volume for a user has been created, the user can create buckets that belong to that volume.

Components

Ozone architecture is based on the following components:

- **Ozone Manager**
  A system service that manages the namespace in the same manner as the HDFS NameNode. It records the blocks located on the DataNodes and provides the IDs of the blocks when requested.
- **Ozone Storage Container Manager**
  A system service that manages the containers. It allocates blocks and assigns them to DataNodes. Clients read and write these blocks directly. Container Manager keeps track of all the block replicas. If there is a loss of data or a disk, Container Manager detects it and instructs data nodes to make copies of the missing blocks to ensure high availability.
- **Ozone DataNode**
  The Ozone service that aggregates blocks into storage containers.
- **Ozone Recon**
  Ozone Recon Server interacts with all other Ozone components and provides a unified management API and user interface.



Ozone architecture

Data operations

Ozone has three main functional layers:

- The metadata data layer, controlled by Ozone Manager and Storage Container Manager.
- The data storage layer, controlled by Ozone DataNodes.
- The replication layer, controlled by Apache Ratis and used to replicate metadata.

Below is the description of how these layers interact during read and write operations.

Write operation

In Ozone, a write operation goes through the following steps:

1. To write a file to Ozone, a client makes a request to Ozone Manager. The request contains information about the bucket and volume, where the key will be created.
2. To allocate the blocks for the new key, Ozone Manager sends a request to Storage Container Manager.
3. Storage Container Manager allocates the blocks on the DataNodes and returns their IDs to Ozone Manager.
4. Ozone Manager records the metadata and returns the block token (a security permission to write data to the block) to the client.
5. The client uses the block token and writes data to the DataNode.

Read operation

In Ozone, a read operation goes through the following steps:

1. The client requests the list of blocks, which belong to the key, from the Ozone Manager.
2. Ozone Manager returns the block list and the block tokens.
3. Client connects to the DataNode and reads the data.

HDFS vs Ozone

Sergei Tikhomirov

Contents

To Table of Contents

Feature comparison
Pros and cons
- HDFS
- Ozone
Use cases

Both Ozone (O3) and HDFS are open-source suitable distributed Hadoop storages, but there are several key differences which are discussed in this article.

Feature comparison

Key feature comparison is presented in the table below.

Feature	HDFS	Ozone
Data model	A file-based storage system where data is stored in files and directories	An object store that works with large amounts of unstructured data and is optimized for cloud
Data replication	Replication among DataNodes to ensure fault tolerance by default	Software-defined storage that allows for custom data replication policies and data redundancy
Scalability	Good scalability for handling massive processing jobs	Designed to provide even better scalability than HDFS
Namespace management	Single namespace for the entire cluster	Multiple namespaces for different use cases
Object storage	No	Yes
Support for S3 and other object storage protocols	No	Yes
Access control	POSIX-style permissions	S3-style permissions and bucket-level access controls
Authentication and authorization	Kerberos	Kerberos, Ozone Token
Data consistency	Eventual consistency	Strong consistency due to protocols like RAFT

Pros and cons

HDFS

HDFS is the default file system in Hadoop, and it has the following pros:

- massive data storage support;
- quick detection and response to hardware failures;
- support for data streaming;
- simplified consistency model;
- high fault tolerance and easy recovery;
- designed for commercial hardware.

However, there are also some disadvantages to it:

- not suitable for a large number of small files;
- doesn't support file modification (HDFS 2.x supports appending content to files);
- struggles with over 400 million files;
- doesn't support parallel writing.

Ozone

With HDFS' cons leading to a big discomfort with modern big data storage needs, a new solution had to be implemented with the following key advantages:

- strong consistency;
- designed to store more than 100 billion objects in a single cluster;
- great scalability due to layered architecture;
- just as fault tolerant and easily recoverable as HDFS;
- can work alongside with HDFS on the same hosts.

Since the project is rather new, there are some cons:

- little deployment cases to learn from;
- designed to integrate with Hadoop ecosystem, it's still not widely supported, and some services may require additional configuration to work with Ozone;
- no local socket, and overall performance is slower.

Use cases

Apache Ozone is advantageous over HDFS in environments requiring scalability for small files, S3 compatibility, or cloud-native capabilities. However, HDFS remains suitable for Hadoop workloads with fewer demands for storage of small files without the possibility of combining them or cloud integration.

Phoenix overview

Vladimir Adamenkov

Contents

To Table of Contents

- Introduction
- Features
- Architecture
- Phoenix in ADH
- Service interactions
  - HBase
  - ZooKeeper
  - MapReduce
  - Spark
  - Hive
- Implementation details and key operations
  - Index management
  - Data storage format
  - Salting
  - UPSERT operation
- Use cases
- Benefits
- Limitations and considerations

Introduction

Apache Phoenix is an SQL layer over HBase represented by a relational database engine.

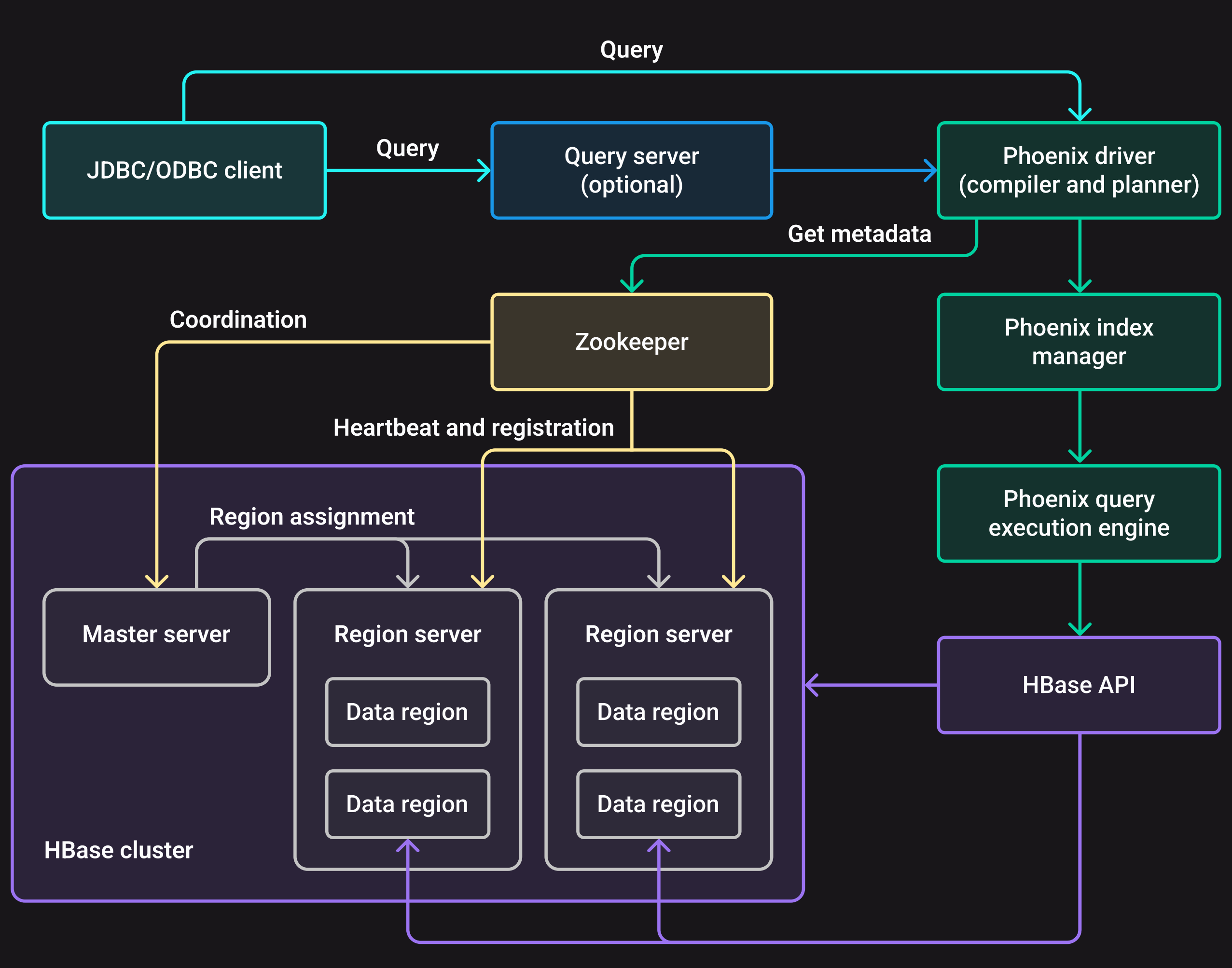
Features

Phoenix main features are:

- Low latency and high performance achieved through parallelism, optimized query execution, and secondary indexing.
- Bulk data loading (e.g. using MapReduce).
- Schema management: data schemas and metadata are stored in a centralized catalog, and the columns can be added and deleted.
- Multi-tenancy: multiple users or applications can share a single HBase cluster while maintaining data isolation and security.
- Use of salted tables.
- Support of atomic operations and transaction-like behavior for data consistency.
- Seamless integration with Hadoop and Spark.

Architecture

The general architecture of Phoenix is shown below.



Phoenix architectural schema

In this schema, the data flows the following way:

1. An SQL query is submitted through a JDBC/ODBC client.
2. A query server (for example, Apache Calcite Avatica), if used, acts as a mediator between the client application and the core Phoenix engine. It performs the following functions:
   - Listens for incoming SQL queries from the client applications over the network through the RESTful API.
   - Optionally, controls access to Phoenix by using authentication and authorization mechanisms.
   - Manages connections to the Phoenix/HBase cluster by pooling connections. It helps to reduce the overhead of establishing new connections for each query.
   - Forwards the SQL queries to the Phoenix driver.
3. The query is sent to the Phoenix driver which transmits the metadata to ZooKeeper. The compiler and planner components within the Phoenix driver parse the query, validate it, then optimize it, and generate an execution plan.
4. If possible, Phoenix index manager determines if any indexes can be used for further optimization of the query.
5. The query engine executes the plan, interacting with HBase through the HBase API.
6. HBase retrieves the data from the relevant data regions.
7. Phoenix processes the data (filtering, aggregating, etc.) and returns the results to the client the same way.

Phoenix in ADH

As was already said, Phoenix is fundamentally built on top of HBase. It is an SQL layer that leverages HBase strengths — scalability, fault tolerance, NoSQL data storage. While Phoenix focuses on query processing, the underlying data resides in HBase, which stores its data in HDFS. Phoenix uses HBase to access this data.

Phoenix can be used in conjunction with other Hadoop ecosystem components, such as Spark and Hive. This integration allows you to leverage these tools for more complex data processing and analytics.

Service interactions

Phoenix interacts with different services in ADH.

HBase

The most crucial interaction is with HBase. Phoenix uses the HBase client API extensively for all read and write operations. It translates SQL queries into HBase operations, such as `get`, `scan`, `put`, `delete`, and others.

ZooKeeper

Phoenix relies on ZooKeeper for the following:

- **HBase cluster discovery**. Phoenix clients and the query server (if present) use ZooKeeper to discover the location of the HBase Master and Region servers.
- **Leader election**. In some scenarios, ZooKeeper is used for leader election among Phoenix instances.
- **Coordination of metadata**. Management of the distributed metadata about Phoenix tables, indexes, and schemas.

MapReduce

Phoenix may interact with MapReduce for the following:

- **Bulk loading**. Phoenix provides a mechanism for bulk loading data into HBase using MapReduce. This is more efficient than inserting rows one at a time using SQL `INSERT` or `UPSERT` statements.
- **Custom processing**. You can write custom MapReduce jobs that read data from Phoenix tables, perform transformations, and write the results back to Phoenix or another data store.

Spark

Phoenix may interact with Spark for the following:

- **Phoenix Spark connector**. This connector allows Spark applications to read and write data to Phoenix tables. Spark can be used for complex data transformations and analytics that are difficult or inefficient to perform directly in Phoenix.
- **Spark SQL**. You can use Spark SQL to query data in Phoenix tables, leveraging Spark distributed query engine.

Hive

Phoenix can interact with Hive indirectly. Data loaded into HBase using Phoenix can potentially be accessed by Hive using storage handlers, although this is less common than using Spark.

Implementation details and key operations

Index management

When creating an index, Phoenix creates a separate HBase table to store the index data. Writes to the base table automatically trigger updates to the corresponding indexes (for mutable indexes). During a query execution, the query engine analyzes it and determines if an index can be used to speed up the query.

Data storage format

Phoenix stores data in HBase in a row-oriented format. The row key is typically composed of one or more columns from the Phoenix table. Column families are used to group related columns together.

Salting

To prevent hotspotting on a single Region server, the salting mechanism is used.

UPSERT operation

The `UPSERT` statement in Phoenix is used to either insert a new row into a table or update an existing row. If a row with the specified primary key already exists, the `UPSERT` statement will update the existing row. If the row does not exist, the `UPSERT` statement will insert a new row.

Use cases

The following are the examples of real world scenarios where the use of Phoenix is particularly beneficial:

- **Operational analytics**. Phoenix excels at performing real-time or near-real-time analytics on operational data stored in HBase. This is in contrast to traditional batch-oriented data warehousing. Examples include:
  - Real-time monitoring — monitoring system performance, application activity, or network traffic.
  - Fraud detection — identifying fraudulent transactions in real-time.
  - Personalization — providing personalized recommendations based on user behavior.
- **Time-series data analysis**. HBase is often used to store time-series data (e.g. sensor readings, stock prices, log events). Phoenix can be used to query and analyze this data efficiently.
- **Internet of Things (IoT)**. Phoenix is well-suited for processing data from IoT devices. This data is typically high-volume and low-latency.
- **Clickstream analysis**. Phoenix can be used for analyzing website clickstream data to understand user behavior and improve website design.
- **Telecom**. Phoenix can be used for network monitoring, call detail record (CDR) analysis.
- **Data warehousing**. While not a replacement for a full-fledged data warehouse, Phoenix can be used for smaller-scale data warehousing scenarios where real-time query performance is important.
- **Serving layer for machine learning**. Phoenix can act as a serving layer, providing real-time access to features for machine learning models.

Benefits

Use of Phoenix can deliver the following advantages:

- **SQL familiarity**. The biggest benefit of Phoenix is the use of SQL. This makes HBase accessible to a much wider audience, as SQL is a widely known and understood language.
- **Performance**. Phoenix provides optimized query execution on HBase, often achieving much better OLTP performance than other SQL-on-Hadoop solutions.
- **Low latency**. Phoenix is designed for low-latency queries, which makes it suitable for real-time or near-real-time applications.
- **Scalability and fault tolerance**. Phoenix leverages the scalability and fault tolerance of HBase.
- **Secondary indexes**. Phoenix enables faster queries by using secondary indexes.
- **Transaction Support**. Phoenix ensures data consistency and reliability with ACID transactions.
- **Integration with Hadoop ecosystem**. Phoenix seamlessly integrates with other Hadoop components, such as Spark, MapReduce, and Hive.
- **Reduced development effort**. Phoenix simplifies development by providing an SQL interface to HBase, reducing the need for complex HBase API programming.

Limitations and considerations

Phoenix has the following limitations and considerations:

- Phoenix does not support the ANSI SQL standard in its entirety. Refer to the Apache Phoenix documentation for details.
- Managing secondary indexes can be complex, especially for large tables. You need to carefully consider which columns to index and how to maintain the indexes.
- The NoSQL nature of HBase influences how data is modeled in Phoenix. Relational database modeling practices may not always be directly applicable.
- If the row key is not designed properly, writes can be concentrated on a single Region server, leading to hotspotting and performance degradation. Salting can mitigate this.

Solr overview

Konstantin Alpashkin

Contents

To Table of Contents

Indexing

Querying

Ranking the results

Solr is a search server, and it deals with large sets of data. Since Solr can also store data, it is a NoSQL, Non-Relational storage, and a processing technology.

When you submit a query to the Solr search server, it separates queries into different pieces/entities, then matches the query against the document's inverted index that was created earlier. The Solr search server returns a set of documents as a response based on the similarity in class or other characteristics defined in the *schema.xml* and *solr.config* files.

Solr follows a three-step process of indexing, querying, and ranking.

Indexing

There are various methods using which Solr indexes documents and other rich text-based data. Solr allows users to directly upload their documents in PDF, CSV, XML formats, and the system can read and index data from these sources automatically. Further, it can also upload texts and documents from email and attachments.

Solr uses an inverted index to store data where it uses keyword-centric rather than page-centric data structure; a simpler way to understand the concept is how words are indexed at the end of any book where the word on the page is mentioned along with its meaning. Hence, it can achieve a faster response time and gives relevant search results in no time.

Querying

A query can be anything like searching for text, images, or geolocation. When a query is sent, Solr processes it with a query handler which returns the document from the Solr index.

Ranking the results

As the system is matching the query with the data from the indexed files based on keywords, it ranks the results based on the relevance. This process creates a hierarchy of results based on relevance.

Spark Connect

Konstantin Alpashkin

Contents

To Table of Contents

Overview

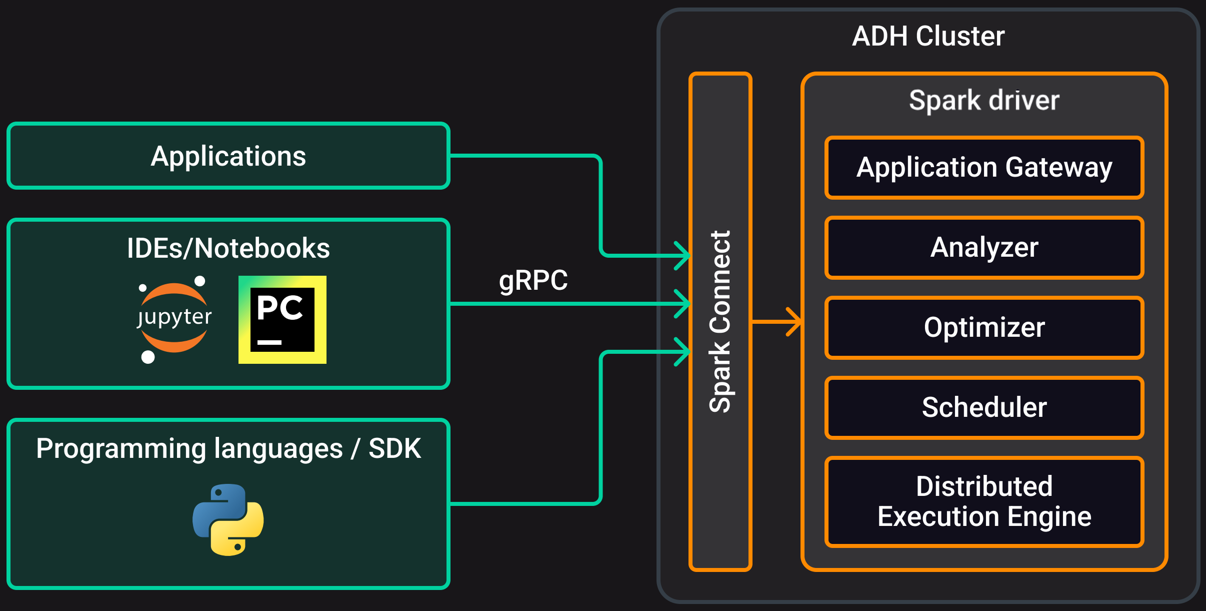
How Spark Connect works

Spark Connect benefits

Overview

Spark Connect is a component of the Spark3 ADH service that acts as a thin client to provide connectivity to Spark clusters from a remote location. Using Spark Connect, you can work with your Spark cluster from a remote location, for example, using your favorite IDE on a user-grade laptop.

The following diagram illustrates the high-level Spark Connect architecture.



Spark Connect architecture

How Spark Connect works

Spark Connect implements the decoupled client-server architecture. The server side is the Spark Connect ADH service that listens for commands from clients. A client can be represented by a Spark application/Spark shell instance that uses the Spark Connect client library to speak to the server.

The main feature of Spark applications that use Spark Connect is the creation of a **remote** Spark session object in the application code. A remote session object establishes a connection to the Spark Connect server and automatically maintains the communication with the server — it sends DataFrame operations to the server and receives the results of their execution. Once a remote Spark session is created, it is used in the same way as a regular Spark session object.

Spark Connect uses the DataFrame API and unresolved logical plans as a protocol to communicate with the Spark cluster. The Spark Connect client library translates DataFrame operations into unresolved logical plans requests, encodes them, and then sends to the Spark Connect server over gRPC. The Spark Connect server receives and converts the unresolved logical plans into Spark logical plan operators. From this point, the standard Spark execution process kicks in, including all the Spark's optimizations and features.

For more information on how to work with Spark via Spark Connect, see Spark Connect Usage.

Spark Connect benefits

The major operational benefits of using Spark Connect are as follows:

- **Lightweight client environment**. You can work with your massive Spark cluster from your favorite IDE/Jupyter notebook on an ordinary user-grade laptop. The major requirement is to use a remote Spark session instance that communicates with the Spark Connect server using the client library.
- **Better resource isolation**. Working with Spark via Spark Connect creates isolated sandboxes on the ADH hosts with resources allocated in advance. Spark executors that belong to different applications coexist within these isolated sandboxes without competing for resources. This helps to avoid cases of massive resource consumption, for example, when a heavy Spark computation consumes all available RAM on the cluster node (e.g. due to misconfigured startup parameters).
- **Upgradability**. Spark driver can be updated separately, regardless of the client part. For example, you can update your Spark core to pick up the latest security fixes without modifying your client applications. The only requirement is that gRPC interfaces of Spark Connect client/server must remain compliant after the update.

SSM architecture

Elena Kostyuchenko

Contents

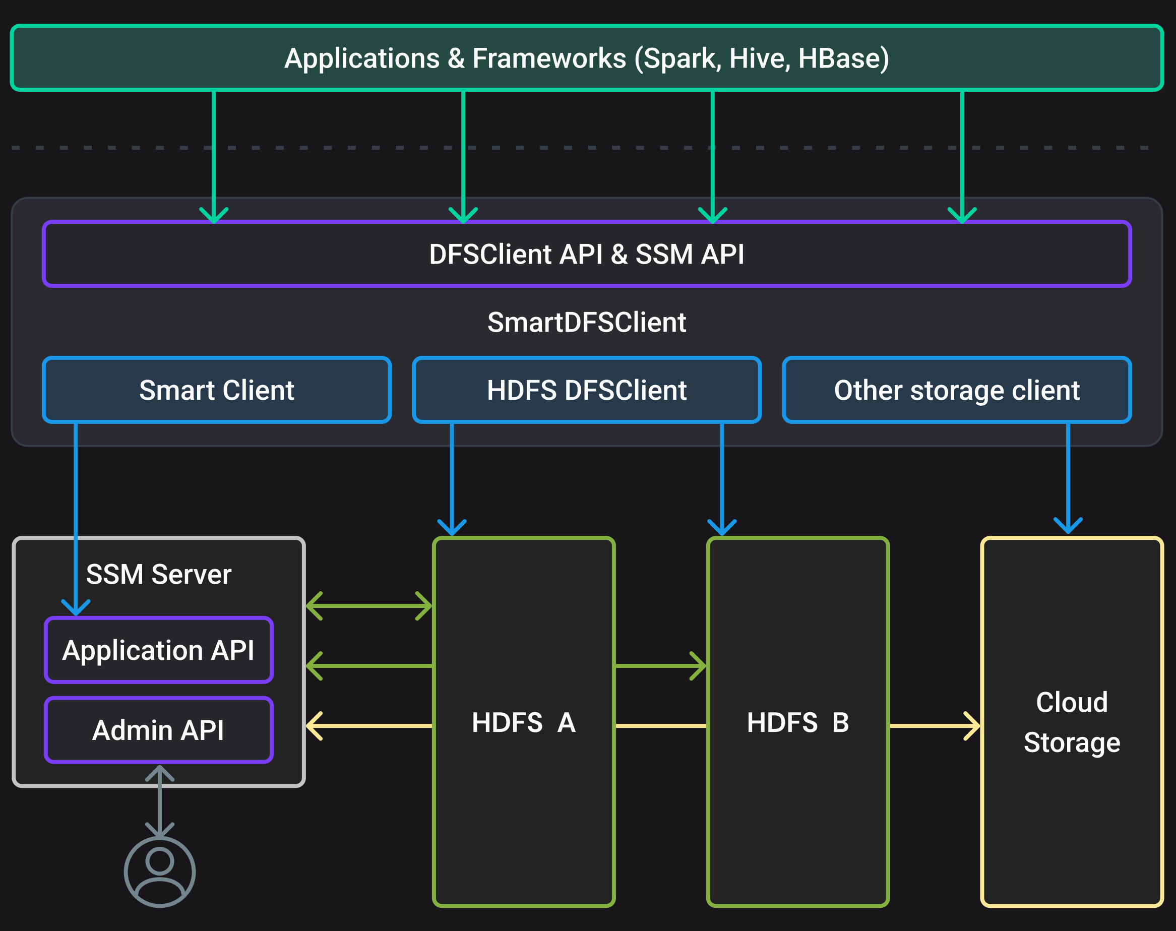
To Table of Contents

- Overview
- SSM Server
  - Rules
  - Actions
- SSM Agent
- SSM Client
- SSM Metastore

Overview

SSM (Smart Storage Manager) is an optimization software that analyzes HDFS metadata and helps to find a more efficient way to store and operate data in compliance with user-created rules and existing data policies.

SSM monitors clusters by collecting data metrics, such as file access count, and marks data according to its temperature: popular data is HOT, rarely accessed data is COLD, and somewhere in the middle is WARM data. The precise criteria for data labeling is provided with rules set by users.



Applications & Frameworks (Spark, Hive, HBase)
DFSClient API & SSM API
SmartDFSClient
Smart Client
HDFS DFSClient
Other storage client
SSM Server
Application API
Admin API
HDFS A
HDFS B
Cloud Storage

SSM architecture

SSM introduces five new features to HDFS:

- **Data moving** — move data between different storage types depending on existing rules.
- **Asynchronous replication** — replicate data in current HDFS cluster to other HDFS clusters or cloud storage. SSM monitors data modification operations (e.g. create, delete, append, and rename) on HDFS to achieve real-time data synchronization and avoid MapReduce computational costs.
- **Small files optimization** — compact a certain number of small files into a container file. The container file is stored in HDFS, but it's transparent to upper applications.
- **Erasure coding (EC)** — set rules for converting between EC files and replication files, or between any pair of files with different EC policies.
- **Compression** — compress data in HDFS without impeding access to it for external applications.

SSM provides a web UI, where users can submit rules and actions, check execution status and cluster metrics.

You can manage SSM and its components using ADCM or by manually updating its configuration files. For more information on SSM configuration properties, see the Configuration parameters page.

SSM Server

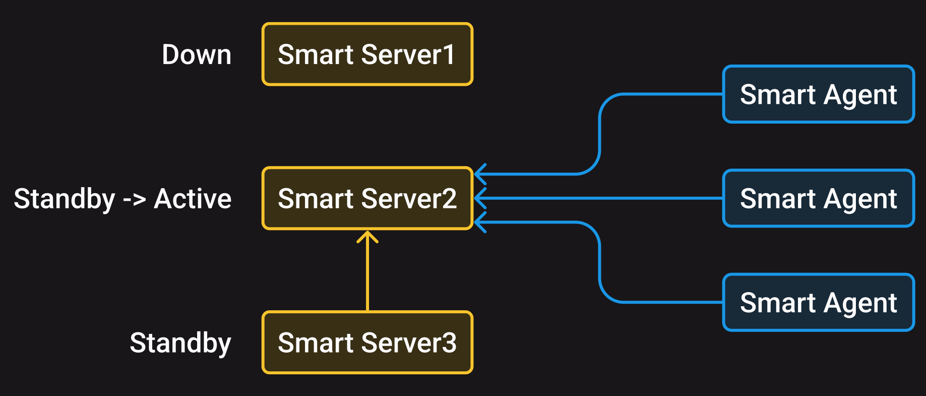
SSM Server (Smart Server) is the main SSM component. It's responsible for collecting metadata from the NameNode, implementing rules, and performing actions.

The server polls edits log from the NameNode, analyzes it and stores in its SQL database — **SSM Metastore**. If the information pulled from the cluster triggers a certain rule, the server will start the appropriate action.

A user can interact with the server via APIs:

- Admin API
- Applications API

To achieve high availability (HA), it is possible to deploy additional Smart Servers. In this case, the redundant servers will act as Standby servers.



Down
Smart Server1
Smart Agent
Standby -> Active
Smart Server2
Smart Agent
Smart Agent
Standby
Smart Server3

SSM Server failover scenario

If the active server is down, one of the Standby servers becomes the active one.

Rules

SSM rules are scripts that users can create to automate data management tasks. The rules are written in a domain-specific language and have four parts: object, trigger, condition, and command.

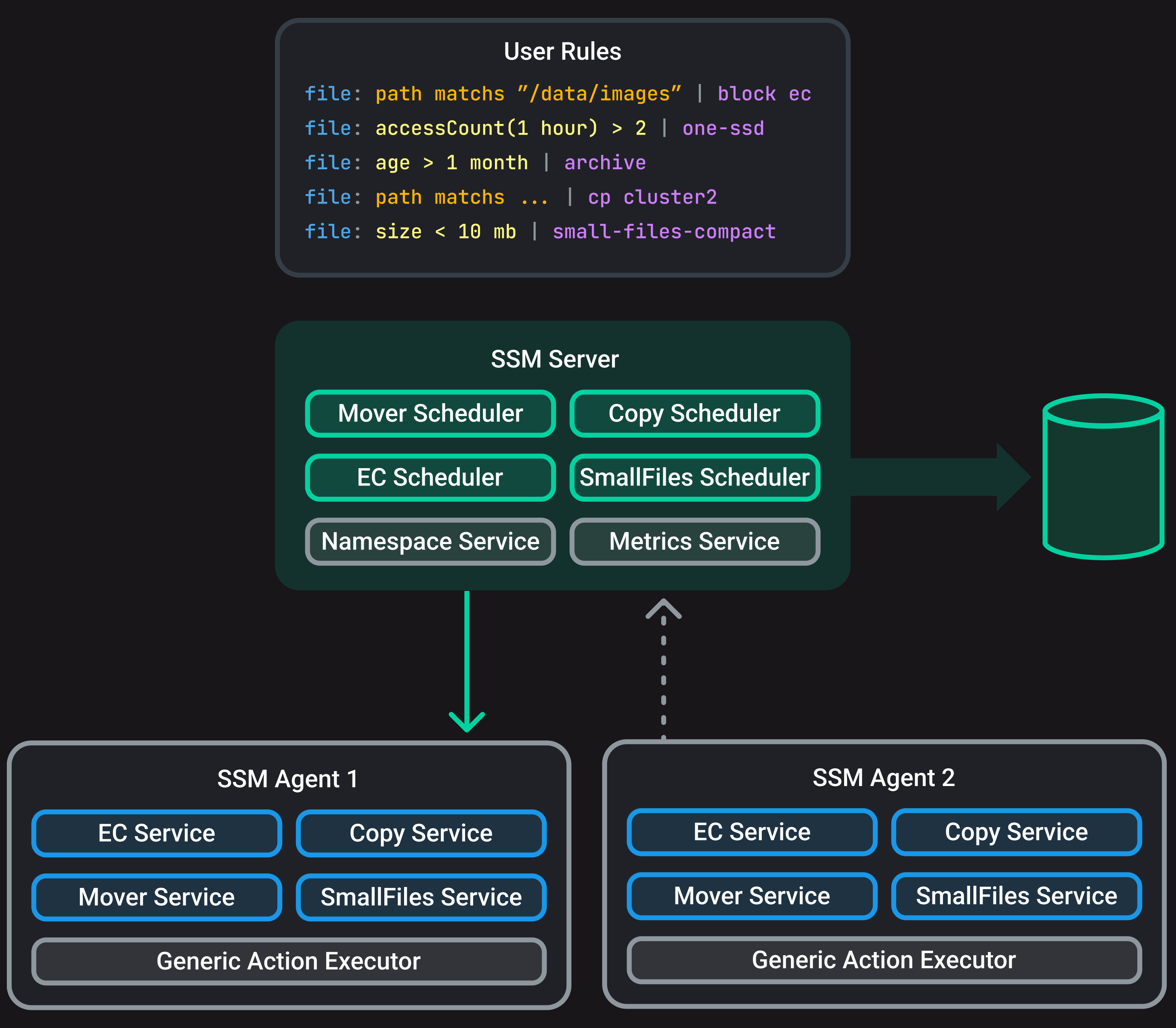
For example, if certain data has been accessed over 3 times during the last 10 minutes, a user can categorize such data as hot data, and submit a customized rule, according to which SSM will automatically move such data to a faster storage medium.

For more information on how to create rules in SSM, see the Define actions in SSM article; and for more usage examples, see SSM rules usage examples.

Actions

SSM action is a task executed by SSM when a condition is met or if started by user. Each action is performed by a dedicated scheduler service:

- Mover Scheduler — responsible for operation related to moving the data;
- Copy Scheduler — replicates the data;
- EC Scheduler — manages the erasure coded files;
- SmallFiles Scheduler — manages the containers with small files;
- NameSpace service — parses the edits logs;
- Metrics service — collects access count information.



User Rules
SSM Server
Mover Scheduler
Copy Scheduler
EC Scheduler
SmallFiles Scheduler
Namespace Service
Metrics Service
SSM Agent 1
EC Service
Copy Service
Mover Service
SmallFiles Service
Generic Action Executor
SSM Agent 2
EC Service
Copy Service
Mover Service
SmallFiles Service
Generic Action Executor

```
file: path matchs "/data/images" | block ec
file: accessCount(1 hour) > 2 | one-ssd
file: age > 1 month | archive
file: path matchs ... | cp cluster2
file: size < 10 mb | small-files-compact
```

SSM components interaction

More information about actions in SSM is available in the Define actions in SSM article.

SSM Agent

Smart Agent is a dedicate process to execute tasks dispatched by SSM Server. To expand SSM's task processing capability, you can deploy multiple Smart Agents in the cluster.

SSM Client

SSM Client (SmartDFSClient, Smart Client) is a wrapper for the HDFS client that additionally gathers data for the SSM Server.

When an application uses this client to access files in HDFS, SmartDFSClient will report the access event to Smart Server. Once the event is received, Smart Server will update the access count in SSM Metastore for corresponding data.

If the data is compressed or compacted by SSM, the client provides applications with the ability to read it.

SSM Metastore

The **SSM Metastore** is the SSM database that stores such data as access count, rules, and metrics.

Trino architecture

Elena Kostyuchenko

Contents

To Table of Contents

Overview

Concepts

Components

Connectors

Overview

Trino is an open-source SQL query engine used for processing data in parallel, distributed over multiple storages, such as object storages, databases, and file systems.

It was designed for OLAP (Online Analytical Processing) type of workloads and is not efficient for OLTP (Online Transaction Processing).

A single server is enough to run all Trino components. Multiple nodes running Trino, which are configured to collaborate with each other, make up a Trino cluster.

Concepts

Trino is based on the following concepts:

- **Connector**
  A connector is a Trino plugin that creates a connection to a data source using Trino's service provider interface (SPI). A single connector can be used to access one or more data sources of the same type.

- **Catalog**
  A catalog is a Trino abstraction to access a specific data source (e.g. MySQL database). It contains connection settings, credentials, URLs, etc. Every catalog uses a specific connector specified in the `connector.name` property. A catalog contains one or more schemas, which also contain objects such as tables or views. For more information about catalogs, refer to the Catalog management in Trino article.

- **Catalog manager**

  A catalog manager is a mechanism responsible for loading, validating, and managing Trino catalogs. The Trino service provides several catalog management implementations, namely:

  - Static. Reads configuration files and creates catalog entities only on the server startup. Adding a catalog requires a restart of the Trino service.
  - Dynamic. Allows managing catalogs dynamically using the `CREATE CATALOG` and `DROP CATALOG` commands.
  - DBCatalogManager. Allows managing catalogs dynamically and stores catalog metadata in an ADPG/PostgreSQL database. Allows updating catalog configurations without restarting the Trino service.

- **Schema**
  A schema defines the data that can be queried. For a relational database, a schema translates to the same concept in the target database, but for other data sources, a schema might be represented as a different type of table organization.

- **Statement**
  In Trino, a statement is the textual representation of a statement written in SQL. When a statement is executed, Trino creates a query plan consisting of a series of interconnected stages running on Trino workers.

- **Stage**
  In Trino, each execution is comprised of several stages that have a tree hierarchy. When Trino aggregates data from a data source, it creates a root stage to aggregate the output of several other stages, and other stages implement different sections of a distributed query plan.

- **Task**
  A Trino task is a part of a query that is executed on a Trino worker. Each stage of execution may be represented by several tasks.

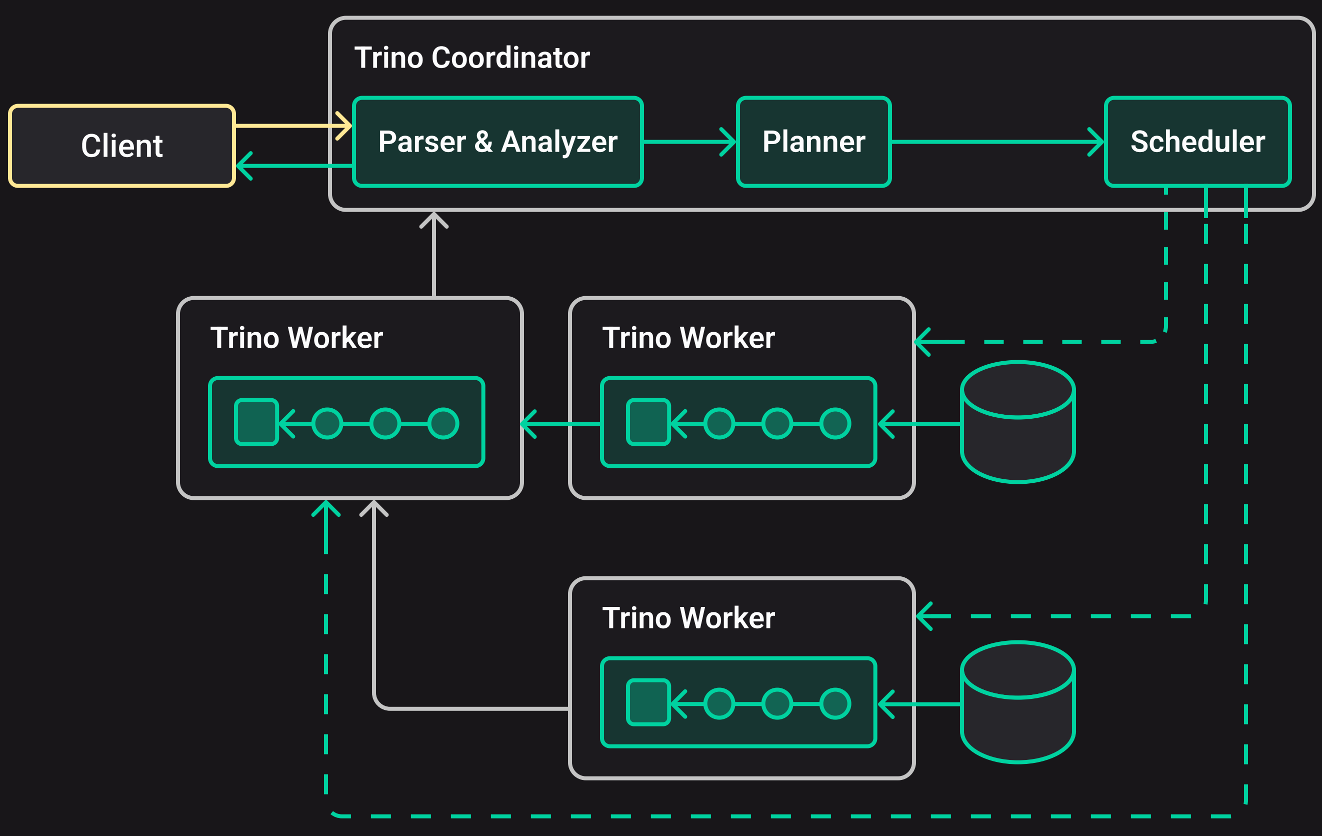
- **Split**
  A split is a smaller part of a target dataset defined in the connector. When Trino is scheduling a query, the coordinator queries a connector for a list of all splits that are available for the table.

Components

Trino architecture is based on the following components:

- **Trino Coordinator**
  A coordinator is a Trino server that handles incoming queries and manages the workers that execute them. The coordinator keeps track of which nodes are running which tasks and what splits are being processed by tasks.

- **Trino Worker**
  A worker is a Trino server responsible for executing tasks and processing data.



Trino architecture

Trino cluster includes one coordinator and multiple worker nodes. All communication and data transfer between Trino components happens via HTTP/HTTPS.

Connectors

Trino contains many built-in connectors for different data sources:

- Data lakes and warehouses: HDFS, Hive, Iceberg, and other connectors.
- Relational databases: MariaDB, MySQL, PostgreSQL, Oracle, ADPG, etc.
- Other systems and utilities: Cassandra, ClickHouse, OpenSearch, Prometheus, Snowflake, JMX, etc.

For more information about connectors, see the Overview of Trino connectors.

YARN architecture

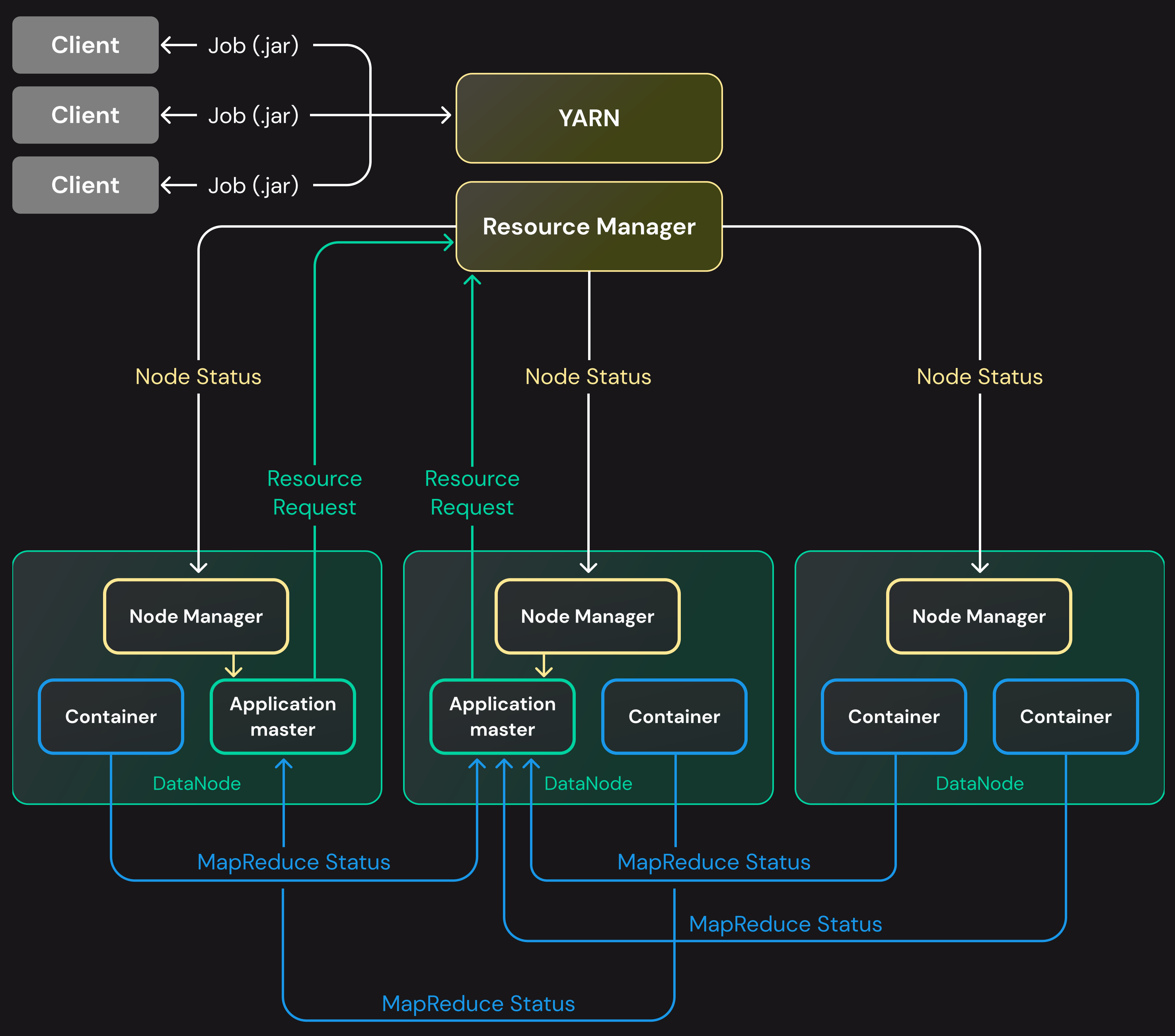
Mikhail Serov

Contents

To Table of Contents

- Resource Manager
  - Scheduler
  - Application Manager
- Node Manager
- Application Master
- Container

YARN is used for scheduling application launches and organizing the cluster resource distribution among various types of data processing mechanisms, such as batch, interactive, and stream processing of data stored in HDFS.



YARN components

YARN includes four main components:

- **Resource Manager** runs as a master daemon and manages resource allocation in the cluster.
- **Node Manager** is installed on every DataNode and is responsible for allocating local resources to application tasks running on those nodes. Node Managers are subordinate to the Resource Manager.
- **Application Master** coordinates an application lifecycle. It interacts with the Resource Manager and Node Managers to run and monitor the application tasks.
- **Container** is a logical bundle of resources including RAM, CPU, network, hard drive, and other countable resources on a single node. It is allocated to an application task.

Resource Manager

This component is the ultimate authority of resource allocation in the cluster. The Resource Manager inspects the cluster resources and makes decisions on allocation of available resources to competing applications. When it receives a request, the Resource Manager splits it into separate resource requests for the respective Node Managers where the actual processing will take place. Resource Manager contains two major components: Scheduler and Application Manager.

Scheduler

This is a part of Resource Manager with the following features:

- The Scheduler is responsible for allocating resources to applications with various restrictions such as resource limits, queues, and so on.
- This is a pure scheduler, which means it does not do any monitoring or application status tracking. If an application fails due to hardware failure or other reasons, the Scheduler does not guarantee failover.
- It performs scheduling based on resource requirements coming from applications.
- This is a plugin object. You can use either Capacity Scheduler or Fair Scheduler plugin in your Resource Manager. By default, the Capacity Scheduler is used.

Application Manager

This is another part of Resource Manager with the following features:

- It is responsible for accepting job submissions.
- It queries the Resource Manager for creating the first container to run the application-specific Application Master.
- It manages the running Application Masters and provides a service for restarting the Application Master container on failure.

Node Manager

Node Manager takes care of individual nodes in the Hyperwave cluster and manages application tasks on the given node. Its primary goal is to manage application containers that it creates by requests from the Resource Manager.

The Node Manger performs the following operations:

- Reports to the Resource Manager about its health and resource capacities of the node.
- Creates and monitors containers and runs processes in them as requested by the Resource Manager and Application Master.
- Controls resource usage by individual containers.
- Provides the log service for container processes.
- Destroys containers as directed by the Application Master or Resource Manager.

Application Master

An application is submitted as a single job in YARN. Each application has a unique Application Master associated with it. Each type of applications has its own Application Master implemented as a JAR file, which is one of resources required by applications. For example, MapReduce applications require the `MRAppMaster` Application Master.

The Application Master coordinates the application processes in the cluster and manages task failures. It negotiates about resources with the Resource Manager and works with Node Managers to create containers, execute application tasks in them, and monitor the task processes. Once started, it sends a heartbeat to the Resource Manager to affirm its health and update the record of its resource demands.

Container

This is a collection of physical resources such as RAM, CPU cores, and disk space allocated to a specific application task on a cluster node. Each YARN container has its own launch context used during the container lifecycle. This record contains an environment variable map, security tokens, a payload for Node Manager services, the command required to create the process in the container, and other parameters. It grants the task rights to use a specific amount of resources (memory, CPU, and others) on a specific node.

Zeppelin

Daria Barysheva

Contents

To Table of Contents

Overview
Architecture
Interpreters
Notes and paragraphs

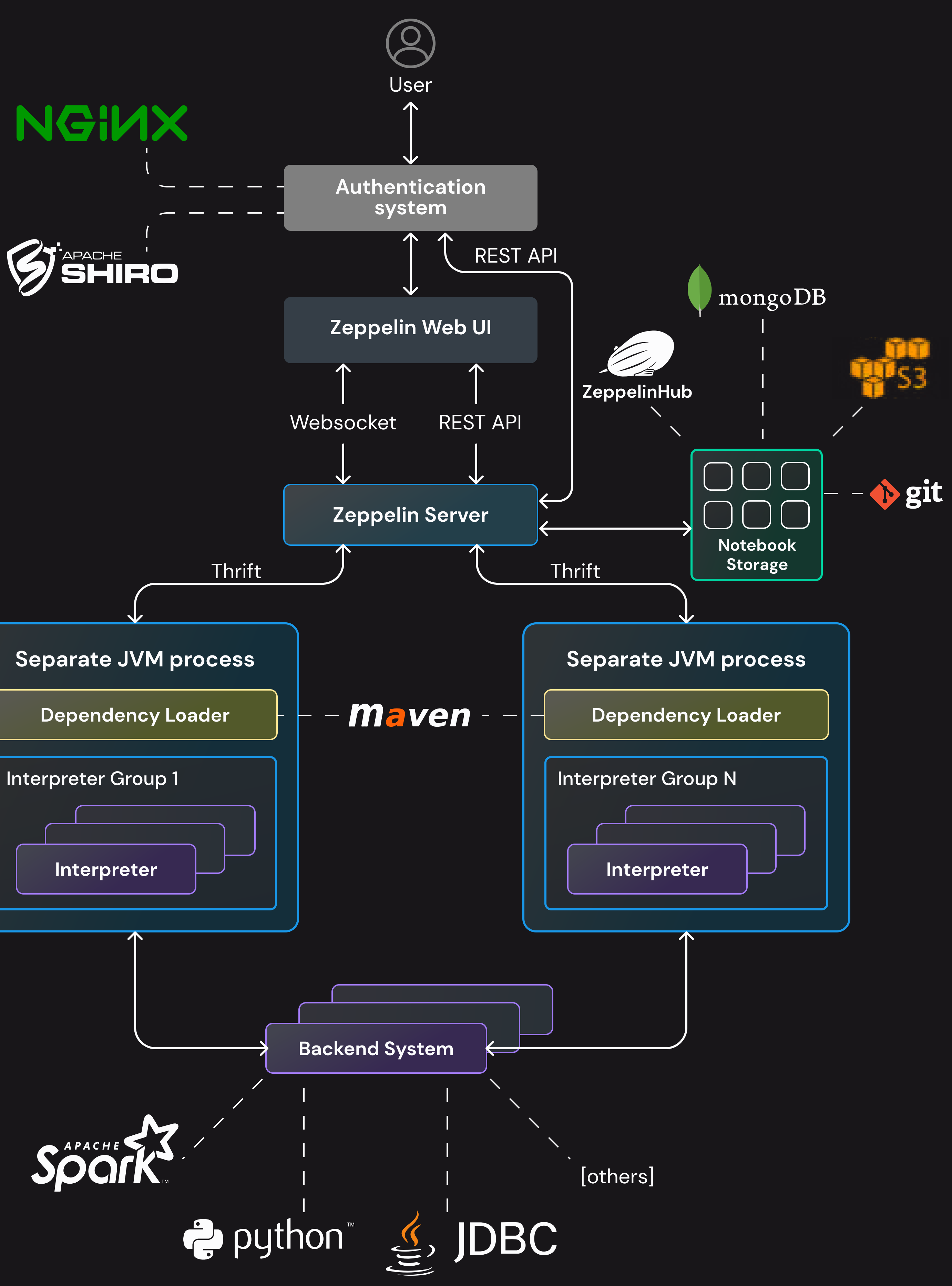
Overview

Zeppelin is an interactive web-based notebook useful for multiple purposes while working with Big Data:

- **Data ingestion/discovery from different sources**. Due to the Interpreters mechanism, in fact, it is possible to plug any language or data processing backend into Zeppelin.
- **Data visualization and analytics**. Zeppelin Web UI supports several built-in tools for visualizing the results of code execution: tables, bar charts, pie charts, area charts, line charts, scatter charts. To add some input forms into notes, you can use dynamic forms. Besides that, Zeppelin allows to add html into any output. All these features help data scientists to get the comprehensive data for analytics.
- **Data collaboration**. A Zeppelin Note URL can be shared among different collaborators. Any changes are broadcasted in the real time.

Architecture

The high-level architecture view of Zeppelin is shown below.


User
NGINX
APACHE SHIRO
Authentication system
REST API
Zeppelin Web UI
Websocket
REST API
Zeppelin Server
mongoDB
ZeppelinHub
S3
Notebook Storage
git
Thrift
Thrift
Separate JVM process
Dependency Loader
Interpreter Group 1
Interpreter
maven
Separate JVM process
Dependency Loader
Interpreter Group N
Interpreter
Backend System
APACHE Spark
python
JDBC
[others]

Zeppelin architecture

Let's discuss the main points of the scheme:

- Users connect to Zeppelin via one of the configured authentication systems. For example, Nginx, Apache Shiro, etc.
- Authorized users can interact either with the Zeppelin Web UI or directly with the Zeppelin server — via the available REST API.
- Notes created by users can be saved in different storages. Among them are Git, S3, MongoDB, ZeppelinHub, etc.
- The main functionality in Zeppelin is performed by interpreters that are responsible for using different programming languages and data processing backends: Spark, Python, JDBC, SQL, etc. Interpreters communicate with Zeppelin Server via Apache Thrift.
- By default, Zeppelin uses the central Maven repository for resolving dependencies and loading libraries that are required for interpreters. You can also load libraries from the local file system and add external repositories. Read more about dependency management in the Zeppelin documentation.

Interpreters

A Zeppelin **Interpreter** is a plug-in that enables to use a specific language or data processing backend platform for querying and analyzing data. For example, the interpreter `%sh` is designed for running shell commands, the interpreter `%jdbc` — for jdbc calls, etc.

Every interpreter belongs to some group. Interpreters from the same group are running in the same JVM process. So a group is a unit of starting and stopping interpreters. Every group can contain several interpreters, but one interpreter belongs only to one group. For example, the Spark interpreter group includes such interpreters as `%spark`, `%sql`, `%pyspark`, `%ipyspark`, `%r`, and the dependency loader `%dep`. Interpreters in the same group can reference each other.

The list of available community managed interpreters can be found in the Zeppelin documentation. In fact, these are interpreter groups, some of them contain one interpreter, others — several. You can always add a new interpreter within one of the existing groups. Besides that, there are some 3rd party interpreters that can be loaded to Zeppelin too. At least, it is possible to create your own interpreter that is not based on existing groups.

The table below shows the main actions available for interpreters.

Actions with interpreters

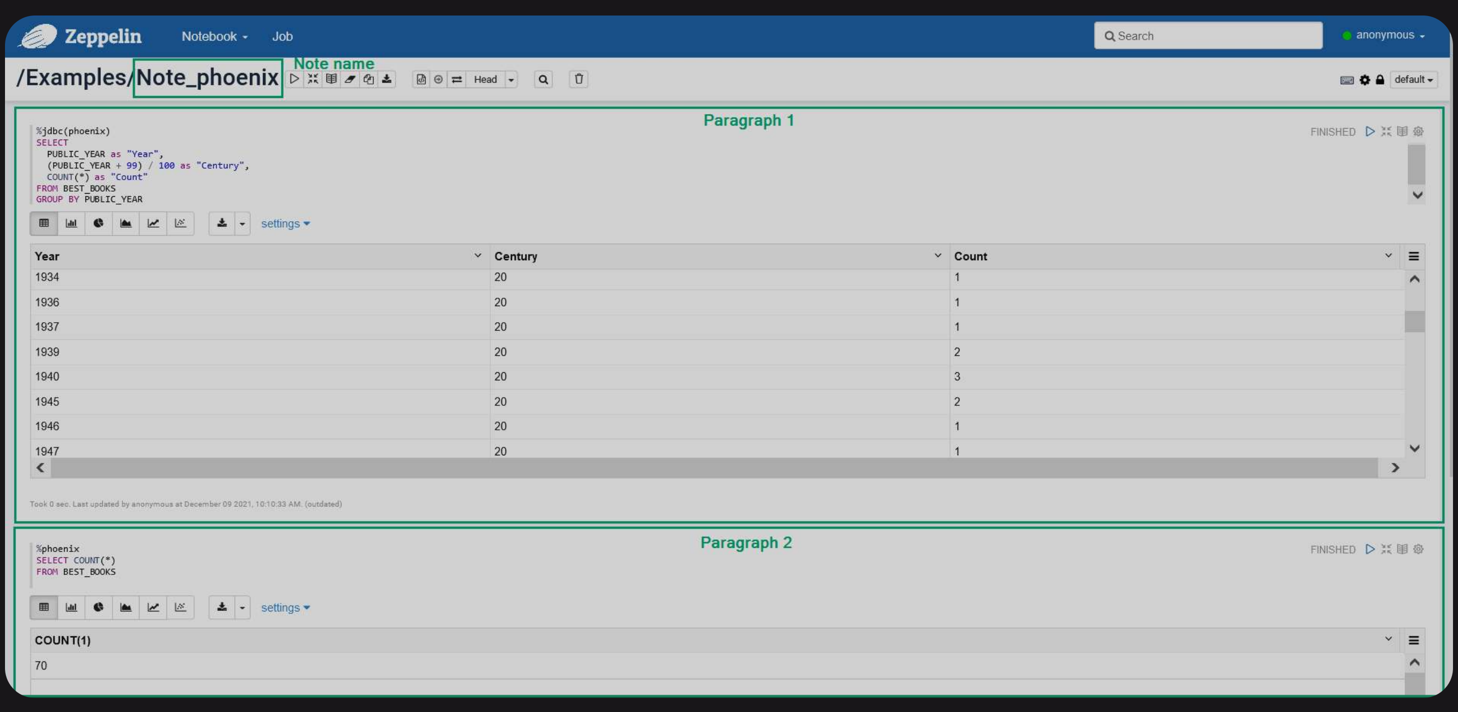
Action	Example/Documentation
Adding a new interpreter into the existing group	Example
Writing an interpreter	Documentation
Installing a 3rd party interpreter	Example

Notes and paragraphs

A Zeppelin **Note** is a collection of code snippets written in the languages that are supported by the interpreters bound to this note. Every note has a URL that can be shared with other users.

One note can include several **Paragraphs**. These paragraphs can be both requests to the different backend systems and explanatory text labels built with html. Every paragraph includes the code and the result. Any of these sections can be hidden.

Zeppelin Web UI supports several viewing modes of notes including the complete hiding of the source code.


Zeppelin
Notebook
Job
Search
anonymous
/Examples/Note_phoenix
Note name
Head
default
Paragraph 1
FINISHED
settings
Paragraph 2
FINISHED
settings

```
%jdbc(phoenix)
SELECT
  PUBLIC_YEAR as "Year",
  (PUBLIC_YEAR + 99) / 100 as "Century",
  COUNT(*) as "Count"
FROM BEST_BOOKS
GROUP BY PUBLIC_YEAR
```

Year	Century	Count
1934	20	1
1936	20	1
1937	20	1
1939	20	2
1940	20	3
1945	20	2
1946	20	1
1947	20	1

Took 0 sec. Last updated by anonymous at December 09 2021, 10:10:33 AM. (outdated)

```
%phoenix
SELECT COUNT(*)
FROM BEST_BOOKS
```

COUNT(1)
70

Notes structure

NOTE

For more information about notes and paragraphs, see the Zeppelin documentation.

ZooKeeper

Daria Barysheva

Contents

To Table of Contents

- Architecture
- Data model
  - Znodes
  - Watches
- How it works
  - Sessions
  - Read and write operations

ZooKeeper is a centralized coordinating service performing the following functions in Hyperwave clusters:

- Maintaining configuration information.
- Providing a naming registry.
- Ensuring distributed synchronization.
- Avoiding such errors, as race conditions and deadlocks.
- Providing group services.

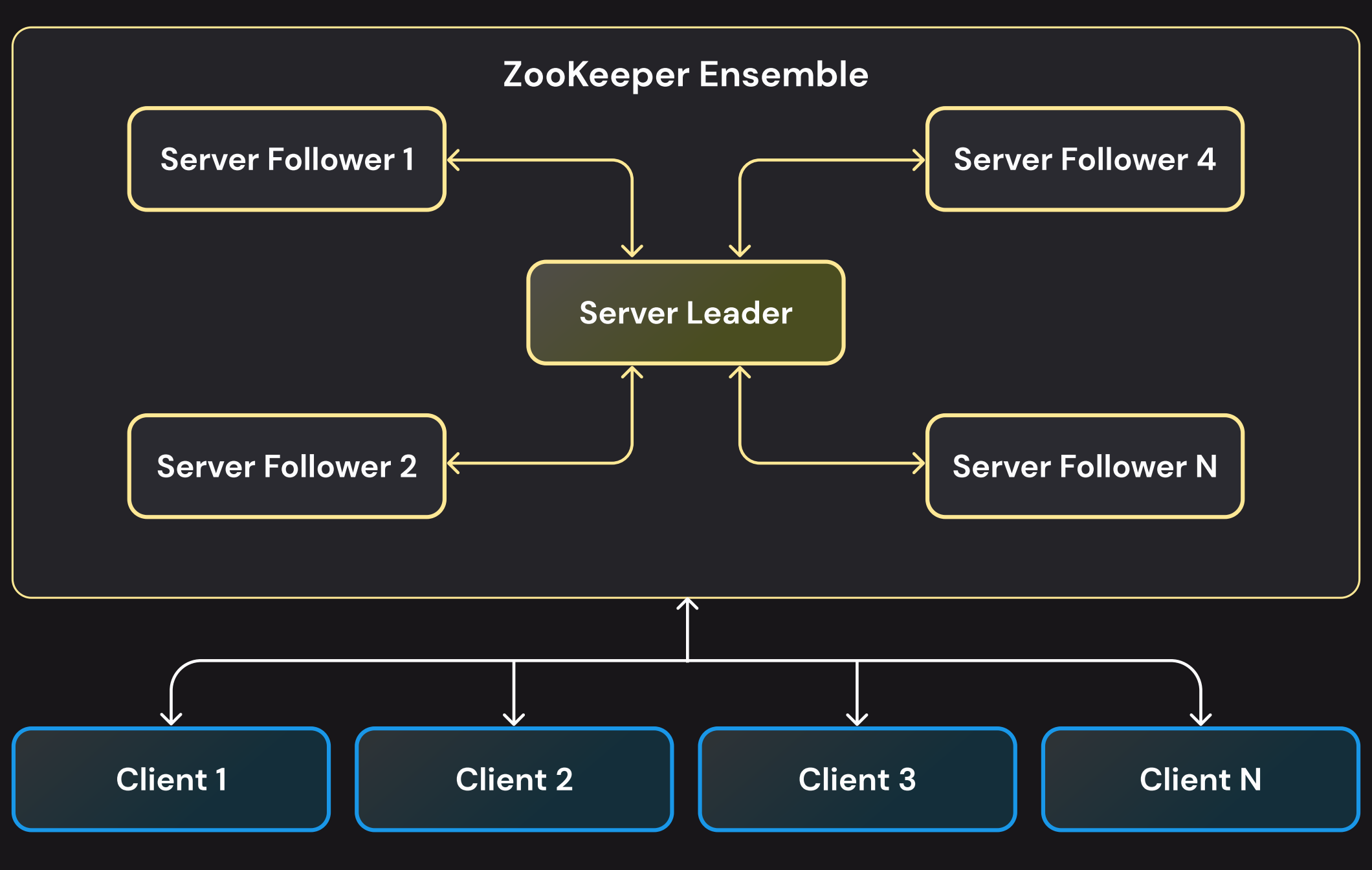
It is a Java-based service and has bindings for Java and C. Its simple architecture and personalized API helps to prevent possible errors when managing large distributed systems.

Architecture

Zookeeper uses the cient-server architecture and contains the following components:

- **Ensemble** is the collection of all ZooKeeper servers. The ensemble requires at least three nodes for its reliability. When one server goes down, the other two make up a **quorum** that keeps working. It is recommended to use an odd number of servers in the cluster: 3, 5, 7, and so on.
- **Server** is one of the nodes in the Zookeeper ensemble. Its duty is to provide all kinds of services to clients. There are two possible types of servers:
  - **Server Leader** performs automatic recovery in case other servers fail. The leader is elected at the ZooKeeper startup.
  - **Server Follower** is each server in the ensemble other than the leader. It follows the instructions sent by the Leader.
- **Client** is a consumer of ZooKeeper services.

ZooKeeper components are displayed in the following diagram.


ZooKeeper Ensemble
Server Follower 1
Server Follower 4
Server Leader
Server Follower 2
Server Follower N
Client 1
Client 2
Client 3
Client N

ZooKeeper components

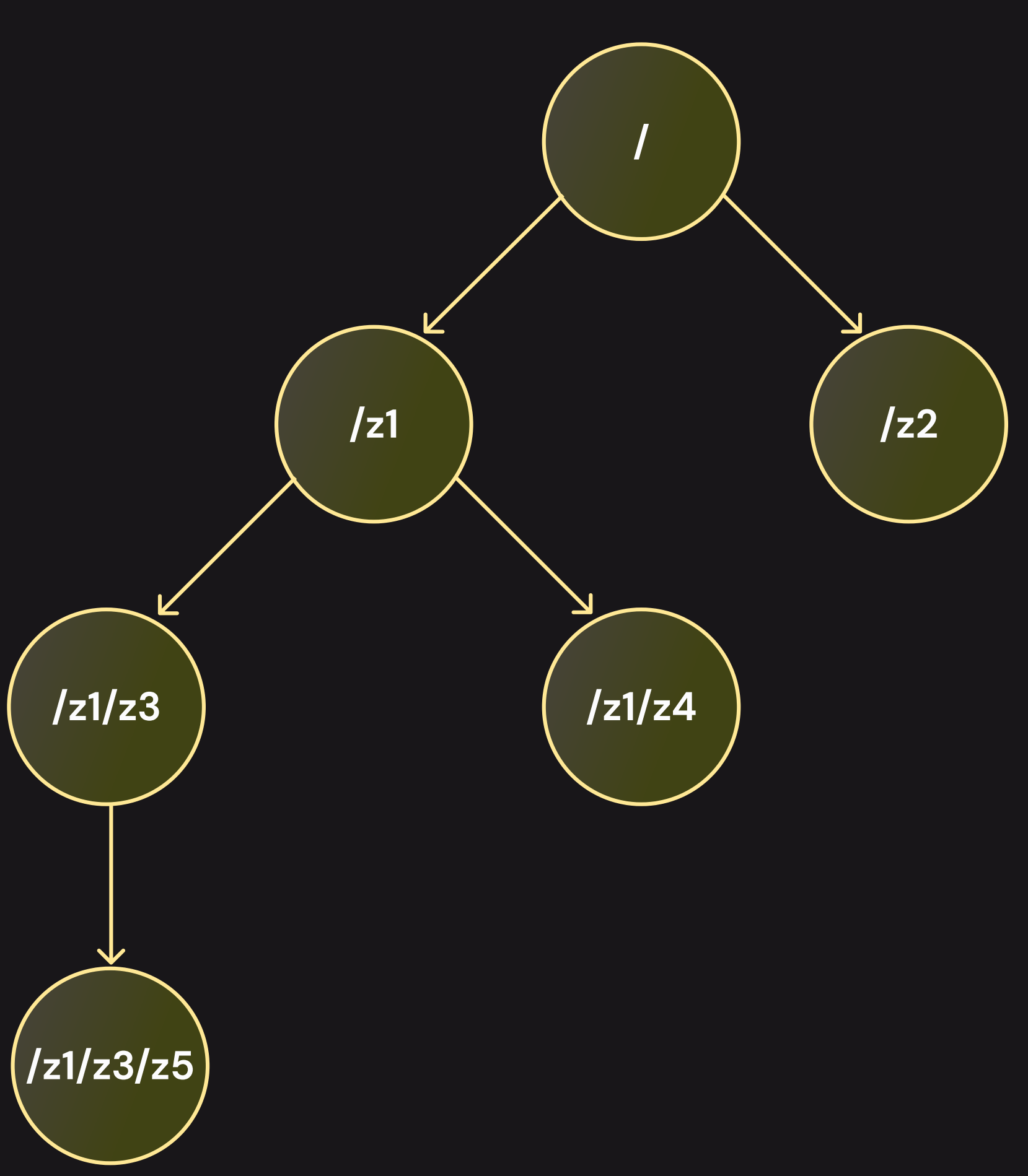
Data model

Znodes

ZooKeeper allows distributed processes to communicate with each other through a shared hierarchal namespace, which is organized similar to a standard file system. Every node in this namespace is called **znode** and identified by its path. Unlike a typical file system, znodes can perform the functions of both a file and a directory simultaneously. All data is kept in memory, which allows ZooKeeper to provide high throughput and low latency.

> **CAUTION**
>
> Since ZooKeeper was designed to keep coordination data (configurations, statuses, location information, and so on), it expects that the stored data will not be too large. Znode size cannot exceed 1 MB.

The following diagram presents an example structure containing one root znode identified by / and two children: */z1* and */z2*. These znodes, in turn, can have their own children, and those — their own, and so on.


/
/z1
/z2
/z1/z3
/z1/z4
/z1/z3/z5

ZooKeeper data model

In each znode, data is always read and written atomically:

- A read request gets all the data bytes associated with the specified znode.
- A write request replaces all data in the znode.

Each znode has metadata in a structure called stat. It includes version numbers, timestamps, related transaction IDs, and other metadata. Znodes are protected by ACLs (Access Control Lists) containing a list of users with privileges assigned to them.

The possible types of znodes are described in the following table.

Types of znodes

Type	Description
Persistent	The default type of znodes. Such a znode stays alive even after the client created it is disconnected
Ephemeral	Less durable type. Such a znode stays alive while the client that created it is connected to ZooKeeper. It cannot have children
Sequential	When a znode is marked as sequential during its creation, ZooKeeper appends a 10-digit sequence number to the original name, for example, */top/myspace/mynode0000000001*. Sequential znodes play an important role in locking and synchronization processes. A sequential znode must be persistent or ephemeral, which means we have not one sequential type but two: **persistent-sequential** and **ephemeral-sequential**
Container	The type created to simplify cleaning of unused nodes (parent nodes left without child nodes) by ZooKeeper. A container node is similar to a persistent node, but with an additional function of being put into candidates for deletion by the ZooKeeper server as soon as its last child znode is deleted

Watches

Watches is a special mechanism that enables clients to receive notifications about changes in the specified znodes. A watch will be triggered and removed when the monitored znode is changed.

Watches can be set for those operations that read the state of ZooKeeper: checking a znode existence, reading data from a znode, and retrieving a list of znode children.

> **CAUTION**
>
> Watches are based on one-time triggers. After a watch notification is triggered, no further watch notifications will be sent unless the client configures a new watch.

How it works

Sessions

ZooKeeper uses the session mechanism. A **session** is a period of time wherein a user interacts with a server to receive the requested service. Requests in each session are executed in the FIFO order. When a client connects to a server, a session is established and the client receives the session ID.

A client connects to exactly one server to submit its requests. It can be either the leader or a follower. If the client does not receive confirmation of the successful connection, it tries to connect to another server in the ZooKeeper ensemble.

After the connection is established, the client sends a heartbeat to the server to keep the session alive. If the ZooKeeper ensemble does not receive a heartbeat within the session timeout specified when the service was started, it assumes the client is unavailable.

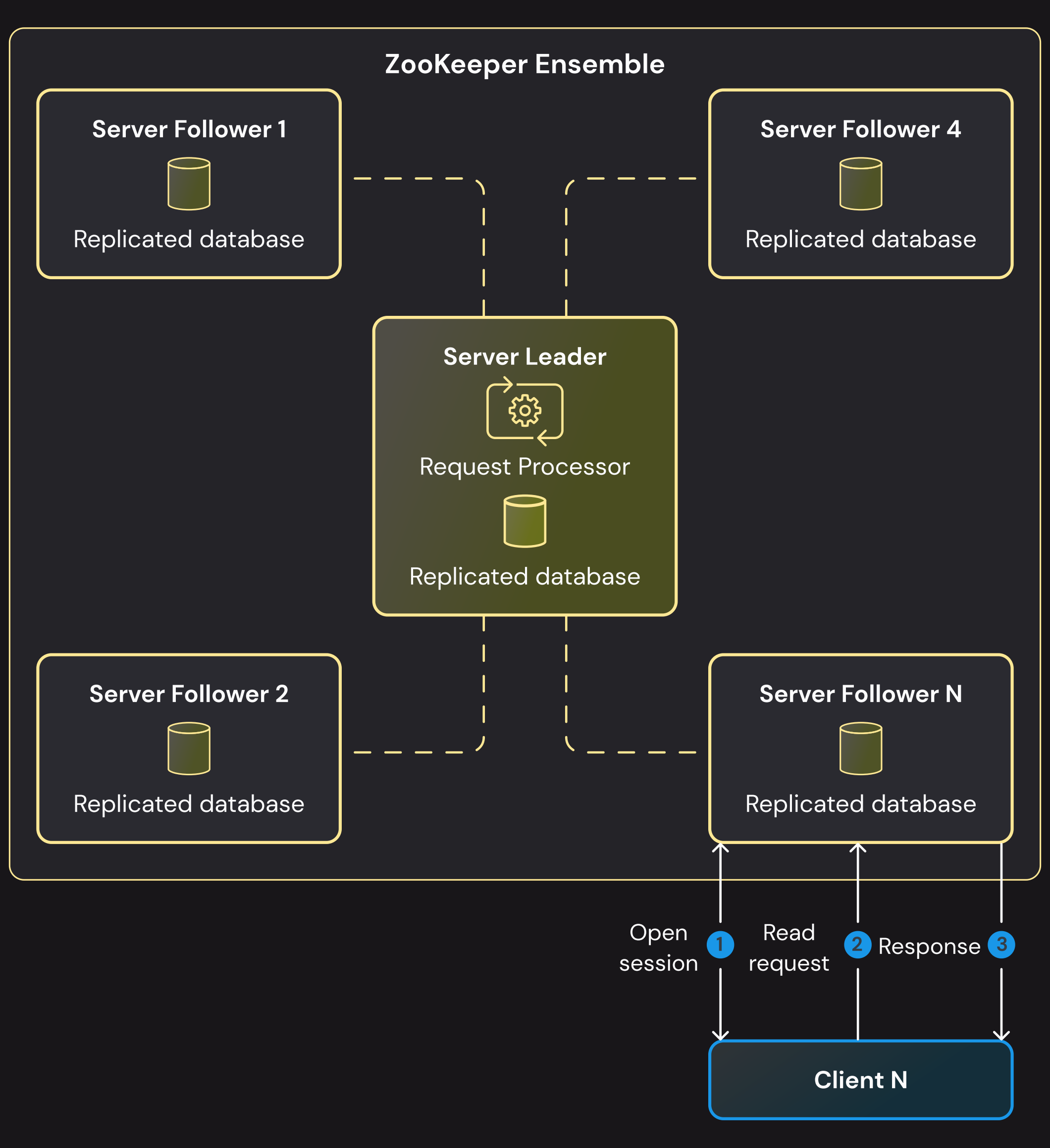
Read and write operations

Each server in the ZooKeeper ensemble has a local replica of the ZooKeeper database. It is an in-memory database containing the entire data tree. Updates to the znode tree are logged to the disk storage so that they can be restored.

With the replicated database, all read requests are processed directly by the servers without involving the leader. Requests to change data are processed by the leader using a special agreement protocol.

The typical read operation includes the following steps:

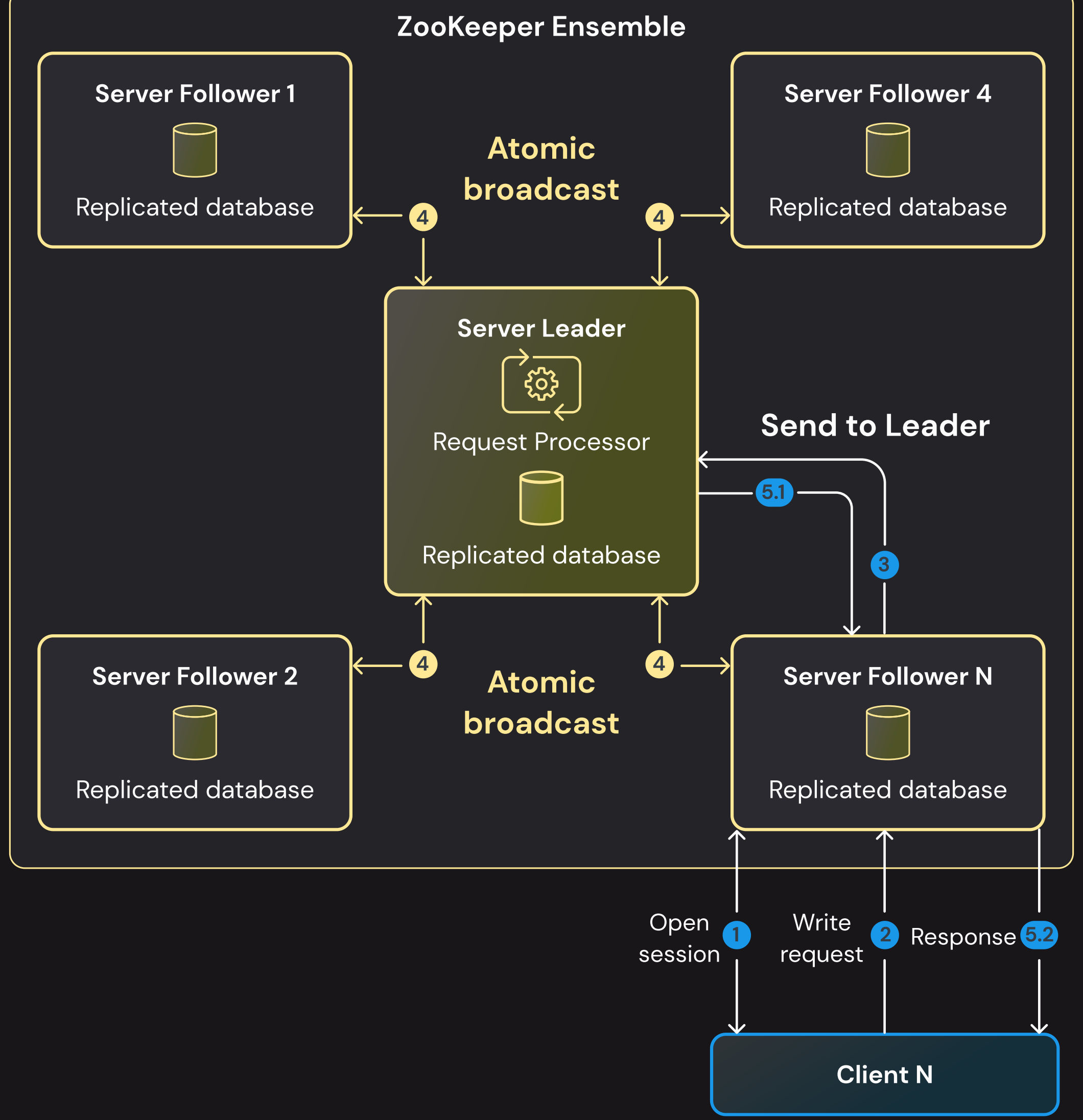
1. The client starts a session with one of the ensemble servers.
2. The client sends a read request to the server. This request includes the path to the znode containing the required data.
3. The server provides the client with the required information stored in the database replica on that server. In this operation, communication with the leader is not required.


ZooKeeper Ensemble
Server Follower 1
Replicated database
Server Follower 4
Replicated database
Server Leader
Request Processor
Replicated database
Server Follower 2
Replicated database
Server Follower N
Replicated database
Open session
1
Read request
2
Response
3
Client N

Typical read sequence

The typical write sequence looks as follows:

1. The client starts a session with one of the ensemble servers.
2. The client sends a write request to the server. The request includes the path to the znode where the data is to be changed.
3. The server sends the write request to the ensemble leader.
4. The ensemble leader forwards the write command to all the followers using the atomic broadcast. The write request is only processed if the most part of the followers (called quorum) returns a positive response. In this case, the requested changes are committed to all database replicas.
5. The response is returned to the client.


ZooKeeper Ensemble
Server Follower 1
Replicated database
Atomic broadcast
Server Follower 4
Replicated database
4
4
Server Leader
Request Processor
Replicated database
Send to Leader
5.1
3
4
4
Server Follower 2
Replicated database
Atomic broadcast
Server Follower N
Replicated database
Open session
1
Write request
2
Response
5.2
Client N

Typical write sequence

> **NOTE**
>
> - For more information on HBase and ZooKeeper interaction, refer to HBase architecture.
> - Examples of basic commands that can be executed via ZooKeeper CLI are given in the article Work with Znodes.
> - For more details on ZooKeeper concepts, see ZooKeeper documentation:
>   - Overview
>   - Programmer’s Guide
>   - ZooKeeper Internals

Kerberos

Mikhail Serov

Contents

To Table of Contents

Kerberos overview
Authentication process
  Example
KDC types

Kerberos overview

The strict authentication and verification of user identity is necessary for the secure access in Hyperwave. All users should be unambiguously identified within the whole Hyperwave cluster. The users who were successfully identified gain access to the resources (for example, files or directories) and can interact with the cluster (for example, perform MapReduce tasks).

If a user was granted access to the cluster under some name, then all cluster components also trust that name. This process is facilitated by utilizing the Kerberos database that contains cluster access credentials for all users. Hyperwave cluster resources (such as hosts or services) must also undergo mutual authentication. This is necessary to prevent various malicious systems pretending as cluster components from accessing the cluster.

Kerberos concepts

Concept	Description
Key Distribution Center, KDC	The reliable source for authentication in a system that supports Kerberos
Kerberos KDC Server	Machine or server that serves as KDC
Kerberos Client	Any machine of the cluster that authenticates via KDC
Kerberos Principal	Unique name of a user or service that authenticates via KDC. This name consists of the basic user name and the full host domain name the server works at
Key	Unique key that is assigned to every principal. This key is sent to the authentication system
Keytab	File that contains one or several principals along with their keys. This file is stored in the Kerberos database where it's being accessed by server principal. The file is then stored in the protected directory at the server component node
Realm	Kerberos network that includes KDC and one or several clients
KDC Admin Account	Administrator account that is used by ADCM for creating principals and generating KDC keys
Ticket Granting Server (TGS)	Server that grants tickets and permissions
Authentication Server (AS)	Server that performs the initial authentication check and issues a Ticket Granting Ticket (TGT)
Ticket Granting Ticket	Ticket for granting a ticket, also referred to as TGT. TGT includes the following objects: second copy of the session key, user name, end of the ticket life period. The ticket is encrypted with the use of KDC own master key that is known only to KDC. This means that TGT can be decrypted only by KDC itself

Authentication process

Hyperwave utilizes Kerberos for the strict authentication and to verify identities for users and services. The Kerberos server is called **Key Distribution Center** (KDC). It consists of the three following components:

- **Users and services MIT KDC database** — includes principals known to the server and the corresponding Kerberos passwords.
- **Authentication Server (AS)** — performs the initial identity checking task and issues the Ticket Granting Ticket (TGT).
- **Ticket Granting Server (TGS)** — issues a Kerberos token based on the initial TGT.

A simplified Kerberos flow is presented below:

1. The principal requests authentication from AS.
2. The AS replies by sending a TGT that is encrypted with the use of the Kerberos principal password. This password is known only to the principal and AS.
3. The principal decrypts the received TGT locally via the Kerberos password. From this moment on, every principal can use TGT for obtaining the tickets from TGS. Those tickets allow the principal to access various services.

Due to cluster resources (services or hosts) not being able to provide a password for the decryption of TGT every time, a *keytab* file is used. This file contains credentials for resource authentication. The set of hosts, users, and services controlled by the Kerberos server is called the **realm**.

Example

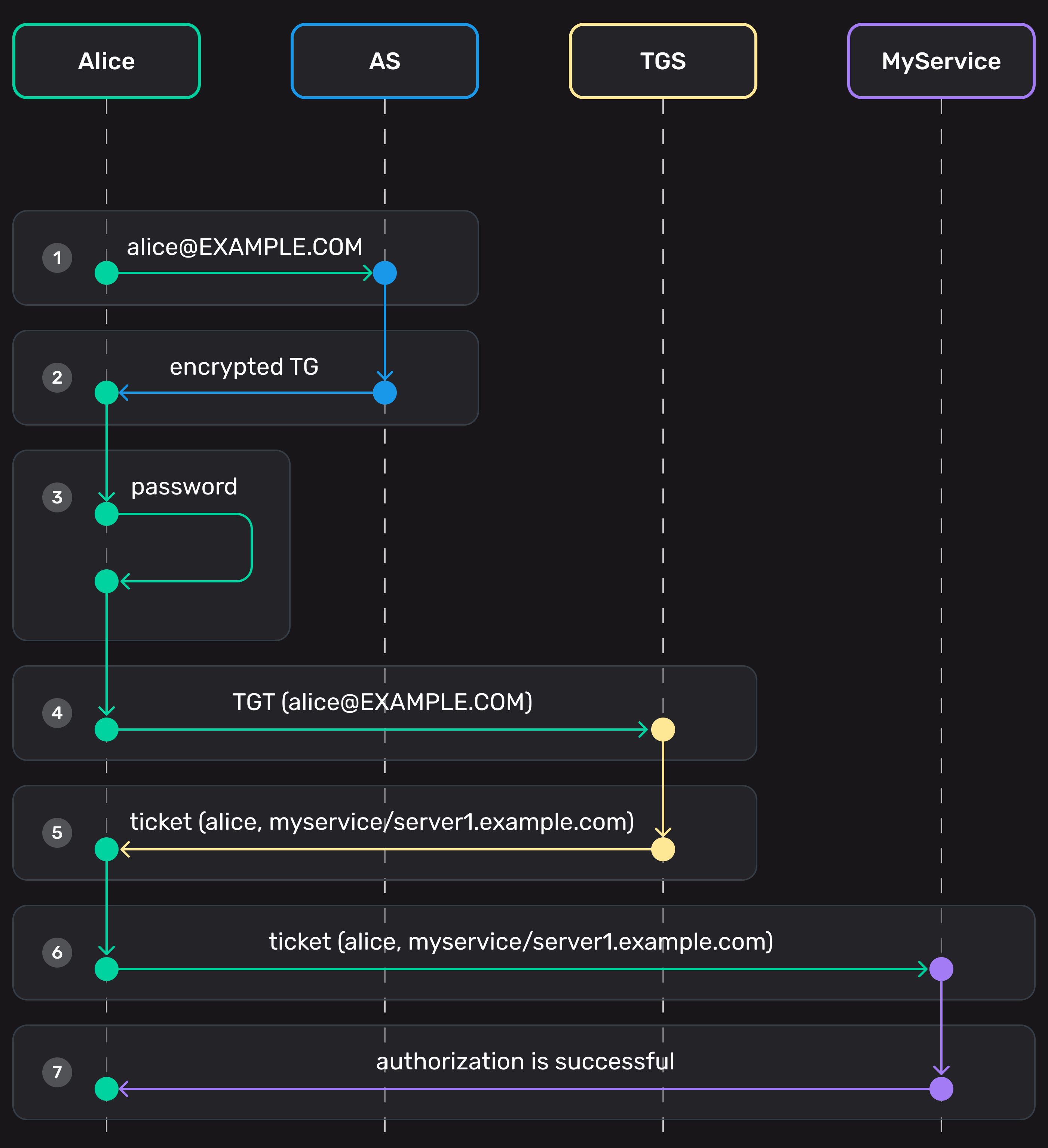
Below is a simple example of the authentication process via Kerberos. This example includes the following components:

- *EXAMPLE.COM* — a Kerberos realm.
- *Alice* — a user that is assigned the *alice@EXAMPLE.COM* user principal name (UPN).
- *myservice* — a service located at the *server1.example.com* node and is assigned the *myservice/server1.example.com@EXAMPLE.COM* service principal name (SPN).
- *kdc.example.com* — a key distribution center (KDC) for the *EXAMPLE.COM* realm.

In order to get access to the *myservice* service, Alice must provide a valid ticket for the *myservice*. This is implemented via the following steps (some details are omitted for brevity):

1. Alice must obtain a ticket for granting a ticket (TGT). In order to do this, Alice initiates a request to the Authentication Server (AS) located at *kdc.example.com* and identifies herself as *alice@EXAMPLE.COM*.
2. The AS responds to Alice and grants her a ticket for granting a ticket (TGT) encrypted with the use of the *alice@EXAMPLE.COM* principal password.
3. After receiving the encrypted message Alice enters the *alice@EXAMPLE.COM* principal password to decrypt the message.
4. After the message with TGT is successfully decrypted, Alice requests a service ticket to access the *myservice/server1.example.com@EXAMPLE.COM* from the ticket granting service (TGS) located at *kdc.example.com*. The request from Alice contains the TGT and the response contains the service ticket.
5. Ticket granting service (TGS) checks the received TGT and grants Alice the service ticket encrypted with the use of the *myservice/server1.example.com@EXAMPLE.COM* principal key.
6. Alice sends the service ticket to the *myservice* service. The *myservice* service is able to decrypt the service ticket with the use of the *myservice/server1.example.com@EXAMPLE.COM* principal key. The *myservice* service is also able to check the decrypted service ticket.
7. Alice gets permissions to use *myservice* after successful authentication.

The above process is shown in the image below.


Alice
AS
TGS
MyService
1
alice@EXAMPLE.COM
2
encrypted TG
3
password
4
TGT (alice@EXAMPLE.COM)
5
ticket (alice, myservice/server1.example.com)
6
ticket (alice, myservice/server1.example.com)
7
authorization is successful

Kerberos authentication flow

KDC types

Cluster kerberization is possible via the following KDC types:

- MIT Kerberos that consists of the principal database and Kerberos keys storage.
- MS Active Directory that consists of the principal database and Windows Server keys storage.
- FreeIPA which is a free open source identity management system for Linux/UNIX environments.
- Samba — software that can be used as an Active Directory domain controller.

Using either type, the identification and authentication process is practically the same. The only difference is the KDC type that is being used.

ADH releases

Sergei Tikhomirov, Konstantin Alpashkin, Daria Barysheva

Contents

To Table of Contents

- 4.1.0
- 4.0.0
- 3.3.6.2
- 3.3.6.1
- 3.2.4.3
- 3.2.4.2
- 3.2.4.1
- 3.1.2.1
- 2.1.10
- 2.1.8
- 2.1.7
- 2.1.6
- 2.1.4
- 2.1.3
- 2.1.2
- 2.1.1
- 2.1.0

4.1.0

4.1.0.b1

Date: 30.10.2025

New services

Added the Flink2 2.0.0_arenadata1 service with the following patches:

- Fixed OrcShim to support hive 4.
- Added flink-connector-hive 3.0.0_arenadata1 with support for Hive4.

Added the HAproxy component for Trino

Added the Trino Client component for Trino

Added the Ozone Client component for Ozone

Added the Spark 4.0.0_arenadata1 service with the following patches:

Spark 4 patches

- ADH-4519: Fixed scalastyle
- ADH-4519: Arenadata patches
- ADH-4519: Updated `io.grpc.version`, set `hadoop.version` and `zookeeper.version`
- ADH-4519: Added a configurable artifact directory path for Spark Connect
- ADH-4803: Added a blacklist filter to exclude certain properties
- ADH-6342: Skip rewriting stats for checkpointed LogicalRDD

- ADH-5569: Force a purge during table/partition drop in case the `external.table.purge` table option is present
- ADH-6416: Wildcard replace `*` → `%`
- ADH-4519: Use Hive 2.3.10_arenadata1
- ADH-4519: Use purgeable external tables for path updates and partition handling when renaming tables

Upgrades

Upgraded Trino to 476_arenadata1

Trino updates

- ADPS-1168: Support logic of working with trino-hive-ranger-chained-plugin
- ADH-6568: Remove dependency on Joda-Time in the ADB connector module
- ADH-6569: Fix serialization of Date type objects
- ADH-5781: Fixed *trino-cli-startup.sh*
- ADH-6309: Resolve `_HOST` in Kerberos principal before keytab login
- ADH-6284: Added `JAVA_HOME` support
- ADH-6306: Kerberos/UGI for Ozone/HDFS exchange + auto TGT refresh
- ADH-6568: Fixed Trino-ADB Connector tests
- ADH-6284: Build Trino on Java 23
- ADH-6006: Implemented the cleanup job launch audit

Implemented OpenAPI Connector 1.80_arenadata1

Upgraded Ozone to 2.0.0_arenadata1

Ozone updates

- HDDS-13067: Container Balancer delete commands were sent with an expiration time in the past
- HDDS-13366: Bucket ACLs were being overwritten when viewing other ACLs
- HDDS-12816: Ozone debug replicas chunk-info had incorrect or None in FileLocations
- HDDS-13234: Expired secret key could abort leader OM startup
- HDDS-12233: Atomically import a container
- HDDS-13023: Container checksum was missing after container import
- ADH-5987: Set `ranger.version` to `2.5.0` and `rocksdb.version` to `7.10.2`
- ADH-5988: Added new JARs to *hadoop-ozone/dist/src/main/license/jar-report.txt*
- HDDS-12974: Docker could not parse extra host IP
- HDDS-12843: ntermittent NPE in TestDecommissionAndMaintenance

Upgraded HUE to 4.11.0_arenadata4

HUE updates

- ADH-6971: Use environment-based Java discovery in JDBC connector
- ADH-5927: Flink-SQL interpreter support
- ADH-6768: Unable to open Notebook page in HUE UI
- ADH-6490: Display content file in HUE UI in HDFS/Ozone interpreters didn’t work

Upgraded Kyuubi to 1.10.1_arenadata2

Kyuubi updates

- ADH-6776: Shade jersey packaged

- KYUUBI-7055: Support to filter out server only configs with prefixes
- KYUUBI-7051: Fix usage without sslTrustStore of JDBC driver

---

Upgraded Spark to 3.5.4_arenadata2

Spark updates

- ADH-6415: Skip rewriting stats for checkpointed LogicalRDD
- ADH-6416: Wildcard replace `*` → `%`

Upgraded ADB Spark connector to 1.1.0-3.5.4_arenadata2

ADB Spark connector updates

- Improved the server address resolution with fallback.
- Added auto search for an HTTP server bind address by NIC.

Upgraded Hive to 4.0.1_arenadata2

Hive updates

- ADH-6473: Bumped the HBase service version from 2.5.10 to 2.6.3
- ADH-6474: Bumped DataNucleus version from 5.2.8 to 5.2.10
- HIVE-28749: The default `hikaricp.leakDetectionThreshold` was not valid
- HIVE-28545 Removed a global lock on `HMSHandler.getMSForConf` which is prone to deadlock
- HIVE-28900: Make the objectstore-secondary connection pool size configurable
- HIVE-28839: Connection starvation in HMS if DataNucleus value generation fails
- HIVE-28788: ObjectStore failed to commit the transaction could lead to connection starvation

---

Upgraded HBase to 2.6.3_arenadata1

HBase updates

- ADH-6295: Upgraded hadoop.guava to 32.0.1-jre

Upgraded Phoenix to 5.2.2_arenadata1

Phoenix updates

- ADH-6575: Added HexQualifierFunction UDF to convert VARBINARY to HEX
- ADH-6575: Downgraded `hadoop-three.version` to 3.3.6

Upgraded Phoenix Query Server to 6.0.0_arenadata4

Phoenix Query Server updates

- ADH-6588: Added default config
- ADH-6588: Bumped the HBase service version from 2.5.10 to 2.6.3

Upgraded SSM to 2.1.1

SSM updates

- ADH-6574: Added filtering for the `-ruleId` arg from the arguments of DistCp call
- ADH-6442: Added support for free port picking on SSM server level
- ADH-6281: Fixed cmdlet stopping logic
- ADH-6189: Fixed sorting field for file_access.access_time column
- ADH-5916: Added new Rules API integration tests

- ADH-6152: Added support for setting the number of cmdlet executors for specific node
- ADH-6071: Made search of several columns case-insensitive

Upgraded Solr to 8.11.4_arenadata1

Solr updates

- ADH-6698: Detect `JAVA_HOME` via bigtop and use proper JVM path in Solr scripts

New features

Added support for Astra Linux 1.7 "Voronezh"

Added support for AltLinux 10 SP

Implemented a utility for explicit selection of Java for all services

Preconfigured the HUE PySpark interpreter

Preconfigured the HUE Flink 2 interpreter

Added the dashboards to monitoring for the following services:

- Trino
- Zeppelin
- HBase
- ZooKeeper
- Solr

Implemented the Observer NameNode in a HA HDFS cluster

Implemented the HA mode for Trino Coordinator

Added the ability to prepare hosts (install packages on them) after adding them to a cluster

Implemented the ability to store passwords and credentials in JCEKS for the Ozone service

The Trino service now supports HadoopSQL policies when integrated with Ranger

Implemented OverwriteOnCommitProtocol for Spark3

Improvements

Improved configuration capabilities of the Trino service:

- Added the `--add-opens` option to the ***jvm.config*** parameter for each component.
- Added management capabilities for the following authentication types:
  - Password file
  - Header
  - Certificate
  - JWT
  - OAuth 2.0
- Added control over the following access settings:
  - File-based
  - OPA Policy

Implemented the ability to configure a proxy server to connect to a repository in the cluster settings

Added the ability to set custom attributes for the ***Manage Kerberos*** action

Improved the Kyuubi service's logging template

The `kyuubi-beeline` client is now available in global path

Added the ability to auto-cleanup Trino catalogs when using DBCatalogManager

Added automatic creation of user directories from configuration for the following services:

- Ozone
- Impala
- Zeppelin
- Spark
- Hive
- HBase
- Solr

Added the ***Metastore DB schema init/upgrade*** checkbox in a modal window for the ***Restart*** action of the Kyuubi service

The HDFS service's ***Check*** action now takes into account the presence of an Ozone service in the cluster with the Core configuration's ***defaultFS*** parameter configured on Ozone

The Hive service's ***ACID Transactions*** property has been replaced with an activated group where you can set additional properties required for more efficient operation

Improved the behavior of the ***Add/Remove components*** action. Now the cluster topology is committed to ADCM immediately after the main job is completed even if some errors occur

The ZooKeeper service actions are now available on the ***Host*** → ***Host-Components*** tab

Improved the default settings of the Ranger Spark plugin to meet security requirements:

Changes in *ranger-spark-audit.xml*:

- `xasecure.audit.destination.solr.batch.filespool.perms` = `666`
- `xasecure.audit.destination.solr.batch.filespool.dir.perms` = `777`

Changes in *ranger-spark-security.xml*:

- `ranger.plugin.spark.policy.cache.file.perms` = `666`
- `ranger.plugin.spark.policy.cache.dir.perms` = `777`

The `hadoop.proxyuser.*` settings are now editable

Updated HUE dashboard in Monitoring, added the ability to filter by host

Integration with Iceberg is now available in the CE edition

Bug fixes

ADB Spark3 connector: `DataFrame.count()` failed if a partition count was set

Ozone: fixed parameters in ***ozone-httpfs.xml*** for a kerberized cluster

Wrong JCEKS in ***hbase-site.xml*** after the ***Manage credential encryption*** action

Couldn't add the ***Spark3 Client*** component to a cluster with Ozone if the ***spark.yarn.archive*** and ***spark.history.fs.logDirectory*** parameters were set to the OFS path

Trino Hive/Iceberg catalogs depended on HDFS service

Core: incorrect `any` values of the Kerberos-related parameters after installation in a kerberized cluster

Ozone: HttpFS UI didn't work with Kerberos

Incorrect errors while running the ***Export configuration*** action if a host was unreachable

Trino failed with enabled Kerberos and the `TASK` retry policy

Issues integrating Impala and an external object storage

Enabling the Ranger plugin for Solr failed

ADB Spark3 connector: the ***usehostname*** parameter didn't affect the explicitly specified node address

The ***Manage SSL*** action with the ***force*** parameter set to `true` didn't update configuration files if another keystore/truststore password was already added

`Admprom set_ssl_certs_rights` wasn't working correctly for the Prometheus component

Changing `nginx_http_port` did not release the default port 8089

Error when using DistributedShell in YARN

Rager Trino plugin: by default user `rangerlookup` did not have permissions on several policies

Client configurations: some log4j files were missing from the archives

Truststore and keystore passwords were visible after enabling credential encryption

Duplication of *spark-history-env.sh* in the service settings

Spark History Server IllegalArgumentException

Error changing HttpFS log directories

HUE: incorrect protocol in ***webhdfs_url*** after enabling SSL

Duplicate ***Check*** action entries in the HBase actions list

Airflow: ***Manage SSL*** failed on Airflow check

Misc/Internal

Due to the MariaDB service deprecation, it's mandatory to remove it from a cluster with an upgrade

The minimum ADCM version now is 2.7.0

4.0.0

4.0.0.b1

Date: 25.06.2025

New services

Added the Flink History Server component for Flink

Added new Monitoring service. Includes Grafana, Prometheus, and all the necessary exporters and pre-configured dashboards for the following services and components:

- HDFS (NameNode, DataNode, JournalNode)
- Ozone
- HUE
- Kyuubi
- YARN (ResourceManager, NodeManager, TimeLine Server)
- Hive Server
- Hive Metastore
- Impala (Daemon, Catalog, State Store)

The old service is marked as deprecated and will be removed in future releases

Upgrades

Upgraded Trino to 468_arenadata2

Trino updates

- Implemented Trino ADB connector.
- Implemented DBCatalogManager with versioning and maintenance.
- Implemented Catalog Migration functionality.
- Propagating credentials for the JDBC connectors.
- Ozone support.

Upgraded Ozone to 1.4.1_arenadata2

Ozone updates

- Trino support.

Upgraded SSM to 2.1.0

SSM updates

- Support for user impersonation.
- Refactored local file system cache.
- [Bugfix] Table fix on the ***Cluster Info*** page.
- [Bugfix] Filter settings are now reset on the ***Actions*** and ***Rules*** pages.
- [Bugfix] Some actions that started after running sync failed in short time with the `Timeout` error.
- [Bugfix] Failed action’s finish date was the same as start date.
- Added execution start time to the action-related entities.
- Added the `SCHEDULED` logical action state.

Upgraded Impala to 4.5.0_arenadata1

Impala updates

- Apache Hive 4.0.1 support.

Upgraded Iceberg to 1.6.1_arenadata1

Iceberg updates

- Added the `drop.base-directory.enabled` table option to control the removal of table base directory after a table is dropped.

Upgraded Flink to 1.20.1_arenadata1

Flink updates

- Applied patch FLINK-35332.

Upgraded HUE to 4.11.0_arenadata3

HUE updates

- Support for Python 3.10.
- Kerberos authentication support for Trino interpreter.
- [Bugfix] Catalogs were not displayed in Trino interpreter.
- [Bugfix] When executing queries, the status hanged indefinitely and there were errors in the logs for Trino interpreter.

Upgraded Kyuubi to 1.10.1_arenadata1

Kyuubi updates

- [Bugfix] Fixed audit logging to Solr for `spark-authz`.
- Added `OpenMetadataLineageDispatcher` for sending lineage events to OpenMetadata server.
- Added new possible value `OPEN_METADATA` for the `spark.kyuubi.plugin.lineage.dispatchers` option.

Upgraded Spark to 3.5.4_arenadata1

Spark updates

- Now a purge is forced during table/partition drop in case the `external.table.purge` table option is present.
- Purgeable external tables are used for path updates and partition handling when renaming tables.
- Applied patch SPARK-51795.
- Applied patch SPARK-51821.
- Applied patch SPARK-52025.
- Applied patch SPARK-43221.
- Applied patch SPARK-51638.
- Applied patch SPARK-44756.
- Applied patch SPARK-50648.
- Applied patch SPARK-50853.
- Applied patch SPARK-45991.
- Applied patch SPARK-51272.
- Applied patch SPARK-48218.
- Applied patch SPARK-43301.
- Applied patch SPARK-44247.
- Applied patch SPARK-46718.
- Applied patch SPARK-47531.

Upgraded ADQM Spark connector to 1.0.0-3.5.4_arenadata1

Upgraded ADB Spark connector to 1.0.5-3.5.4_arenadata1

ADB Spark connector updates

- Moved transformation of the JDBC `ResultSet` rows iterator to `Seq` in the `JdbcService.extractData()` method to properly close resources.
- Support for the `gp_parallel_retrieve_cursor` read mode.
- Added the support of aggregate pushdown.
- Added the support of `SupportsPushDownTopN`, `SupportsPushDownOffset`, and `SupportsPushDownLimit` to `BatchReader`.

Upgraded Livy to 0.8.0_arenadata1

Upgraded Hive to 4.0.1_arenadata1

Hive updates

- [Bugfix] Explicitly specified SQL types for DB, table, and partition parameters in `removeTxnComponents` for compaction.
- [Bugfix] Added a `sanitize` error message in compaction queue update.
- [Bugfix] Generated the `utf8Sanitize` utility method and used it in `CompactionTxnHandler`.

Upgraded Tez to 0.10.4_arenadata1

New features

Eliminated interservice dependencies and added the ability to install multiple clusters with the required set of services and manual integration

The bundle now includes extensions for integrating Flink with Kafka — `flink-kafka-connector` 3.4.0

PyFlink is now available out of the box

HUE now has a pre-configured Trino interpreter

It is now possible to download client configurations from a cluster for HDFS, YARN, Hive, Solr, Core, HBase, Ozone, Flink, Impala, and Spark3

Added the ability to automatically install Java from the Arenadata repository

Added HA support for Impala State Store

Added HA support for Impala Catalog

Implemented Ranger plugin management for Trino

The management capabilities of Trino have been expanded. The following parameters are provided for configuration:

- logging settings;
- Trino fault-tolerant execution control with preset values for current storage (HDFS/Ozone);
- session property settings;
- resource group settings.

Added the ability to generate SQL Lineage and integrate with Open Metadata for Kyuubi when using the SparkSQL engine

Trino now supports Ozone via the `ofs` protocol

Added the ability to perform rolling restart for HDFS, ZooKeeper, YARN, and HBase

Implemented support for impersonation to perform actions in SSM

Implemented the ability to configure Ozone Topology awareness

Added the ability to manage logs from ADCM for Kyuubi

Added the ability to enable and configure LDAP authentication for Trino from ADCM

Improvements

The ***Add***, ***Remove***, ***Shrink***, ***Expand***, and ***Move*** actions have been replaced with the ***Add/Remove components*** action for Kyuubi, Airflow, ADPG, Solr, Impala, SSM, Zeppelin, Zookeeper, Flink, HBase, Ozone

All the Trino configuration properties are now explicitly presented in ADCM

The ***Manage SSL*** action now also manages the state of the SSM service

Trino preconfigures `discovery.uri` when using the SSH hostprovider

Improved the ability to run Flink in a YARN cluster

The web links update has been added to the ***Start*** and ***Restart*** component actions

When adding/removing services/components, you can now disable service checking to perform the action faster

Kyuubi now supports the Trino engine

Added hints to the ***Add/modify node-to-labels mapping*** and ***Add/modify node labels*** actions

Added the ability to edit the */etc/security/limits.d/om.conf* file for Ozone in ADCM

Added the `rest.port` parameter into ***flink-conf.yaml*** for Flink

Added the `Check` job in the ***Manage SSL*** action for HDFS

The ADB Spark connector now respects the order of user-provided schema columns in the ADB batch entry

Bundle now manages directory creation from configuration parameters for the following services:

- Trino — *node.data-dir*
- YARN — *yarn.nodemanager.local-dirs*, *yarn.nodemanager.log-dirs*, *mapreduce.cluster.local.dir*, *yarn.timeline-service.leveldb-timeline-store.path*, *yarn.nodemanager.recovery.dir*, *ranger.plugin.yarn.policy.cache.dir*, *ranger-yarn-audit.xml/xasecure.audit.destination.solr.batch.filespool.dir*
- HDFS — *dfs.namenode.name.dir*, *dfs.datanode.data.dir*, *dfs.journalnode.edits.dir*, *dfs.namenode.checkpoint.dir*, *ranger.plugin.hdfs.policy.cache.dir*, *ranger-hdfs-audit.xml/xasecure.audit.destination.solr.batch.filespool.dir*

Further configuration and cluster topology changes are also taken into account

Bug fixes

If Trino components had user-defined working directories, `systemd` would continually restart them

Couldn’t change the SSL options for Hive catalog in Trino

Spark3: no web links were updated after removing the Livy and Connect components

Zeppelin: queries in the Spark interpreter failed because Zeppelin couldn’t create a directory

impalarc wasn’t present on the hosts to which Impala Client was expanded

The *log4j2-repl.xml* file for Kyuubi had an incorrect template

Installing a standalone Hive Metastore failed with an error

Hive: no web links were updated after removing the Hive TezUI component

An error while removing a decommissioned YARN NodeManager component from a host in maintenance mode

Spark AuthZ plugin did not create audit logs

Spark3 Livy did not work with Ozone

Incorrect parsing of arguments for the `-q` flag in `impala-shell`

[ADB Spark connector] The service did not work in some cases when the Spark Ranger plugin was enabled

[ADB Spark connector] Reading from a table returned no results if the partitions count was explicitly set to 1

[ADB Spark connector] `DataFrame.count()` failed if the partition count was manually set

Misc/Internal

Starting with the 4.0.0 version, the distribution's name has changed from Arenadata Hadoop (ADH) to Arenadata Hyperwave (ADH). Nevertheless, it's still possible to perform a direct upgrade

Due to the Sqoop and Spark2 services deprecation, it's mandatory to remove them from the cluster with the upgrade. The Spark3 Thrift Server component is also being deprecated, and it's recommended to switch to Kyuubi

The ability to automatically install Java from the Arenadata repository when installing a cluster was added in this release. However, Arenadata is not an official JDK provider and this functionality is provided to enable fully automatic installation on systems that do not have the necessary dependencies. In production environments, we recommend using JDKs from official vendors

The minimum ADCM version now is 2.5.0

Implemented package refactoring to eliminate interservice dependencies and configuration generation logic. The current major upgrade may require more time to fully update packages on all hosts. To increase the update speed, it is recommended to increase the number of Ansible forks in the cluster configuration

3.3.6.2

3.3.6.2.b1

Date: 31.01.2025

New services

Ozone 1.4.1_arenadata1

- Implemented security functionality: support for TLS, Kerberos, ADPS integration with a Ranger plugin, SPNEGO.
- Implemented Ozone HttpFS integration with HUE.
- Implemented Ozone topology awareness settings via ADCM.
- Implemented support for the Ozone write pipeline v2.
- Implemented the S3 Gateway support.
- Some limitations are still in place, see ADH known issues.

The following patches are also included:

Ozone patches

- HDDS-11543: Track OzoneClient object leaks via LeakDetector framework
- HDDS-11574: Ozone client leak in TestS3SDKV1
- HDDS-8005: [disabled] Intermittent failure in TestOmSnapshot.testSnapDiffWithMultipleSSTs/
- HDDS-10148: TestOmSnapshotFsoWithNativeLib should be tagged as native test/
- HDDS-10990: Fix memory leak in native lib/
- HDDS-10560: Link rocksdb lib to Ozone rocksdb tools lib relative path instead of absolute path
- HDDS-3415: Enable TestContainerPlacement#testContainerPlacementCapacity
- HDDS-11347: Add rocks_tools_native lib check in Ozone CLI checknative subcommand
- HDDS-10118: hdds-rocks-native fails to build with Java11+

Trino 468_arenadata1

- Implemented security functionality: support for TLS, Kerberos, SPNEGO.
- Implemented pre-configuration of internal catalogs (Iceberg, Hive).
- Added the ability to use a Java version separate from the rest of the cluster for the service (manual installation is required).
- Implemented the ability to manage Trino catalogs via ADCM.

Core configuration

- The Hadoop configuration has been moved to a service independent of HDFS.

Upgrades

Upgraded HUE to 4.11.0_arenadata2

HUE updates

- The HUE JDBC interface did not support retrieving table metadata from ADB.
- The Impala jobs did not work with SSL.

Upgraded Kyuubi to 1.9.2_arenadata2

Kyuubi updates

- Added support for handling multiple Java options in HiveProcessBuilder.

Upgraded SSM to 2.0.1

SSM updates

- SSM 2.0.0
- SSM 2.0.1

Upgraded Spark3 to 3.5.2_arenadata2

Spark3 updates

- Added the `TRUNCATE` support with Hive4.
- Added the `DROP` support with Hive4.

New features

Implemented the SPNEGO authentication for the Impala service

Implemented support for RedOS 7.3 in the Enterprise version of the ADH bundle

Implemented the ability to import an ADH cluster into ADS for Kafka tiered storage configuration

Improvements

The ***Add***, ***Remove***, ***Shrink***, ***Expand***, and ***Move*** actions have been replaced with the ***Add/Remove components*** action for Kyuubi, Sqoop, Airflow, ADPG, Solr, Impala, SSM, Zeppelin, ZooKeeper, Flink, and HBase

The visible password was removed when Samba Kerberos was enabled

Removed empty and logically redundant tasks related to Kerberos from the logs when deleting components

Corrected services where `bigtop-detect-javahome` is not used (ZooKeeper, SSM, Livy, and jmxtrans)

Corrected the naming of the ***Maintenance/Decommiss DataNodes*** and ***Decommiss/Recommiss NodeManagers*** actions in HDFS

Improved the usability of the ***Enable custom ulimits*** parameter

Added the ability to manage *log.conf* via ADCM for HUE

Added the ability to manage ***impalad_flags*** for the Impala interpreter via ADCM for HUE

Improved management of the **-env.sh* files for HBase

Added custom configuration for *ranger-solr-audit.xml*, *ranger-solr-security.xml*, and *ranger-solr-policymgr-ssl.xml* via ADCM for Solr service

Added the ability to manage `llma-site` and `fair-scheduler` templates via ADCM for Impala

Added database, user, and grants creation for HUE when the ***Restart*** action is performed with the ***Metastore DB schema init/upgrade*** flag is active

Provided the ability to configure the following parameters in the configuration group for Hive:

- `hive.server2.support.dynamic.service.discovery`

HDFS did not restart after disabling its encryption with the ***Manage credential encryption*** action

Bug fixes

The JDBC interface did not support retrieving table metadata from ADB/Greenplum in HUE

The Impala jobs did not work with SSL in HUE

Inability to execute commands in Hive JDBC after moving Kyuubi to a new host

Inability to perform `TRUNCATE` with Hive4 in Spark3

Inability to perform `DROP` with Hive4 in Spark3

The `desktop.http_port` parameter had no effect for HUE

ZooKeeper didn't get disabled after being moved/removed from hosts

The JDBC connection did not work with the `THRIFT_HTTP` mode in Kyuubi

The keystore and truststore were not hidden after enabling encryption

The ***Add/Remove*** actions without changing host-component mapping had excessive tasks

The ADCM status checker did not start with the host boot

The `no password` option was not applied in Solr after enabling encryption

Error launching the Solr's ***Add Server(s)*** action on an empty host with SSL enabled

The ***Manage credential encryption*** action failed if a selected service wasn't installed

The Hadoop services didn't have rights to the */var/lock/subsys/* folder during installation

Misc/Internal

The Java 23 version is required for the Trino service. Manual installation on the hosts is required. Also, add the installation path to the cluster configuration (`TRINO_JAVA_HOME`)

For Ozone kerberization, it is necessary to install it in a kerberized cluster

The minimum ADCM version now is 2.4.0

Marked as deprecated and will be removed in future releases — Spark 2 service, Sqoop service, Spark3 Thrift Server (a Spark3 component)

When upgrading to this version, you have to manually install the Core configuration service

3.3.6.1

3.3.6.1.b1

Date: 30.10.2024

Upgrades

Upgraded HDFS to 3.3.6_arenadata1 with the patches listed below.

Hadoop patches

- HDFS-17299: HDFS is not rack failure tolerant while creating a new file
- HDFS-17453: IncrementalBlockReport can have race condition with Edit Log Tailer
- YARN-10377: Clicking on queue in Capacity Scheduler legacy ui does not show any applications
- YARN-11107: When NodeLabel is enabled for a YARN cluster, AM blacklist program does not work properly
- HDFS-17024: Potential data race introduced by HDFS-15865
- HDFS-16316: Improve DirectoryScanner: add regular file check related block
- YARN-10975: EntityGroupFSTimelineStore#ActiveLogParser parses already processed files
- YARN-9877: Intermittent TIME_OUT of LogAggregationReport
- HDFS-16676: DatanodeAdminManager$Monitor reports a node as invalid continuously
- YARN-11073: Avoid unnecessary preemption for tiny queues under certain corner cases
- YARN-10855: yarn logs cli fails to retrieve logs if any TFile is corrupt or empty
- YARN-10652: Capacity Scheduler fails to handle user weights for a user that has a "." (dot) in it
- YARN-10442: RM should make sure node label file highly available
- YARN-10448: SLS should set default user to handle SYNTH format
- HDFS-15543: RBF: Write Should allow, when a subcluster is unavailable for RANDOM mount points with fault Tolerance enabled
- HADOOP-18613: Upgrade ZooKeeper to version 3.8.3
- HADOOP-19084: prune dependency exports of hadoop-* modules
- HADOOP-18487: Make protobuf 2.5 an optional runtime dependency
- YARN-11657: Remove protobuf-2.5 as dependency of hadoop-yarn-api
- YARN-8056: Possible NPE when start Application Master

Also guava was upgraded to 32.0.1-jre

Upgraded Hive to 4.0.0_arenadata1 with the patches listed below.

Hive patches

- HIVE-28211: Restore hive-exec-core jar
- HIVE-28323: Iceberg: Allow reading tables irrespective whether they were created with hive engined enabled or not
- HIVE-28306: Iceberg: Return new scan after applying column project parameter
- HIVE-28291: Fix Partition spec is incorrect in getPartitionsByFilter RPC
- HIVE-28286: Add filtering support for get_table_metas API in Hive metastore
- HIVE-28282: Merging into iceberg table fails with copy on write when values clause has a function call
- HIVE-28278: Iceberg: Stats: IllegalStateException Invalid file: file length 0
- HIVE-28271: DirectSql fails for AlterPartitions
- HIVE-28270: Fix missing partition paths bug on drop_database
- HIVE-28266: Iceberg: select count(*) from data_files metadata tables gives wrong result
- HIVE-28260: CreateTableEvent wrongly skips authorizing DFS_URI for managed table
- HIVE-28225: Iceberg: Delete on entire table fails on COW mode
- HIVE-28202: Incorrect projected column size after ORC upgrade to v1.6.7
- HIVE-28224: Upgrade Orc version in Hive to 1.9.3

- HIVE-28248: Upgrade opencsv to v5.9
- HIVE-27653: Iceberg: Add conflictDetectionFilter to validate concurrently added data and delete files
- HIVE-28364: Iceberg: Upgrade iceberg version to 1.5.2
- HIVE-28119: Iceberg: Allow insert clause with a column list in Merge query not_matched condition
- HIVE-28352: Schematool fails to upgradeSchema on dbType=hive
- HIVE-28369: LLAP proactive eviction fails with NullPointerException
- HIVE-28254: CBO (Calcite Return Path): Multiple DISTINCT leads to wrong results
- HIVE-28196: Preserve column stats when applying UDF upper/lower
- HIVE-28190: Materialized view rebuild lock heart-beating is broken
- HIVE-28166: Iceberg: Truncate on branch operates on the main table
- HIVE-28121: Use direct SQL for transactional altering table parameter
- HIVE-28098: Fails to copy empty column statistics of materialized CTE
- HIVE-28082: HiveAggregateReduceFunctionsRule could generate an inconsistent result
- HIVE-27847: Prevent query Failures on Numeric <→ Timestamp
- HIVE-26018: The result of UNIQUEJOIN on Hive on Tez is inconsistent with that of MR
- HIVE-23772: Relocate calcite-core to prevent NoSuchFiledError

Also fixed the `java.sql.Driver: Provider org.apache.calcite.jdbc.Driver` not being found issue and implemented the Impala support

Upgraded Tez to 0.10.3_arenadata1

Upgraded Impala to 4.4.0_arenadata2 with the patches listed below.

Impala patches

- IMPALA-13272: Analytic function of collections can lead to crash
- IMPALA-13252: Filter update log message prints TUniqueId in non-standard format
- IMPALA-13519: Running queries been cancelled after statestore failover
- IMPALA-12712: INVALIDATE METADATA <table> should set a better createEventId
- IMPALA-13034: Add logs for slow HTTP requests dumping the profile
- IMPALA-13270: Bug when comparing ExprSubstitutionMap.size()
- IMPALA-13138: Never smallify existing StringValue objects, only new ones during DeepCopy
- IMPALA-13107: Invalid TExecPlanFragmentInfo received by executor with instance number as 0
- IMPALA-13129: Hit DCHECK when skipping MIN_MAX runtime filter
- IMPALA-13130: Under heavy load, Impala does not prioritize data stream operations
- IMPALA-13161: impalad crash — impala::DelimitedTextParser<true>::ParseFieldLocations
- IMPALA-13150: Possible buffer overflow in StringVal::CopyFrom()
- IMPALA-13170: InconsistentMetadataFetchException due to database dropped when showing databases
- IMPALA-9441: TestHS2.test_get_schemas is flaky in local catalog mode
- IMPALA-13203: ExprRewriter did not rewrite 'id = 0 OR false' as expected
- IMPALA-13119: CostingSegment.java is initialized with wrong cost
- IMPALA-13077: Equality predicate on partition column and uncorrelated subquery doesn't reduce the cardinality estimate
- IMPALA-12800: Queries with many nested inline views see performance issues with ExprSubstitutionMap
- IMPALA-8042: Better selectivity estimate for BETWEEN

Also implemented support for Hadoop 3.3.6, Hive 4.0.0, and Hive Driver 4.0.0

Upgraded Spark 3 to 3.5.2_arenadata1 with the patches listed below.

Spark 3 patches

- SPARK-47679: Use `HiveConf.getConfVars` or Hive conf names directly
- SPARK-47468: Exclude `logback` dependency from SBT like Maven
- SPARK-49197: Redact `Spark Command` output in `launcher` module
- SPARK-46037: When Left Join build Left, ShuffledHashJoinExec may result in incorrect results

Also implemented the support for Hive 4.0.0, added the `spark.sql.security.confblacklist` blacklist filter parameter to the ***spark-defaults.conf*** group in the service configuration to exclude certain properties, added a configurable directory path for Spark Connect, and fixed the driver logging

Upgraded ADQM Spark Connector to 1.0.0_3.5.2

Upgraded ADB Spark Connector to 1.0.5_3.5.2

Upgraded Flink to 1.19.1_arenadata1 with support for Hive 4.0.0

Upgraded HBase to 2.5.10_arenadata1 with support for Hive 4.0.0

Upgraded hbase-operator-tools to 1.2.0_arenadata3

Upgraded phoenix-queryserver to 6.0.0_arenadata3

Upgraded Phoenix to 5.2.0_arenadata1

Upgraded Iceberg to 1.5.2_arenadata1

Upgraded SSM to 2.0.0-alpha with the following features:

- Introduced new UI and removed the Zeppelin dependency.
- Support for HDFS 3.3.6.
- Optimized access count strategy.

Upgraded Kyuubi to 1.9.2_arenadata1 with the following features:

- Support for Hive 4.0.0.
- Added an additional KYUUBI_ADDITIONAL_CLASSPATH environment variable.
- AuthZ fixes.
- Upgraded the Spark version.
- The Flink engine module preliminary now supports building with Scala 2.13.

Upgraded Solr to 8.11.3_arenadata1

Upgraded ZooKeeper to 3.8.4_arenadata1

Upgraded Sqoop to 1.4.7_arenadata3

Upgraded ADPG to 16

New features

Added support for Ubuntu 22.04.2 LTS

Added a new cluster action — Manage Credential Encryption. It encrypts sensitive data in the configuration files for HDFS, YARN, Hive, HBase, Spark, Impala, Zeppelin, Kyuubi, and Solr

Implemented the ability to enable the SSL/TLS protocol for inter-component communication for the Flink service

Implemented the HUE SPNEGO support

The distribution includes the following extensions and formats for Flink:

- The Flink Kafka extensions.

- The Flink Iceberg extensions.
- The Flink Hive extensions.
- The parquet format.
- The orc format.
- The avro format.
- The protobuf format.

Added log settings for Livy in ADCM

Added the ***Custom nginx.conf*** checkbox parameter with template for the Hive service

Now all the `proxyusers` HDFS parameters are displayed explicitly

Improvements

Changed the `mem_limit` Impala Daemon component configuration parameter type to `string`

Changed classpaths properties to array type for Flink, Spark, Hive, Kyuubi, and Hive

The ***Add***, ***Remove***, ***Shrink***, ***Expand***, and ***Move*** actions have been replaced with the ***Add/Remove components*** action for Spark2, Spark3, Hive, and YARN

Hive now uses ADPG as the default metastore during installation

`kyuubi.authentication.ldap.binddn` is no longer a required parameter

Actualized Flink configuration:

- The following parameters were added:
  - `taskmanager.memory.flink.size`
  - `taskmanager.memory.process.size`
  - `jobmanager.memory.flink.size`
  - `jobmanager.memory.process.size`
  - `jobmanager.memory.heap.size`
  - `flink.yarn.appmaster.vcores`
  - `flink.yarn.container.vcores`
- The following parameters were removed:
  - `jobmanager.heap.size`
  - `taskmanager.heap.size`

Added the new sections for YARN in ADCM: ***mapred-env.sh***, ***Custom mapred-env.sh***

Relocated the `HADOOP_JOB_HISTORYSERVER_OPTS` parameter from ***yarn-env.sh*** to ***mapred-env.sh*** for YARN

Added the ***Remove*** action for Airflow and Zeppelin in the community edition

Added the ***Custom yarn-env.sh*** parameter to the YARN configuration

Added notifications to services that notify the user of the need to restart if the topology of a dependent service has changed

Bug fixes

Errors when connecting an external database to the SSM service

Incorrect process ID was written to the PID file for the SSM service

Error related to an undefined variable while enabling the YARN Ranger plugin

Spark3 Thrift Server didn't restart

Kyuubi: the ***Check*** action failed if only the REST protocol was enabled

The `Install CE` error on Spark3 Connect check

The incorrect permissions on container-executor binary

Impala: couldn't enable Ranger on some topologies

HUE: the bundle didn't change `spark_history_server_security_enabled`

Flink: excessive parameters on host when HA was disabled

Incorrect auth-to-local rules in the Spark3 History Server process

Invalid LDAP authentication configuration for Airflow

Misc/Internal

The SSM service received a major upgrade that is not compatible with the previous version. During the update, all security settings will change and the metastore will be cleaned and migrated

The minimum ADPS version is 1.2.0_b1

The minimum ADCM version now is 2.2.0

3.2.4.3

3.2.4.3.b1

Date: 23.07.2024

New features

Added the new HUE service

Bumped Zeppelin to 0.11.1_arenadata1

Bumped Spark3 to 3.4.3_arenadata1 and fixed the Hive version regex

Bumped Kyuubi to 1.9.0_arenadata1 and added the missing Impala alias for the JDBC engine

Upgraded Impala to 4.4.0_arenadata1 with the following patches:

- IMPALA-13035: Querying metadata tables from non-Iceberg tables throws IllegalArgumentException
- IMPALA-13040: SIGSEGV in QueryState::UpdateFilterFromRemote

Added the Kyuubi AuthZ plugin for Spark3

Implemented support for the Samba domain controller

Implemented the Apache Iceberg support for Spark3

Implemented the Apache Iceberg support for Hive

Added the Spark3 and PySpark support for Zeppelin

- Implemented the LDAP authentication for Kyuubi
- Implemented the LDAP authentication for Impala

Improvements

- A new ***Manage SSL*** action has been added instead of ***Enable/Disable SSL***, which allows you to more conveniently manage the SSL encryption of all services in an ADH cluster
- Custom properties will now overwrite the existing ones, even if they are read-only
- Provided the ability to add an additional Livy Server for Spark3
- Added the ability to specify multiple directories for Impala `scratch_dirs`
- Now the Arenadata PostgreSQL service cannot be removed when Airflow2 is installed
- Now the `hadoop.proxyuser` parameter is registered when adding a new Kyuubi service
- Now the `hadoop.kms.proxyuser.kyuubi.*` parameter in Ranger's *kms-site.xml* is registered for the Kyuubi service
- Setting the `IMPALA_SHELL_HOME` environment variable is no longer required to launch the Impala shell

Bug fixes

- Couldn't change a web port for Solr
- No changes were applied to the log4j configuration for the HBase service
- The Kerberos parameters disappeared from the configuration file after Spark3 History Server restart
- The ***Restart YARN*** action didn't work when the `local-dirs` was unavailable
- Minor errors during Impala installation on Red Hat
- Impala service configuration wasn't applied during a cluster start
- The ***Precheck packages*** parameter being set to `true` led to upgrade errors

3.2.4.2

3.2.4.2.b2

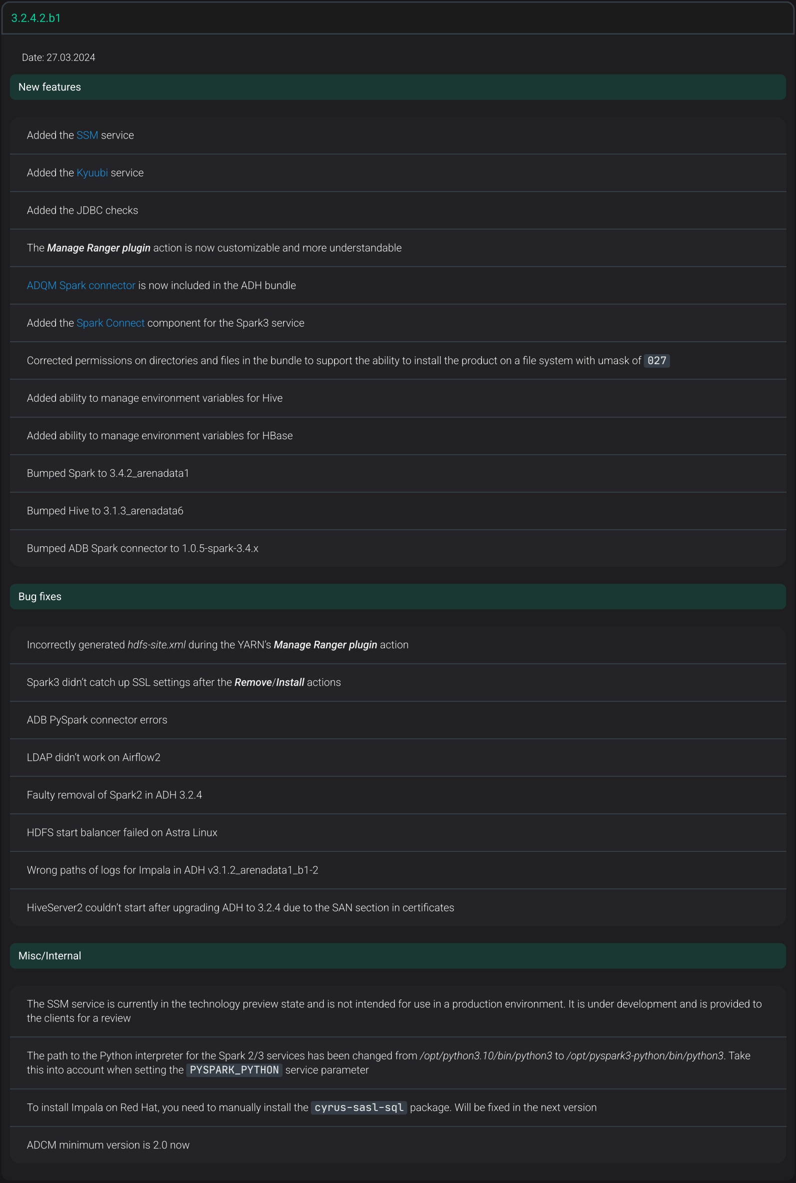
Date: 27.04.2024

Improvements

- Removed the need to install Axiom JDK when using Astra Linux
- Added the ability to set a custom value to the `JAVA_HOME` variable

Bug fixes

- Configuration for Shiro Simple Authentication displayed passwords in plain text
- Package conflict when updating ADH and installing the SSM service



3.2.4.2.b1

Date: 27.03.2024

New features

- Added the SSM service
- Added the Kyuubi service
- Added the JDBC checks
- The ***Manage Ranger plugin*** action is now customizable and more understandable
- ADQM Spark connector is now included in the ADH bundle
- Added the Spark Connect component for the Spark3 service
- Corrected permissions on directories and files in the bundle to support the ability to install the product on a file system with umask of `027`
- Added ability to manage environment variables for Hive
- Added ability to manage environment variables for HBase
- Bumped Spark to 3.4.2_arenadata1
- Bumped Hive to 3.1.3_arenadata6
- Bumped ADB Spark connector to 1.0.5-spark-3.4.x

Bug fixes

- Incorrectly generated *hdfs-site.xml* during the YARN’s ***Manage Ranger plugin*** action
- Spark3 didn’t catch up SSL settings after the ***Remove***/***Install*** actions
- ADB PySpark connector errors
- LDAP didn’t work on Airflow2
- Faulty removal of Spark2 in ADH 3.2.4
- HDFS start balancer failed on Astra Linux
- Wrong paths of logs for Impala in ADH v3.1.2_arenadata1_b1-2
- HiveServer2 couldn’t start after upgrading ADH to 3.2.4 due to the SAN section in certificates

Misc/Internal

- The SSM service is currently in the technology preview state and is not intended for use in a production environment. It is under development and is provided to the clients for a review
- The path to the Python interpreter for the Spark 2/3 services has been changed from */opt/python3.10/bin/python3* to */opt/pyspark3-python/bin/python3*. Take this into account when setting the `PYSPARK_PYTHON` service parameter
- To install Impala on Red Hat, you need to manually install the `cyrus-sasl-sql` package. Will be fixed in the next version
- ADCM minimum version is 2.0 now

3.2.4.1

3.2.4.1.b3

Date: 27.04.2024

Improvements

Removed the need to install Axiom JDK when using Astra Linux

3.2.4.1.b2

Date: 16.01.2024

Bug fixes

Faulty Spark 2/3 configuration when kerberizing

3.2.4.1.b1

Date: 26.12.2023

New features

Upgraded Flink to 1.17.1_arenadata1 with the FLINK-32976 patch

Upgraded Tez to 0.10.1_arenadata1

Upgraded ADB Spark Connector to 1.0.5-spark-3.3.x with performance boosts, security improvements, and bug fixes

Added jdbc-tools 1.0 package

Upgraded Spark2 to 2.3.2_arenadata2 with the SPARK-31644 patch

Upgraded Hadoop to 3.2.4_arenadata1 with the following patches:

- HDFS-14768: Busy DN replica should be consider in live replica check
- HDFS-15186: Erasure Coding: Decommission may generate the parity block's content with all 0 in some case
- HADOOP-17340: TestLdapGroupsMapping failing -string mismatch in exception validation
- HADOOP-15783: TestSFTPFileSystem.testGetModifyTime fails
- YARN-9554: TimelineEntity DAO has java.util.Set interface which JAXB can't handle

Added zstd support in HDFS

Added PMDK support in HDFS

Upgraded Spark3 to 3.3.2_arenadata1 with the SPARK-39910 patch (internal development)

Upgraded Spark3 Livy to 0.7.2_arenadata5

Upgraded Sqoop to 1.4.7_arenadata2

Upgraded Phoenix Query Server to 6.0.0_arenadata2

Upgraded Phoenix to 5.1.3_arenadata2

Upgraded HBase operator tools to 1.2.0_arenadata2

Upgraded Hive to 3.1.3_arenadata5 with the following patches:

- HIVE-20344: PrivilegeSynchronizer for SBA might hit AccessControlException
- HIVE-21844: HMS schema Upgrade Script is failing with NPE

Upgraded HBase to 2.4.17_arenadata1

Upgraded Airflow to 2.6.3

Excluded the vulnerability of the log4j library up to the version 2.15

Added log settings for Solr in ADCM

Added log settings for ZooKeeper in ADCM

Excluded Airflow1 from the bundle

Added the Thrift Server component for Spark3

Added the template for changing Ulimits to ADCM component configuration

Added support for AstraLinux 1.7 SE "Орел"

Improvements

Reconfigured Hive connection settings to the metadata store database to improve flexibility

Updated the Postgres JDBC driver supplied with the distribution

Now there is no complete recreation of existing keytabs when expanding/installing services

Bug fixes

Error adding roles to *shiro.ini* when setting up Zeppelin

Spark3 Server failed to start after upgrade

Misc/Internal

Because of the Airflow 1 deprecation, it’s mandatory to remove it from the cluster with the upgrade. Airflow 2 can be installed instead

Minimum ADPS version — 1.1.0

3.1.2.1

3.1.2.1.b2

Date: 30.11.2023

Bug fixes

Reconfiguration error due to HDFS being in SafeMode for a while

Error updating repository URL during offline upgrade

Upstream changes related to Phoenix Query Server are taken into account while updating HBase

Errors upgrading Community Edition cluster

3.1.2.1.b1

Date: 20.10.2023

New features

Upgraded Flink to 1.16.2

Incorporated the following upstream updates:

- HIVE-5312: Let HiveServer2 run simultaneously in HTTP (over thrift) and Binary (normal thrift transport) mode
- HIVE-20187: Incorrect query results in hive when hive.convert.join.bucket.mapjoin.tez is set to true
- HIVE-21940: Metastore: Postgres text <-> clob mismatch for PARTITION_PARAMS/PARAM_VALUE
- HIVE-22914: Make Hive Connection ZK Interactions Easier to Troubleshoot
- HIVE-19825: HiveServer2 leader selection shall use different zookeeper znode

Bumped Solr to 8.11.2 with fixed vulnerability of the log4j library

Bumped Sqoop to 1.4.7 with fixed vulnerability of the log4j library

Bumped HBase to 2.2.7

Bumped Phoenix to 5.13

Implemented Impala support

ADB Spark 3 Connector is now included in the ADH bundle

Implemented hbck2 (an HBase component)

Introduced the Maintanence mode which provides an ability to remove any node from a cluster

Intoroduced High Availability auto-management for ADH services

Added the SQL Gateway component for Flink

Added PySpark 3 for customer installation

Added Knox SSO authorization for Zeppelin

Improvements

Tweaked Kerberos management (enable, disable, configure)

Added logging settings for Spark in ADCM

Added the ***Precheck packages*** cluster parameter that enables/disables package checks. By default the checks are disabled

Added logging settings for Sqoop in ADCM

Bug fixes

Airflow 2 services failed after kerberization and restart

Metadata and statistics errors for Hive

Hive configuration failed without an installed ZooKeeper on CE

The problem with config groups for Hive

Spark 3 check failed if Spark 2 was not installed

Inability to remove Spark 3 without mapped components

The problem with HDFS Balancer with enabled Kerberos

Misc/Internal

Minimum ADPS version — 1.0.5

Minimum ADCM version — 2023.10.10.08

2.1.10

2.1.10.b1

Date: 21.06.2023

New features

Added an ability to select a TLS version for ADH services

Added support for custom Zeppelin interpreters

Added the new component Spark History Server for Spark3

The following upstream updates have been incorporated:

- HIVE-20192: HS2 with embedded metastore is leaking JDOPersistenceManager objects
- HIVE-20209: Metastore connection fails for first attempt in repl dump
- HIVE-20511: REPL DUMP is leaking metastore connections
- HIVE-20522: HiveFilterSetOpTransposeRule may throw assertion error due to nullability of fields
- HIVE-20627: Concurrent async queries intermittently fail with LockException and cause memory leak
- HIVE-20682: Async query execution can potentially fail if shared sessionHive is closed by master thread
- HIVE-21018: Grouping/distinct on more than 64 columns should be possible
- HIVE-26743: Backport HIVE-24694 to 3.1.x
- HIVE-21206: Bootstrap replication is slow as it opens a lot of metastore connections
- HIVE-22393: HiveStreamingConnection: Exception in beginTransaction causes AbstractRecordWriter to throw NPE, covering up real exception
- HIVE-22841: ThriftHttpServlet#getClientNameFromCookie should handle CookieSigner IllegalArgumentException on invalid cookie signature
- HIVE-24552: Possible HMS connections leak or accumulation in loadDynamicPartitions
- HIVE-25830: Hive::loadPartitionInternal occur connection leak
- HIVE-25268: date_format udf returns wrong results for dates prior to 1900 if the local timezone is other than UTC
- HIVE-25075: Hive::loadPartitionInternal establishes HMS connection for every partition for external tables

Improvements

ResourceManager high availability mode activates automatically

Spark can work with a custom Hive Metastore

Added SSL support for Hive Metastore

Added the ***Remove*** action for the MariaDB service with the `created` status

Updated Spark3 version to 3.3.2

ZooKeeper: added links to Admin Server endpoints on the ZooKeeper page in ADCM

Enhanced the ***Move*** action behavior

Bug fixes

Fixed: can’t remove faulty installed Airflow 1 after Airflow 2

Zeppelin: fixed incorrect Hive JDBC string with enabled Hive HA

Hive: fixed timezone for `date_format()`

Fixed NiFi hive3streaming `Unable to close` exception

Fixed GROUPING/DISTINCT limitations for Hive tables with 64+ columns

Hive: fixed the `AssertionError` for requests with union

Misc/Internal

ZooKeeper: changed the default port for Admin Server

2.1.8

2.1.8.b3

Date: 02.03.2023

New features

Airflow2: added the high availability mode

Airflow2: added LDAP authentication/authorization support

Airflow2: added support for external broker configuration

Hive version updated to 3.1.3

The following upstream updates have been incorporated:

- HIVE-20007: Hive carries out timestamp computations in UTC
- HIVE-20221: Increase column width for partition_params
- HIVE-20437: Handle schema evolution from float, double and decimal
- HIVE-20833: package.jdo needs to be updated to conform with HIVE-20221 changes
- HIVE-20839: Fix the "Cannot find field" error during dynamically partitioned hash join
- HIVE-21050: Use Parquet LogicalTypes
- HIVE-21215: Read Parquet INT64 timestamp
- HIVE-21837: MapJoin throws an exception when selected column has completely null values
- HIVE-21987: Hive is unable to read Parquet int32 annotated with decimal
- HIVE-22476: Hive datediff function provided inconsistent results when `hive.fetch.task.conversion` is set to none

- HIVE-22540: Vectorization: Decimal64 columns don't work with `VectorizedBatchUtil.makeLikeColumnVector(ColumnVector)`
- HIVE-22589: Add storage support for ProlepticCalendar in ORC, Parquet, and Avro
- HIVE-22648: Upgrade Parquet to 1.11.0
- HIVE-23345: INT64 Parquet timestamps cannot be read into bigint Hive type
- HIVE-24074: Incorrect handling of timestamp in Parquet/Avro when written in certain time zones in versions before Hive 3.x
- HIVE-25054: Upgrade jodd-core due to CVE-2018-21234
- HIVE-25093: `date_format()` UDF is returning output in UTC time zone only
- HIVE-25104: Backward incompatible timestamp serialization in Parquet for certain timezones
- HIVE-25299: Casting timestamp to numeric data types is incorrect for non-UTC timezones
- HIVE-25458: `unix_timestamp()` with string input returns wrong result
- HIVE-25559: `to_unix_timestamp()` udf result is incorrect
- HIVE-25577: `unix_timestamp()` is ignoring the time zone value
- HIVE-26233: Problems reading back PARQUET timestamps above 10000 years
- HIVE-26270: Wrong timestamps when reading Hive 3.1.x Parquet files with the vectorized reader
- HIVE-26320: Incorrect results for IN UDF on Parquet column of CHAR/VARCHAR type
- HIVE-26612: INT64 Parquet timestamps cannot be read into BIGINT Hive type
- HIVE-26658: INT64 Parquet timestamps cannot be mapped to most Hive numeric types
- HIVE-26955: Select query fails when decimal column data type is changed to string/char/varchar in Parquet

Improvements

Changed the `hive.server2.transport.mode` default value to `binary`

The `cluster_version_before_upgrade` variable hidden from *config.yaml.j2*

HDFS: removed RW permissions from the `dfs.permissions.superusergroup` property

Spark3: added ADH python package to the installation routine

Implemented a custom configuration for *container-executor.cfg*

Fixed CORS errors in ResourceManager UI2

Bug fixes

Fixed: no retries for the ***Disable balancer*** task during the HBase shrink action

Fixed: missing timestamps in the ansible log output in ADCM

Fixed: the configuration for checker-thriftserver is not overwritten during reconfiguration

Fixed: the `livy_spark3_move` action finished execution with unexpected result

Fixed: status checker does not run for Spark3 after reinstalling

Fixed the checker-thriftserver conflict with other services

Fixed wrong setting of `dfs.datanode.https.address` when running the ***Disable Kerberos*** action

Fixed NodeName stop sequence for the JournalNode expand action

Fixed: HDFS doesn't stop with enabled Kerberos

Fixed issues with host cert key permissions

- Fixed *.jceks* file permissions
- Fixed: Sqoop Hive import fails with enabled Ranger plugin
- Fixed the NameNode stop action behavior
- Airflow2: fixed support for external database connections
- Airflow2: fixed Redis configuration
- Airflow2: fixed template usage for *airflow.cfg* (*cfg_properties_template*)
- Airflow2: fixed permissions for Redis config directory
- Airflow2: fixed the Logging level list
- Fixed: a failed check if TEZ submit creates an entry in Timelineserver
- Fixed the ZooKeeper *zkCli.sh* error with enabled Kerberos
- Fixed: Spark checks failed after YARN expand
- Fixed errors with disabling Kerberos

Misc/Internal

- ADH MySQL Service: changed the display name and version
- Fixed the description for `name.dirs` in *config.yml*
- Fixed the description for `spark.ssl.trustStore`
- Updated the `hadoop.http.authentication.signature.secret.file` behavior
- Updated offline package versions

2.1.7

2.1.7.b1

Date: 20.12.2022

New features

- Added the livy-spark3 component to the Spark3 service
- Added Hive delegation token
- Added the ***Apply configs from ADCM*** checkbox for all services
- Flink build 1.15.1 is available
- Added the ability to connect to Flink JobManager in the high availability mode

Improvements

- Added package checks optimizations for the installation

Added a cleanup of MIT credentials after disabling Kerberos in an ADH cluster for the following services:

- Hive
- YARN
- HDFS

Passwords hidden for actions ***Config check***, ***Enable SSL*** for the following services:

- YARN
- HBase
- HDFS

Refactored Livy impersonation options

Added the ability to configure Livy impersonation

Added additional `xasecure.add-hadoop-authorization` parameter

The ***Disable HA HiveServer2*** action uses the `deleteall` command instead of `rmr`

Performed actions optimizations

Added the ability to delete a service in the `created` state

Bug fixes

Fixed the High Availability mode activation for YARN ResourceManager

Fixed: failed to enable SSL with Airflow2 on AltLinux

Fixed: Flink TaskManager doesn't start in the High Availability mode

Fixed: no metrics when Monitoring service was installed in a wrong order

Fixed: missing Spark3 actions ***Add Spark3 Livy***, ***Remove Spark3 Livy***

Fixed: Flink upgrade from 2.1.6 to 2.1.7 fails if Job Manager is not collocated with the HDFS Client

Fixed: missing parameter `container-executor.class` in *yarn-site.xml*

Fixed: Flink installation failed

Fixed the ***Manage ranger plugin*** action in HBase

Fixed SSL settings for the following services:

- HDFS
- YARN
- Hive

Fixed: the ***Enable SSL*** action failed due to absence of ranger-hbase-plugin

Fixed the upgrade from 2.1.6.b4-1 to 2.1.7.b1-pre_rc

Fixed: incorrect minimum version for an upgrade

Fixed: the ***Enable HA***/***Disable HA*** actions lead to a configuration divergence of *hive-site.xml* on servers

Fixed the ***Disable HA HiveServer2*** action error

- Fixed HBase SASL error while connecting to a kerberized ZooKeeper (no *hbase-jaas.conf*)
- Fixed the error when expanding Hive Server 2 on a host with Hive Metastore
- Fixed the ***Add Hive Metastore*** action
- Fixed: no description for the ***Reliability Control*** → ***timeout*** field
- Fixed: the Scheduler type does not pass to the yarn_scheduler_jmx filter
- Fixed: passwords shown in the Ansible log during the ***Enable SSL*** action
- Fixed: wrong Spnego keytab value when expanding Hive Server 2
- Fixed components list on the ***Hosts - Components*** page
- Fixed: need to install the jdbc-mysql driver on nodes with the hive-client service when internal MariaDB is used
- The ***Enable Ranger plugin*** action for Hive failed if no YARN service and politics are defined in Ranger
- Fixed: enabling SSL repeatedly failed
- Fixed: the Spark Thriftserver check fails due to FairScheduler being used
- Fixed: YARN reconfiguration fails if FairScheduler is enabled
- Fixed errors with the order of hosts decommissioning
- Fixed the bug that prevented running Flink on Yarn

Misc/Internal

- Added `install_flag: true` to the ***Upgrade*** action
- Fixed naming for service configs (HDFS, HBase, Hive)
- Fixed naming for ADH Service actions
- Fixed naming and typos for ADH install actions
- Changed the sequence of actions for Kerberos, SSL, and cluster installation

2.1.6

2.1.6.b4

Date: 03.11.2022

Bug fixes

- Fixed: wrong Spnego keytab value when expanding Hive Server 2

2.1.6.b3

Date: 28.10.2022

Bug fixes

- Fixed: expand operation with http principal is failed
- Fixed: the Disable/Enable SSL and Kerberos actions available in any state
- Fixed: Spnego error when disabling Kerberos for Hive

2.1.6.b2

Date: 17.10.2022

New features

- Added support for customization of *krb5.conf* via ADCM

Bug fixes

- Fixed: HDFS gets corrupt because of missing Spark files
- Fixed an issue with custom Hive configuration

2.1.6.b1

Date: 16.09.2022

New features

- Added support for AltLinux 8.4
- Added support for customization of *ldap.conf* via ADCM
- Added support for FreeIPA kerberization

Improvements

- Improved errors handling using cURL
- Hive logging refactored

Bug fixes

- Fixed: cannot change parameters in *ranger-yarn-policymgr-ssl.xml* (YARN)
- Fixed: `hive.metastore.sasl.enabled` cannot be changed (Hive)
- Fixed a configuration name after Flink restarts
- Fixed: Zeppelin user does not exist when enabling Ranger for YARN
- Fixed: cannot change nameservice when faulty installed (HDFS)
- Fixed: a cluster upgrade always switches state to `Installed`
- Fixed: a Spark 3 upgrade does not disable old repositories on Alt Linux
- Fixed: the ***Reconfigure Kerberos*** action has no `allow_to_terminate` attribute

- Fixed: YARN Resource Manager does not start on Alt 8.4
- Fixed `ClassNotFoundException` while executing spark-sql with enabled Ranger plugin
- Fixed inconsistency between security settings in Spark and Spark3 configs

Misc/Internal

- Changed check code due to Scala version upgrade

2.1.4

2.1.4.b11

Date: 08.08.2022

New features

- Added the ability to specify external nameservices
- Added the ability to connect to HiveServer2 in the fault-tolerant mode

Improvements

- Cluster components states refactored
- Refactored the order of ***Stop***, ***Start***, and ***Restart*** actions for the HDFS service
- Enhanced monitoring metrics collection by YARN queues
- Removed read-only attribute from the `hadoop.security.auth_to_local` property

Bug fixes

- Fixed the cluster kerberization status error after a bundle upgrade
- Fixed: Ansible variable is not resolved during HDFS installation
- The `hive.zookeeper.quorum` property gets reset on the Hiveserver2 ***Disable*** action
- Fixed permissions for the `dfs.datanode.data.dir` directory

2.1.4.b10

Date: 16.06.2022

New features

- The ability to install ADH components from custom Docker registry is added
- The check box ***Rewrite current service SSL parameters*** is added for the ***Enable SSL*** action
- New parameters are added for the Zeppelin Hive interpreter — in order to enable SSL and Kerberos
- Retries for generating Kerberos principals are implemented

- Custom authentication (LDAP/AD) is enabled for Hive2Server
- The *Move* action is added for Spark Livy Server
- The *Move* action is added for Spark History Server
- The *Move* action is added for Flink Job Manager
- The *Move* action is added for Sqoop Metastore
- The *Move* action is added for YARN Timeline Server
- The *Move* action is added for YARN MapReduce History Server
- The Ranger plugin for Solr authorization is added
- The ability to remove services from the cluster is added
- The ability to customize configuration files via ADCM is added
- The support of Kerberos REALM is added

Improvements

- The Solr connection for audits is changed from Solr server to ZK node
- Changing the property type from `string` to `option` when upgrading a bundle does not reset the property value anymore
- SSL default configuration parameters are changed from invisible to read-only

Bug fixes

- Fixed: duplicate dictionary keys in the *config.yaml* file that did not pass the new YAML validation in ADCM
- Fixed: the error with Zeppelin installation without Hive
- Fixed: users could change some read-only Kerberos-related parameters in services
- Fixed failing jobs when enabling the ***GPU on YARN*** property
- Fixed the error with applying the ***Remove service*** action to Spark3
- Fixed the error with incorrect value in *interpereter.json* in Zeppelin when SSL is off
- Fixed: after enabling SSL policies did not work in Ranger
- Fixed: the Container DN format when applying ***Enable Kerberos***
- Fixed the error with incorrect saving the `HBASE_MASTER_OPTS` value in HBase
- Fixed: YARN failed to copy *container-executor.cfg*
- Fixed: job status and result did not match when deleting optional (unnecessary) settings in the ZooKeeper service configuration
- Fixed: the action ***Check*** applied to Flink failed if hosts in ADCM had uppercase letters
- Fixed: services did not collect policy from Ranger in SSL
- Fixed: applying the action ***Remove internal database*** actually removed the service itself

Misc/Internal

- A fixed Ranger Solr plugins repository is added for ADH 2.1.4
- The order of bundle upgrades is changed from particular to general
- Dependencies between components and services at the ADCM level are implemented
- Ranger Plugins are bumped to 1.0.3
- The ability to download ADH offline packs from the Arenadata source directory to the customer proxy repository is added

2.1.4.b9

Date: 31.03.2022

New features

- The Kerberos authentication is enabled for Web UI
- SSL for Ranger plugins is enabled
- SSL for Flink is enabled
- SSL for Sqoop is enabled
- The rollback operation is enabled in the case of the failed kerberization process
- SSL for Zeppelin is enabled
- SSL for Airflow is enabled
- SSL for Spark is enabled
- SSL for Solr is enabled
- SSL for Hive is enabled
- SSL for HBase is enabled
- SSL for YARN is enabled
- The ability to configure SSL in the Hyperwave clusters is added
- SSL for HDFS is enabled

Improvements

- The ***Custom hive-site.xml*** block is placed after the ***hive-site.xml*** block in the configuration settings
- The links to NameNodes and HttpFS are moved to the top of the HDFS web links list
- The order of cluster stop actions is reversed
- The ***Reconfig and restart*** action is replaced by the ***Restart*** action that runs three operations: stops the service, applies configuration parameters, and starts the service
- The ability to execute the ***resourcemanager_enable_ha*** action without changing `hc_map` is disabled for Resource Manager

- The ability to execute the ***resourcemanager_expand*** action without enabling High Availability (HA) is disabled for Resource Manager, since it does not work without enabling HA

Bug fixes

- Fixed: the parameters from the *httpfs-site.xml* configuration file did not apply to the HttpFS service
- Fixed: SQL queries launched from Spark3 or Spark2 did not work correctly with the Ranger Hive plugin being enabled
- Fixed: the `keystore/truststore` parameters specified in the service settings did not override the default cluster settings during the per-service installation
- Fixed: the Phoenix Query Server could not work in the thin mode in the kerberized environment
- Fixed: the error with running ***spark-thrift-server-checker*** after enabling and disabling Kerberos
- Fixed: the error with saving the Flink configuration parameters after installation in the kerberized environment
- The ability to work with kerberized ADH clusters is fixed for the Windows operation system
- Fixed the `Cluster Monitoring imported, but not installed` error that occurred during the cluster installation
- Fixed the error with checking privileges to the `public` schema of the `ranger` database at the *preconfigure-check* stage
- The cluster installation errors at the ***HBase check*** stage are fixed
- Fixed: application logs were unavailable in the legacy Resource Manager UI of the kerberized clusters
- Fixed: Kerberization on AD failed if two instances of Ranger Admin were installed

Misc/Internal

- Configuring SSL settings is added before enabling SSL in autotests
- Web links are rewritten to support the http/https schema change

2.1.4.b6

Date: 21.12.2021

Improvements

- Refactoring of the database for Hive Metastore checks is done

Bug fixes

- The error with the `kdc_type` parameter being set to `null` after the cluster upgrade is fixed
- Fixed the error with MapReduce jobs launched in the kerberized cluster not under the `yarn` user
- Fixed: mapped but not installed services caused the errors via installation of other services

2.1.4.b5

New features

- The HTTP mode is added for HiveServer2

- The AD/LDAP/SIMPLE authorization is added for Zeppelin
- The HBase REST Server component is added for HBase
- The ability to use Active Directory as Kerberos storage is implemented

Improvements

- The ability to set Kerberos principal for running Spark jobs via YARN is added. Before that Spark always launched tasks using the `yarn` principal

Bug fixes

- Fixed the error with DataNodes expanding via the ***Add DataNode*** action in the kerberized environment
- Fixed the error with the Livy interpreter working in the kerberized environment
- Fixed the error with the Hive interpreter working in the kerberized environment
- Fixed the error with Zeppelin checks after Kerberos activation
- Fixed the error with removing the monitoring component jmxtrans
- Fixed: the ***Enable Resource Manager HA*** action failed in the clusters with Kerberos and Ranger plugin being enabled
- Fixed the error with Airflow installation in the kerberized environment
- Fixed the error with enabling Kerberos after its disabling (enable → disable → enable)

Misc/Internal

- The full stack testing for using the RedHat 7.9 enterprise license in ADH is added

2.1.4.b4

Date: 01.11.2021

New features

- The ***Reinstall status-checker*** action is implemented. It runs the *status-checker* deployment scripts for services as well as for Docker containers

Improvements

- The Solr check is changed: the number of live Nodes is compared instead of lists
- The timeout/retry count is increased for the Zeppelin check

Bug fixes

- Fixed: the bundle version update error
- Fixed: the Ranger plugin worked incorrectly in the case of using some characters in the cluster name
- Fixed: the error with the inconsistent state of the actual DataNodes maintenance state after upgrading ADH from 2.1.3 to 2.1.4
- Fixed: Keytabs permissions changed during some actions

- Fixed: the error with per-service installation in the kerberized ADH clusters
- Fixed: the error with parsing the list of containers by the Docker status checker in Airflow

Misc/Internal

- The error with the heap size test is fixed
- The broken compatibility with the current dev version of ADCM is fixed
- The test logic for per-service installation in the kerberized ADH clusters is changed: before each service installing it is necessary to add the service to the cluster and add its components to hosts (instead of adding all components to all hosts)

2.1.4.b3

Date: 30.09.2021

New features

- The MIT Kerberos integration is implemented in ADCM
- The ability to add the custom port for Kerberos Server is added
- Ranger plugin and kerberized YARN are integrated
- Ranger plugin and kerberized Hive are integrated
- Ranger plugin and kerberized HBase are integrated
- Ranger plugin and kerberized HDFS are integrated
- The Ranger plugin is made operable on kerberized services
- The split memory option is added for Hive services: resource management options can be configured for HiveMetastore and HiveServer2 separately
- The edit memory size option is added for Flink components
- The edit memory size option is added for Solr components
- The edit memory size option is added for Sqoop components
- The edit memory size option is added for Spark components
- The edit memory size option is added for Zeppelin components
- The edit memory size option is added for HBase components
- The edit memory size option is added for YARN components
- The ***Add/Remove*** actions are added for YARN Timeline server
- The ***Add/Remove*** actions are added for Sqoop Metastore
- The edit memory size option is added for HDFS components
- The ADH memory management option is added
- The ***Add/Remove*** actions are added for Flink Job Manager
- The ***Add/Remove*** actions are added for Spark Thrift Server

- The ***Add/Remove*** actions are added for Spark Livy Server
- The ***Add/Remove*** actions are added for Spark History Server
- The ***Add/Remove*** actions are added for Hive Tez UI
- The ***Move*** action is added for YARN MapReduce History Server
- Kerberos is implemented for ADH in ADCM
- The ability to move any service component to another Node or remove it from the cluster is added

Improvements

- The unnecessary repository/packages check at the HDFS installation step for ADH EE is removed
- The path for *docker-status-checker* files is changed

Bug fixes

- Fixed the error with the `refreshUserToGroupsMappings` task in the kerberized cluster
- Fixed the error with Solr not working after applying host actions
- Fixed the error with the ***Enable Resource Manager HA*** action in the kerberized environment
- Fixed the error with Spark expanding/shrinking in the kerberized environment
- Fixed the error with Solr shrinking in the kerberized environment
- Fixed the error with YARN Node Manager expanding in the kerberized environment
- Fixed the error `Keytab does not exist: $user.home/hadoop.keytab`
- Fixed the error with Flink Server expanding/shrinking
- Fixed the error with Flink JobManager Server Port 6123 availability
- Fixed the error with Sqoop expanding in the kerberized environment
- Fixed the error with YARN Server expanding/shrinking
- Fixed: the Solr role tried to import the absent monitoring role even with the monitoring service being not installed
- Fixed the error with running the ***Install*** service action for Solr
- Fixed the error with Spark Livy server expanding in the kerberized environment
- Fixed the error with Spark Thrift Server shutting down in the kerberized environment
- The Solr shared memory reservation error is fixed
- Fixed the error with Solr kerberization with no Hadoop services being added
- Fixed the error with starting jmxtrans after the host reboot
- The incorrect URL for the Hive Server Web UI is fixed
- Fixed the error with opening link to the HiveServer2 UI after ADH installation

- Fixed the error with availability of host actions after the cluster upgrade from 2.1.3.0 to 2.1.4.b2
- Fixed: the Enterprise cluster installation failing during the per-service action
- Fixed: the ***Reconfig and restart*** action failed for the Monitoring service with Airflow being installed
- Fixed: the Spark ThriftServer process did not stop after the Spark context being killed

Misc/Internal

- In order to speed up autotests and development process, the packages check is made optional for the specified environments
- The http and registry versions are bumped to the current ET release
- The specifications for new Spark and YARN MapReduce History Server actions are added

2.1.4.b2

Date: 20.07.2021

New features

- The ability to use external MySQL in AirFlow is added
- The ability to use external PostgreSQL in Airflow is added
- Host actions are added for the Spark3 service. Hosts actions here and below mean the actions managed at the host level
- Host actions are added for the Monitoring service
- Host actions are added for the Sqoop service
- Host actions are added for the Airflow service
- Host actions are added for the Solr service
- Host actions are added for the Flink service
- Host actions are added for the Zeppelin service
- Host actions are added for the Spark service
- Host actions are added for the MySQL service
- Host actions are added for the Hive service
- Host actions are added for the HBase service
- Host actions are added for the YARN service
- Host actions are added for the HDFS service

Improvements

- The Sqoop ***Check*** action is modified according to the new Hive external DB variables
- Host actions are renamed
- The unnecessary *solr-tools.jar* file is removed from the Solr submodule in the bundle, as it caused errors in CI

Bug fixes

- The error with offline installation in the operation system RH 7.9 is fixed
- The error with applying the ***Reconfig and restart*** action to the Monitoring service is fixed
- Fixed the error with installing MySQL on the host from which it was removed earlier
- Fixed the error with the `CPU utilization` YARN metric after the ADH cluster installation
- Fixed the error with the ***Maintenance DataNode*** action that occurred due to the incorrect content of the *dfs.hosts* file (if another DataNode has been switched to the maintenance state earlier)

Misc/Internal

- For debugging possible problems via Allure reports, logs collecting is implemented for Airflow service
- Fixed the wrong description in the autotest that implements migration to the external MySQL database
- Specifications for testing host actions are changed
- Tests for host actions are added
- Specifications and autotests are added for the ADH shrink scenarios

2.1.4.b1

Date: 22.06.2021

New features

- The ability to define custom HBase environment variables is added
- The action for removing MySQL from the ADH cluster is added
- The ability to use external PostgreSQL in Hive Metastore is added
- The ability to change the Hive Metastore `host:port` is added
- The ability to configure Java Heap for HiveServer2 is added
- The ability to add/change/remove configuration options from the *httpfs-site.xml* file via ADCM is added
- Start checks for JournalNodes and NameNodes are added
- Spark 3.1.1 is implemented for ADH 2.X
- The offline installation is implemented for ADH

Improvements

- The ***Check*** action is improved for Sqoop
- The built process for Solr is changed: Arenadata repositories are used instead of external Maven repositories
- The ability to use Docker Registry from Arenadata repository is implemented

- In order to install services (e.g. Airflow) without DNS, resolving host names in the Docker containers is implemented
- Airflow installation without DNS is implemented
- Implemented the DN check/wait for membership in the cluster from the DN itself
- The Spark component Spark History Server is made mandatory
- Refactoring of the ZooKeeper service is done

Bug fixes

- Fixed: Hive installation failed when adding the Tez component without TezUI
- Fixed: the duplicate key `type: list` is removed from the bundle configuration file
- Fixed: YARN applications could not run jobs after the Ranger plugin being enabled
- Fixed the error with the ***Enable Resource manager HA*** action in the ADH Community Edition
- Fixed the error with Sqoop installation after Ranger being installed
- Fixed problems with HBase logging

Misc/Internal

- Docker images in the package specifications are changed according to the new naming convention
- Packages for the 2.1.4 release are uploaded to the Google repository
- Fixed the logs collecting for HttpFS during autotests
- The repository for the 2.1.4 version of the product is created
- The ability to update a bundle without rebuilding packages is added
- Unnecessary garbage files are removed from the bundle build archive
- To resolve ansible `lint-issue`, the pipefail options for shell tasks are added
- Tests for checking the integration between ADH and Ranger are added
- Build 2.1.3.1 ADH

2.1.3

2.1.3.0

Date: 14.01.2021

New features

- The ***Remove/Add Hive Tez*** actions are added
- The ***Add diamond*** and ***Remove diamond*** actions are added
- Build Ranger 2.0.0

Improvements

- The logic of the YARN Resource Manager expanding is changed
- The validation logic for Spark Client is changed from `parallel` to `in series`
- ADPS integration: the new ADPS bundle that contains Ranger has to be re-integrated with ADH after moving Ranger from it

Bug fixes

- Fixed the error with closing Hive tasks after finishing checks
- Livy checks are temporarily disabled
- Fixed the error with bad cluster name that occurred when creating the HDFS service via Ranger Admin
- Fixed the error with the HDFS action ***Remove Client***
- Fixed unsuccessful Hive CLI checks after the Ranger plugin being enabled
- Fixed the error with connecting multiple clusters to one ADPS

Misc/Internal

- The repository for plugins is added to the release bundle
- Packages for the 2.1.3.0 release are uploaded to the Google repository
- Build 2.1.3.0 ADH
- ADH is bumped to 2.1.3.0
- New repositories for Ranger plugins are added
- Specifications on the workaround for the error with HBase expanding after HDFS expanding are edited
- Specifications on expanding ADH services are created

2.1.2

2.1.2.5

Date: 19.11.2020

New features

- Client components for Flink are added
- Client components for HDFS are added
- Client components for YARN are added

Improvements

- The timeout for the `docker_container` is increased in autotests. Also the checks of the correct starting order are added for services in containers
- The default port number for MySQL in the Airflow Metastore is changed

- The volume for Hadoop configurations (e.g. */etc/hadoop/conf/*, */etc/hive/conf*, etc.) in Docker images with Airflow is increased

Bug fixes

- Fixed the race condition within Sqoop checks (part 2 — with multiple Clients)
- Fixed the cluster installation error that occurred when MySQL being installed
- Fixed the cluster installation error that occurred when checking after installing Spark

Misc/Internal

- Packages for the 2.1.2.5 release are uploaded to the Google repository
- Building the offline package for ADH to the ADH repository is implemented
- ADH is bumped to 2.1.2.5
- Tests for the YARN/HDFS Client are created
- Specifications for autotests of the YARN/HDFS Client are added
- Changes in the Hadoop tests related to uploading bundles are made
- Autotests for Airflow are created
- All file accesses are made independent from the current working directory in autotests
- The dev repository for ADH 2.1.2.5 is initialized

2.1.2.4

Date: 18.09.2020

Bug fixes

- The incorrect repositories in the ADH release and ET packages are fixed

Misc/Internal

- The error with processing the `release_version` flag is fixed
- Packages are copied from 2.1.2.3 to 2.1.2.4

2.1.2.3

Date: 15.09.2020

New features

- The ADH bundle is divided into community and enterprise versions
- The High Availability for NameNodes is implemented

Bug fixes

- Fixed the error that occurred at the ***Restart NameNodes*** step during the ***Remove NameNode*** action
- Fixed the error with checking Hive Tez on multiple hosts
- Fixed the error with switching dynamic allocation
- Fixed: ZKFC ignored the `dfs.namenode.rpc-bind-host` parameter and used the `dfs.namenode.rpc-address` parameter for binding to the host address

Misc/Internal

- Packages for the 2.1.2.3 release are uploaded to the Google repository
- All specifications and BOMs related to ADH20 are moved to the *prj_adh*. Publishing of artifacts to the artifactory is changed
- The release and develop repositories are segregated in bundles

2.1.2.2

Date: 05.06.2020

Improvements

- The `epel-release` installation is disabled

Bug fixes

- The race condition within Sqoop checks is fixed
- Fixed the error with running the cluster ***Check*** action

Misc/Internal

- Packages for the 2.1.2.2 release are uploaded to the Google repository
- ADH is bumped to 2.1.2.2
- Nginx is copied from the Epel repository to the ADH2 repository

2.1.2.1

Date: 21.05.2020

New features

- Sqoop deployment is ported to ALT Linux
- Solr deployment is ported to ALT Linux
- Flink deployment is ported to ALT Linux
- The public ALT Linux repository for ZooKeeper 3.4.14 is created
- Airflow deployment is ported to ALT Linux

- The ability to set `nproc` limits for HBase is added
- Sqoop is added into the ADH bundle
- ADH 2.X packages are built for ALT Linux
- Solr 8.2.0 is added for ADH 2.2

Improvements

- Refactoring of the ADH deployment process for ALT Linux is made

Bug fixes

- The error with commissioning/decommissioning Nodes via ADCM is fixed
- Fixed the error `Port already in use: 10102` that occurred with HBase `hbck`
- Fixed the ordering of Generic components
- Fixed: web links in ADCM did not refresh after HDFS DataNodes or YARN Node Manager shrinking
- Fixed the error that occurred with YARN 3.1.2 in ALT Linux during ansible tasks
- Fixed the absence of `org.json.JSONObject` for Sqoop Metastore
- The */var/run/sqoop* directory is created for Sqoop Metastore
- The missing dependency for Flink-related packages is added
- Fixed the error with installing HBase and Solr when using the external ZooKeeper

Misc/Internal

- Airflow deployment is disabled (visible in ADCM and only for ALT Linux)
- The public repository for the release is changed
- Packages for the 2.1.2.1 release are uploaded to the Google repository
- Changes for libisal are merged
- Changes for bigtop-groovy, bigtop-jsvc, bigtop-tomcat, and bigtop-utils are merged
- Changes for Bigtop are merged
- Changes for Livy are merged
- Changes for ZooKeeper are merged
- Changes for Zeppelin are merged
- Changes for Spark are merged
- Changes for Phoenix are merged
- Changes for Tez are merged
- Changes for Hive are merged

- Changes for HBase are merged
- Changes for Hadoop are merged
- Bigtop branches for CentOS and ALT linux are manually merged
- The repository url is changed to 2.1.2
- Autotests for ADH services are reviewed according to the current stack version

2.1.2.0

Date: 19.02.2020

New features

- The ability to configure Hive ACID is added
- SELinux is disabled for all components during installation
- Support of the Flink 1.8.0 is implemented for ADCM
- Flink is added into the ADH bundle
- The logic of the ***Shrink*** action is improved
- GPU support is enabled for YARN
- Airflow is added into the ADH bundle
- The UI link is added for Solr at the main ADCM page
- The ***Shrink/Expand*** actions are implemented for HDFS HttpFS
- HDFS HttpFS checks are implemented
- The Solr Cloud Mode is implemented
- The Solr deployment is implemented
- Solr is added into the ADH bundle

Improvements

- Tez libraries are installed on Hive Client Nodes

Bug fixes

- Fixed: it was impossible to use Hive with Tez due to the configuration mismatch
- Fixed the error with saving configurations for HDFS and YARN
- Fixed the error with HBase checks after installation
- Fixed the error with YARN checks in the HA mode

Misc/Internal

Tests/example DAGs for checking the Airflow functionality are added

Tests for checking the Solr functionality are added

2.1.1

2.1.1.1

Date: 02.12.2019

Misc/Internal

Conversion to the custom systemd units

2.1.1.0

Date: 21.11.2019

New features

YARN Scheduler configuration is implemented

HDFS mover is implemented

The cluster-wide ***Install*** button is added to the ADCM UI

The ability to define the external ZooKeeper in the *core-site.xml* file is added

The ability to add custom/advanced configuration parameters to the **-site.xml* files is added

YARN Node labels are implemented

HDFS HttpFS is implemented

HDFS Short-Circuit Local Reads are implemented

HDFS Disk Balancer is implemented

HDFS Balancer is implemented

Improvements

The **-site.xml* files are unified

Asserts and fails are replaced with *adcm_check*

Monitoring is refactored: code/dashboards are unified, metrics are redesigned, etc.

The hostname variable is removed from the Zeppelin PID definition

The HDFS dashboard is divided into HDFS and YARN dashboards in Grafana

Hadoop PID file names are changed

Misc/Internal

Manual testing of the ADH 2.1 installation is performed according to the documentation

2.1.0

2.1.0.0

Date: 10.10.2019

New features

Implemented the ability to get a status for the following services:

- Zeppelin
- Livy Server
- Thrift Server
- Spark Server
- Zeppelin
- Phoenix
- HBase Thrift
- HBase
- MySQL
- YARN
- HDFS
- Hive

Implemented service management for the following services:

- Livy Server
- Zeppelin
- Spark Thrift Server
- Spark Server
- Phoenix Server
- HBase Thrift
- HBase Region Server
- HBase Master
- Node Manager
- Resource Manager
- Timeline Service
- WebHCat
- MySQL
- Hive Metastore
- Hive Server
- DataNodes
- Secondary NameNodes
- NameNodes

Prepared deployment scripts for the following services:

- Livy Server
- Spark Thrift
- Spark Server
- MySQL

- HBase Thrift
- Phoenix Server
- Hive
- HDFS

Implemented service checks for the following services:

- Zeppelin
- Spark Thrift Server
- Spark Server
- Livy Server
- Hive
- MySQL
- YARN
- HDFS
- ZooKeeper

Implemented the deployment of the following services:

- Zeppelin
- Livy Server
- Spark Server
- HBase Thrift
- Phoenix Server
- HBase Region Server
- HBase Master
- WebHCat
- Hive Metastore
- MySQL
- Hive Server
- Hive Client
- Node Manager
- Resource Manager
- Timeline Server
- DataNode
- Secondary NameNode
- NameNode

The following builds are available:

- Tez 0.9.2
- Livy 0.6
- Pig 0.17
- Flume 1.8
- YARN UI 2.0
- Sqoop 1.4.7

Monitoring features are implemented for the following services:

- Spark
- YARN

  - HBase
  - HDFS
  - Hive

- Necessary configurations for Hive/Tez are added
- The `hbase-site` block is added to the *config.yml* file
- Necessary configurations for Hadoop services are added
- Quick links for services are added
- The HDFS rack awareness is implemented via custom scripts
- The YARN and MapReduce services are combined into single one
- The Resource Manager High Availability is implemented
- Checks for Decommission/Recommission for Node Managers are implemented
- Checks for Decommission/Recommission for DataNodes are implemented
- Zeppelin is bumped to 0.8.1
- Zeppelin is implemented for ADCM
- Tez UI is implemented for ADCM
- The ability to add a new Node Manager to ADH is added
- The ability to add new DataNodes to ADH clusters is added
- Spark is implemented for ADCM
- Ranger is bumped to 1.1
- The ZooKeeper Quorum configuration is added

Improvements

- The MySQL role is added to the ADH bundle (as a service)
- Multiple configuration directories for Nodes are implemented
- The YARN logs aggregation is enabled
- Spark and Hive roles are reviewed
- The Hadoop role is divided into HDFS, YARN, MapReduce
- The Hadoop role for ADCM is refactored
- Separate roles for Hadoop are implemented
- The ZooKeeper service role is ported from the ADS Bundle
- Basic YARN service features are refactored

Bug fixes

Fixed the error with the `hbase.zookeeper.quorum` parameter missing after the installation

Misc/Internal

- Pre-release preparations are made for ADH 2.1.0
- The *EULA.txt* file is added to the bundle root
- The repository for ZooKeeper packages is added to ADH
- All ADH bundle submodules are switched to Master
- Documentation on Decommission/Recomission/HA is prepared
- Documentation on HBase deployment via ADCM is prepared
- Documentation on Spark deployment via ADCM is prepared
- Documentation on Hive deployment via ADCM is prepared
- Documentation on YARN deployment via ADCM is prepared
- Documentation on HDFS deployment via ADCM is prepared
- Documentation for the ADH bundle is prepared
- Spark autotests are implemented
- Hive autotests are implemented
- YARN autotests are implemented
- HDFS autotests are implemented
- Smoke tests for the Livy Server service check are prepared
- Smoke tests for the Spark Thrift Server service check are prepared
- Smoke tests for the Spark Server service check are prepared
- Smoke tests for the MySQL service check are prepared
- Smoke tests for the HBase service check are prepared
- Smoke tests for the Phoenix service check are prepared
- Smoke tests for the Hive service check are prepared
- Smoke tests for the HDFS service check are prepared
- The latest stable packages for ADH are built

Supported services

Konstantin Alpashkin, Daria Barysheva, Sergei Tikhomirov

To Table of Contents

This article lists services included into the current version of ADH. The ***Version*** column contains references to the release notes of each component.

Service	Component	Version
Apache Airflow2	Broker (Redis)	6.2.7
	Scheduler	2.6.3
	Server	2.6.3
	Worker	2.6.3
Apache Flink	Client	1.20.1_arenadata1
	Job Manager	1.20.1_arenadata1
	Task Manager	1.20.1_arenadata1
	SQL Gateway	1.20.1_arenadata1
Apache HBase	Client	2.6.3_arenadata1
	Master server	2.6.3_arenadata1
	Region server	2.6.3_arenadata1
	Thrift2 server	2.6.3_arenadata1
	Phoenix query server	6.0.0_arenadata4
	REST server	2.6.3_arenadata1
Apache HDFS	Client	3.3.6_arenadata1
	DataNode	3.3.6_arenadata1
	NameNode	3.3.6_arenadata1
	JournalNode	3.3.6_arenadata1
	ZKFC	3.3.6_arenadata1
	HttpFS	3.3.6_arenadata1
Apache Hive	Client	4.0.1_arenadata2
	HiveServer2	4.0.1_arenadata2
	Metastore	4.0.1_arenadata2
	Tez	0.10.4_arenadata1
	TezUI	0.10.4_arenadata1
HUE	Server	4.11.0_arenadata4
Apache Impala	Daemon	4.5.0_arenadata1
	Statestore	4.5.0_arenadata1
	Catalog Service	4.5.0_arenadata1
	Client	4.5.0_arenadata1
Apache Kyuubi	Kyuubi Server	1.10.1_arenadata2
Apache Ozone	Ozone Datanode	2.0.0_arenadata1
	Ozone Manager	2.0.0_arenadata1
	Ozone Recon	2.0.0_arenadata1
	Ozone S3G	2.0.0_arenadata1
	Ozone Storage Container Manager	2.0.0_arenadata1
Apache Solr	Server	8.11.4_arenadata1
Spark3	Client	3.5.4_arenadata2
	History server	3.5.4_arenadata2
	Livy	0.8.0_arenadata1
SSM	Smart Server	2.1.1
	Smart Agent	2.1.1
Trino	Trino Coordinator	476_arenadata1
	Trino Worker	476_arenadata1
Apache YARN	Client	3.3.6
	MapReduce History server	3.3.6
	NodeManager	3.3.6
	ResourceManager	3.3.6
	Timeline server v.1	3.3.6
Apache Zeppelin	Server	0.11.2_arenadata2
Apache ZooKeeper	Server	3.8.4
Arenadata Monitoring	Monitoring	3.2
Arenadata Postgres	Arenadata Postgres	16

Version compatibility matrix

Sergei Tikhomirov

To Table of Contents

ADH version	ADPS min version	ADPS max version	ADCM min version
4.1.0	2.0.0_b1	—	2.7.0
4.0.0	1.2.2	—	2.5.0
3.3.6.2	1.2.1_b1	—	2.4.0
3.3.6.1	1.2.0_b1	—	2.2.0
3.2.4.3	1.1.0_b1	1.1.2_b2	2.0.0
3.2.4.2	1.1.0_b1	1.1.2_b2	2.0.0
3.2.4.1	1.1.0_b1	1.1.2_b2	2023.10.10.08
3.1.2.1	1.0.5_b1	1.0.5_b2	2023.10.10.08
2.1.10	1.0.4_b1	1.0.4_b6	2023.04.05.14
2.1.8	1.0.4_b1	1.0.4_b6	2022.08.08.14
2.1.7	1.0.4_b1	1.0.4_b6	2022.08.08.14

ADH known issues

Sergei Tikhomirov

To Table of Contents

Service/component	Issue	From	Fixed	Workaround
Ozone	There is no option to enable Kerberos authentication after installation	3.3.6.2	—	You must first install a cluster without a service, enable SSL and Kerberos, and then install and configure the Ozone service
Ozone	Clients do not work when secure intercomponent communication is enabled. Do not enable `hdds.grpc.tls.enabled`	3.3.6.2	—	—
Trino, Smart Storage Manager	The Trino service does not currently support the SSM client. This means that when ***Enable SmartFileSystem for Hadoop*** (an SSM configuration parameter) is enabled, Trino will not work	3.3.6.2	—	Do not enable SmartFileSystem for Hadoop if Trino is used
Flink and YARN HA	If YARN high availability is enabled (YARN has multiple ResourceManager components), certain configuration parameters have to be specified manually to allow proper interaction between Flink and highly available YARN	4.0.0	4.1.0	As a temporary workaround, the following parameters must be specified in ADCM Flink settings (***Clusters*** → ***<ADH_cluster>*** → ***Services*** → ***Flink*** → ***Primary configuration*** → ***Custom flink-conf.yaml***):

```
flink.yarn.resourcemanager.ha.enabled: True ①
flink.yarn.resourcemanager.ha.rm-ids: <adh_rm-1>,<adh_rm-2>,<adh_rm-3> ②
flink.yarn.resourcemanager.hostname.adh_rm-1: <adh_rm_fqdn-1> ③
flink.yarn.resourcemanager.hostname.adh_rm-2: <adh_rm_fqdn-2>
flink.yarn.resourcemanager.hostname.adh_rm-3: <adh_rm_fqdn-3>
```

① Notifies Flink that YARN HA is enabled.

② Lists ResourceManager IDs available in the YARN cluster.

③ Specifies FQDNs of each ResourceManager component.

The up-to-date value for `flink.yarn.resourcemanager.ha.rm-ids` can be found in */etc/hadoop/conf/yarn-site.xml*, for example:

```
$ grep "yarn.resourcemanager.ha.rm-ids" /etc/hadoop/conf/yarn-site.xml -A1
```

Sample output:

```
<name>yarn.resourcemanager.ha.rm-ids</name>
<value>rm_ka-adh-2,rm_ka-adh-3</value>
```

Service/component	Issue	From	Fixed	Workaround
Impala	The problem is the integration of Impala with external object storage. During configuration and table creation in an S3 location, the following error occurs: `Error reading from HDFS file: s3://…`	3.3.6.2	4.1.0	Delete or transfer the following files: */usr/lib/impala/lib/aws-java-sdk-core-1.12.765.jar*, */usr/lib/impala/lib/aws-java-sdk-logs-1.12.765.jar*
Trino, Ozone	Trino does not work with Ozone using the `osf`, `o3sf` protocols. Use the `s3` protocol (s3 gateway). Must be specified in the location of objects	3.3.6.2	4.0.0	`ALTER … SET LOCATION 's3://…'`
Impala	When transferring the Impala client to another host, all the necessary files are not created	3.3.6.1	4.0.0	Restart the service after migrating components
Smart Storage Manager	When you enable SSL on a cluster using the Manage SSL cluster action, the service will continue to work over HTTP	3.3.6.1	4.0.0	Manual configuration is required according to the instructions