Arenadata DB

Arenadata DB (ADB) is a distributed analytical data storage built on the open-source massively parallel system Greengage DB and designed to store and process large amounts of structured and semi-structured data (up to tens of petabytes).

Version CURRENT Language: EN

Contents

1. Concepts
    - ADB architecture
    - Data model
2. Planning guide
    - Hardware requirements
    - Network requirements
    - Software requirements
3. Get started
    - Installation
        - Online installation
            - Install ADCM
            - Prepare hosts
            - Install ADB cluster
                - Create a cluster
                - Add services
                - Add hosts to a cluster
                - Add components
                - Configure services
                - Configure a cluster
                - Install a 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 ADB cluster
                - Create a cluster
                - Add services
                - Add hosts to a cluster
                - Add components
                - Configure services
                - Configure a cluster
                - Import ET settings
                - Install a cluster
4. Access management
    - Authentication
        - LDAP
        - PAM with LDAP
        - Kerberos
            - MIT Kerberos
    - Accounts in ADB
5. Database maintenance
    - Cluster management via ADCM
        - Cluster actions
        - Service actions
            - ADB
            - ADB Control
            - ADBM
            - PXF
            - Connectors
            - Chrony
    - Manage DBMS via GUCs
    - Backup and restore
        - Use ADB DDBoost plugin
    - Data optimization
        - Work with arenadata_toolkit
        - Manage spill files
        - Delete orphaned files
6. Data management
    - Overview of database objects
        - Databases
        - Schemas
        - Tables
            - Tables overview
            - Table types
            - Data compression
            - Distribution
            - Partitioning
        - Tablespaces
7. Integration with external systems
    - Arenadata DB Control
        - Overview
        - Configure external databases
        - Connect to ADB Control
        - Access management
            - Authentication
                - Basic
                - LDAP
            - Authorization
            - User sessions
            - SSL encryption
        - Configuration
            - Job policies
            - Cluster management
            - Security policies
            - General settings
        - Dashboard
        - Monitoring
            - Commands
            - Transactions
            - Sessions
            - Work with query tags
        - Audit
        - System control
            - View jobs
            - View information on components
    - Arenadata DB Backup Manager
        - Overview
        - Configure an external database
        - Connect to ADBM
        - Manage configurations
        - Work with backups
            - Run actions
            - View backups
            - View restores
        - Disaster recovery
        - View actions
        - Work with ADBM CLI
    - Use connectors
        - ADB ClickHouse Connector
            - Overview
            - Installation
            - Configuration
            - Usage examples
        - ADB and Kafka
            - ADB to Kafka Connector
                - Installation
                - Configuration
                - Usage examples
            - Kafka to ADB Connector
                - Installation
                - Configuration
                - Usage examples
        - ADB to ADB Connector
            - Overview
            - Installation
            - Configuration
            - Usage examples
8. References
    - Configuration parameters
    - Supported extensions
    - SQL commands not supported in ADB
9. Release notes
    - ADB
        - ADB 7
            - Releases
            - Known issues
        - ADB 6
            - Releases
            - Known issues
        - ADB 5
    - ADB Control
        - Releases
        - Known issues
    - ADBM
        - Releases
        - Known issues

ADB architecture

Anton Vasilev

Contents

To Table of Contents

- Coordinator
- Standby coordinator
- Segments
  - Primary segments
  - Mirror segments
  - Group mirroring
  - Spread mirroring
- Interconnect
- Use ADB for ETL
- Integration with other systems
- Advanced cluster management tools

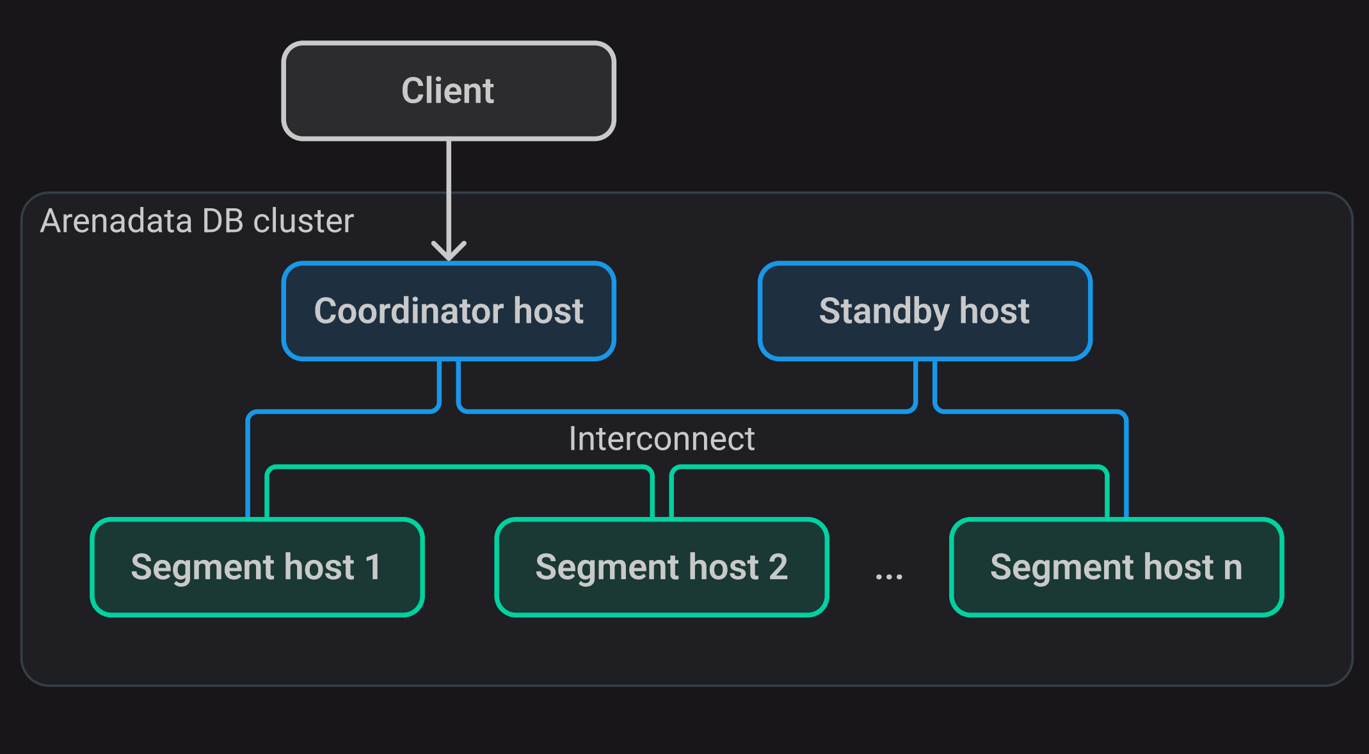
Arenadata DB (ADB) is a Massively Parallel Processing (MPP) database based on Greengage DB.

The MPP architecture consists of a cluster of simultaneously operating nodes. ADB implements this as a cluster of separate PostgreSQL instances.

ADB uses the shared-nothing approach where each segment has its own portion of data. Multiple segments can run on a single host, but they process data independently. The system primarily scales horizontally by adding more physical nodes to the cluster.

The ADB architecture consists of three core components:

- Coordinator
- Standby coordinator
- Segments


Client
Arenadata DB cluster
Coordinator host
Standby host
Interconnect
Segment host 1
Segment host 2
...
Segment host n

ADB architecture

Coordinator

Coordinator (formerly master) is the entry point to the database. End users connect to the coordinator to submit queries.

The coordinator functions include:

- Client authentication. The coordinator controls client access to the system using the standard *pg_hba.conf* file. Segments accept SQL client connections from the coordinator only.
- Query parsing. The coordinator checks the original SQL statement for correct syntax. ADB follows PostgreSQL syntax with some adjustments to support its distributed design.
- Optimization. The optimizer determines the most efficient way to execute an SQL query by evaluating multiple potential execution plans and selecting the one with the lowest estimated cost.
- Workload distribution. The coordinator distributes workload across query executors on all segments.
- Getting results. The coordinator aggregates local results generated by segments (such as filtered rows, aggregated values, or joined data) and returns the final result to the client.
- Administrative operations. Beyond processing queries, the coordinator performs database management. For example, it updates system catalogs after receiving a DDL command (such as `CREATE TABLE`).

The coordinator doesn't store user data; it resides on segments only. The coordinator stores the global system catalog (tables like `pg_database` and `pg_authid`) with metadata about the database itself and about structure and properties of user data. When a user creates a new table, the coordinator stores information about its schema, ownership, and distribution type while the actual content of the table is stored on the segment.

Standby coordinator

The **standby coordinator** serves as a warm standby: it is kept up to date and ready to take over, but it is not processing queries unless activated. The standby coordinator is optional but highly recommended.

If the primary coordinator fails, there is no automatic switching to the standby coordinator. Instead, ADB relies on administrator intervention. For this, you can use the Activate Standby cluster action in the web interface of Arenadata Cluster Manager (ADCM).

Data integrity and recovery is implemented using **write-ahead logging (WAL)** inherited from PostgreSQL. WAL ensures that all changes to data files are first recorded in the WAL log and flushed to permanent storage before the actual data pages are modified on disk. This guarantees that even if a crash occurs, the database can be recovered by replaying the WAL records.

Segments

Each **segment** is an independent PostgreSQL database that stores and processes its set of data. ADB distributes user data across segments based on the distribution key that you select when you create a table. Balanced distribution of data and workloads across segments is the key to obtaining the best performance from ADB.

> TIP
>
> It is advised that all segment hosts have identical hardware resources and configurations. This is because the coordinator aggregates results only after all segments complete their work, so the ADB performance will be as high as the slowest segment server in the cluster. If one segment is significantly slower than the others, it will delay the entire query.

Primary segments

Primary segments are active nodes that store data, receive query plans, and execute queries. They run local database operations on their sets of data and send the results back to the coordinator.

Mirror segments

Mirror segments provide high availability support for primary segments. If a primary segment becomes unavailable, the system automatically fails over to the mirror copy. The mirror then starts acting as a primary segment, and the unavailable primary segment becomes a mirror segment. The administrator must recover the failed segment, and then exchange the primary and mirror segments so they are in their initial (**preferred**) roles.

A mirror segment must always reside on a different host than its primary segment. Two mirror configurations exist:

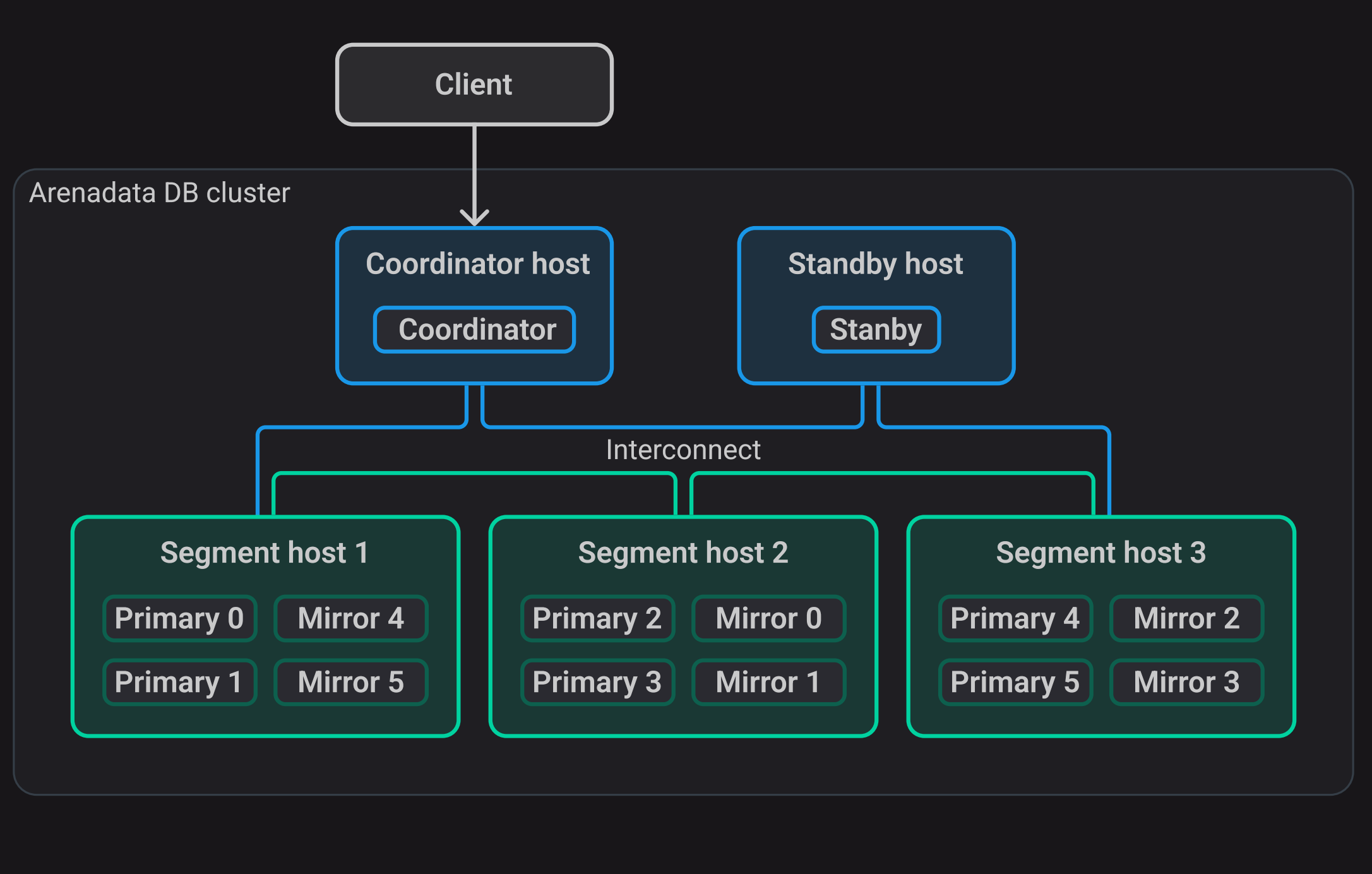
- Group mirroring
- Spread mirroring

Group mirroring

With **group mirroring**, all mirror segments for primary segments of a host are put on one backup host. For example, in the image below, the `Primary 0` and `Primary 1` segments reside on `Segment host 1`, and their mirrors `Mirror 0` and `Mirror 1` reside on `Segment host 2`.

Key characteristics of group mirroring:

- It is the least expensive option because it can be achieved with two hosts only.
- It leads to unbalanced load in case of failure. When a host fails, its mirror host will get double the workload: in addition to its own primary segments (`Primary 2` and `Primary 3` on the image below), its mirrors (`Mirror 0` and `Mirror 1`) will act as primary segments. The host performance will degrade by 50% or more.


Client
Arenadata DB cluster
Coordinator host
Coordinator
Standby host
Stanby
Interconnect
Segment host 1
Primary 0
Mirror 4
Primary 1
Mirror 5
Segment host 2
Primary 2
Mirror 0
Primary 3
Mirror 1
Segment host 3
Primary 4
Mirror 2
Primary 5
Mirror 3

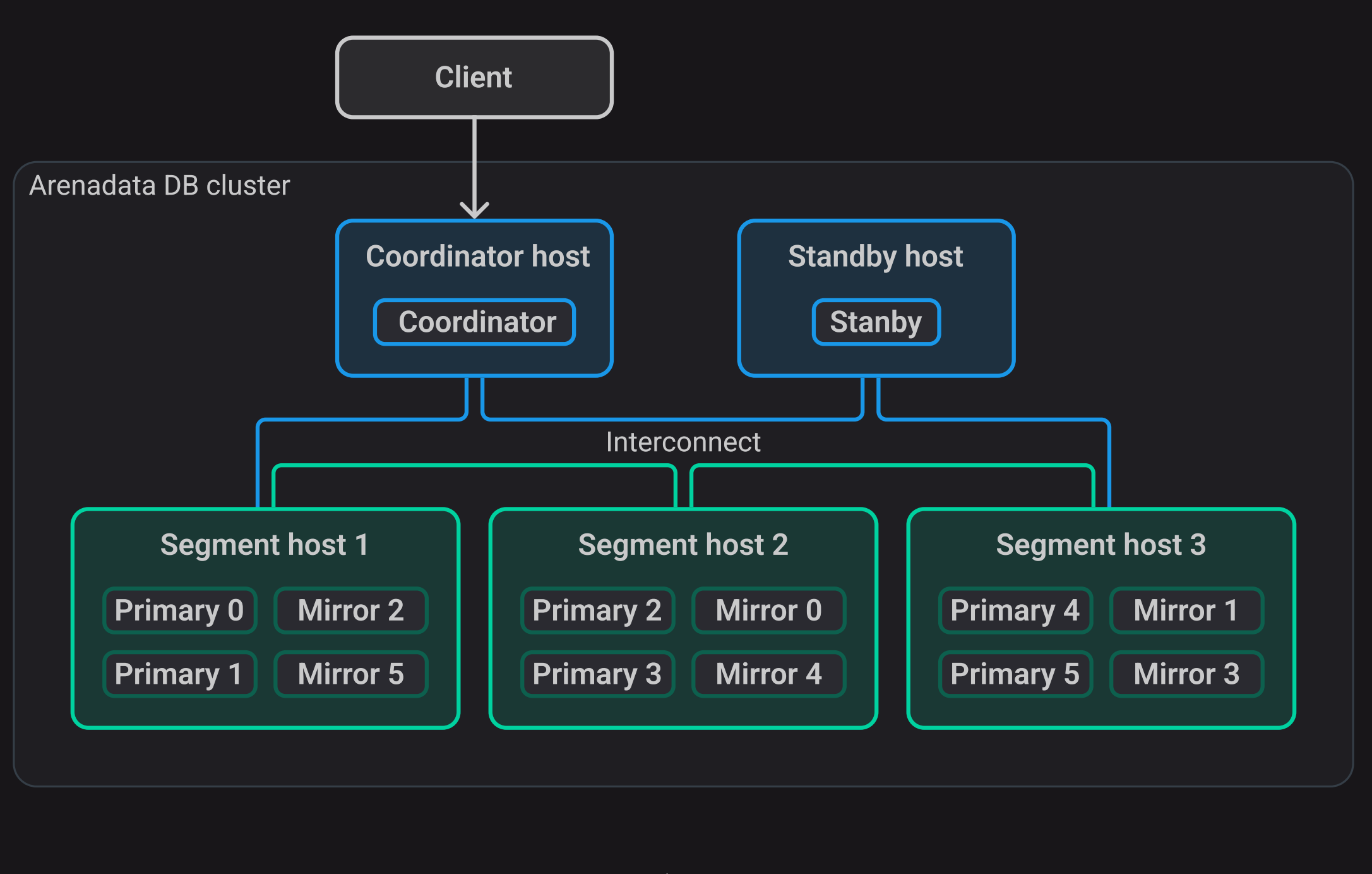
group mirroring

Spread mirroring

With **spread mirroring**, mirrors for each host's primary segments are spread across as many hosts as there are segments per host. For example, in the image below, the `Primary 0` and `Primary 1` segments reside on `Segment host 1`, and their mirrors reside on different hosts: `Mirror 0` on `Segment host 2`, and `Mirror 1` on `Segment host 3`.

Comparing to group mirroring:

- Better fault tolerance. Host failures affect only one mirror per host, minimizing workload disruption. On the image below, if `Segment host 1` fails, `Segment host 2` will have `Mirror 0` activated, and `Segment host 3` will have `Mirror 1` activated. The load on each host will increase by 50%, and this extra load is evenly distributed among hosts.
- More expensive. This configuration requires the cluster to have at least one more host than there are segments per host.


Client
Arenadata DB cluster
Coordinator host
Coordinator
Standby host
Stanby
Interconnect
Segment host 1
Primary 0
Mirror 2
Primary 1
Mirror 5
Segment host 2
Primary 2
Mirror 0
Primary 3
Mirror 4
Segment host 3
Primary 4
Mirror 1
Primary 5
Mirror 3

Spread mirroring

Interconnect

The **interconnect** is a high-speed network layer that enables communication between segments during query execution. A 10-Gigabit system, or faster, is recommended.

Most database operations run across all segments in parallel and require data motion (data moving between segments). The interconnect is used for data motion, and slow interconnect can lead to delays in such operations.

The interconnect is not required for queries executed on a single segment (targeted query), particularly those filtered by the distribution key where all necessary data can be found on the segment.

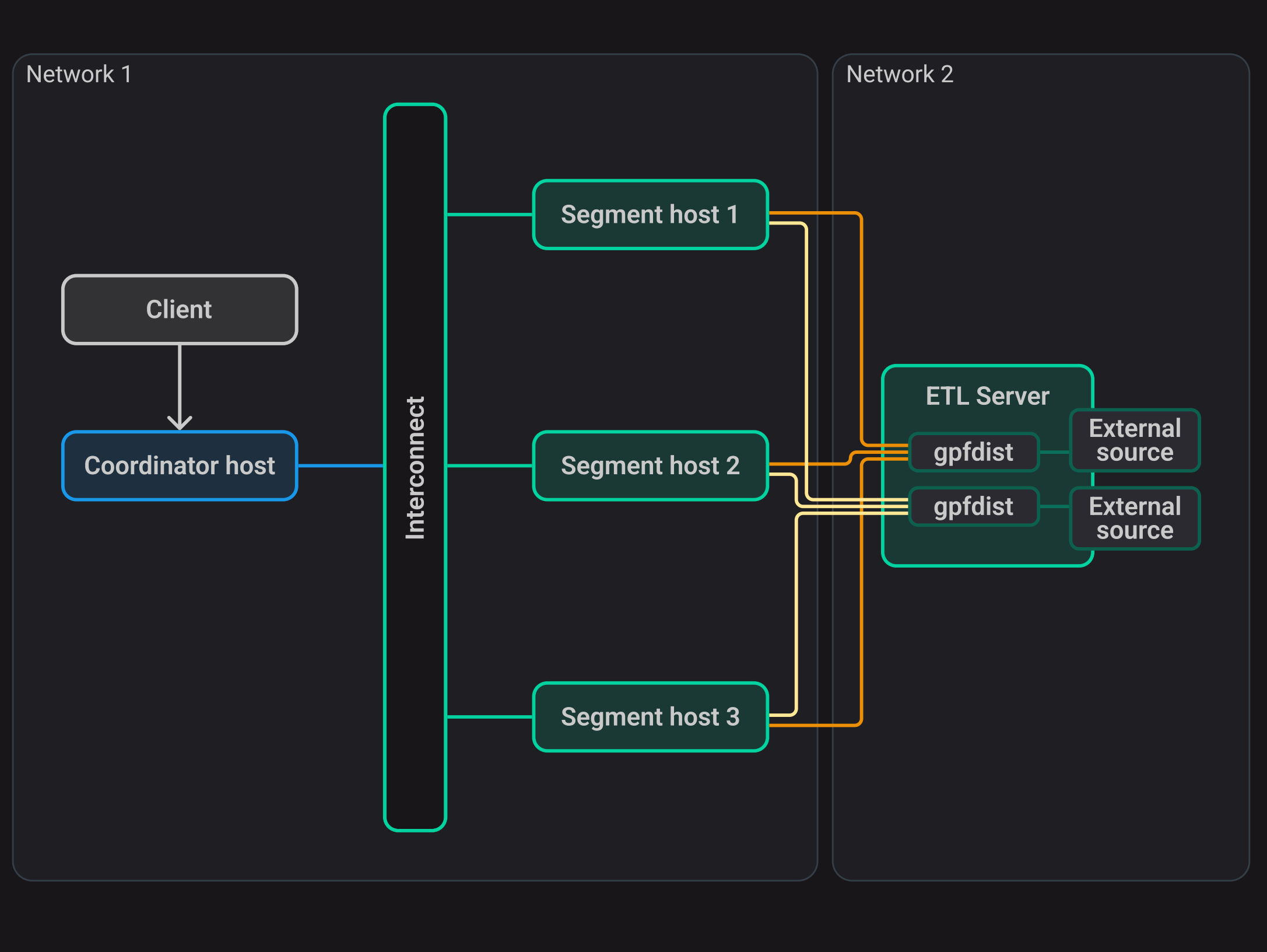
By default, the interconnect uses User Datagram Protocol with flow control (UDPIFC). You can select the interconnect type (`udpifc`, `tcp`, or `proxy`) in the ADB configuration parameters.

Use ADB for ETL

Production systems may deploy designated servers for the extract, transform, and load (ETL) purposes outside the ADB cluster. These external servers can connect to the ADB cluster via standard protocols like ODBC, JDBC, or libpq.

To transfer large volumes of data, you can use the File Distribution Protocol utility (`gpfdist`). You can run it directly on ETL servers where external files reside. With this architecture, ETL operations don't compete with query resources.

To load data using `gpfdist`, define external tables with `gpfdist` as the data source. When you select data using this external table, segments directly connect to `gpfdist` and load data in parallel. The coordinator host only coordinates the process.


Network 1
Network 2
Client
Coordinator host
Interconnect
Segment host 1
Segment host 2
Segment host 3
ETL Server
gpfdist
gpfdist
External source
External source

ETL hosts

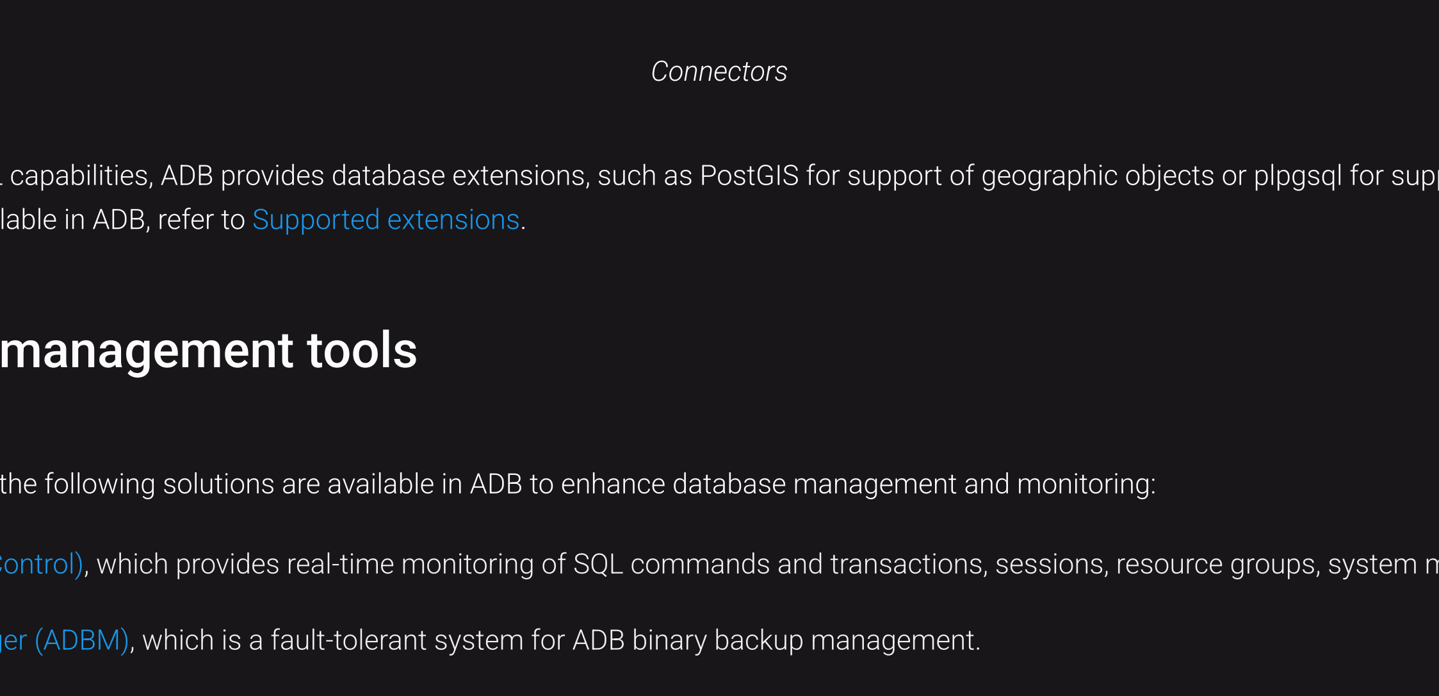
Integration with other systems

For integration with diverse data sources and various data formats, ADB uses Platform Extension Framework (PXF) and specialized connectors.

PXF is a data access framework that allows you to query external systems directly. It enables federated queries: you can access and combine data from different sources with a single query. It also supports filter pushdown, which lets external data sources filter data for the `WHERE` clause on their side and thus reduces the amount of data that is transferred to ADB. PXF allows you to handle multiple data formats including Avro, OCF, CSV, Text, ORC, and JSON.

On top of PXF, connectors are built to enable data transfer between the cluster and particular external systems (such as file storage, databases, and streams).

Built-in PXF connectors enable integration with Oracle Database, S3, HBase, Hadoop, HDFS, Hive, and JDBC sources. The Enterprise Edition of ADB provides the following additional connectors: ADB ClickHouse Connector, ADB to Kafka Connector and Kafka to ADB Connector, ADB Spark 3 Connector, and ADB to ADB Connector.


Arenadata DB cluster
ClickHouse cluster
External data
Client
Coordinator host
Interconnect
Segment host 1
PXF server
Segment host 2
PXF server
ClickHouse node 1
ClickHouse node 2
Kafka cluster
Kafka broker 1
Kafka broker 2
Data format: MergeTree and Distributed engine families
Data format: AVRO OCF, CSV, Text, ORC, JSON
Data format: AVRO OCF, CSV, Text

Connectors

Additionally, to extend native SQL capabilities, ADB provides database extensions, such as PostGIS for support of geographic objects or plpgsql for support of the PL/pgSQL language. For the full list of extensions available in ADB, refer to Supported extensions.

Advanced cluster management tools

As part of the Enterprise Edition, the following solutions are available in ADB to enhance database management and monitoring:

- Arenadata DB Control (ADB Control), which provides real-time monitoring of SQL commands and transactions, sessions, resource groups, system metrics, and more.
- Arenadata DB Backup Manager (ADBM), which is a fault-tolerant system for ADB binary backup management.

Data model

Daria Barysheva

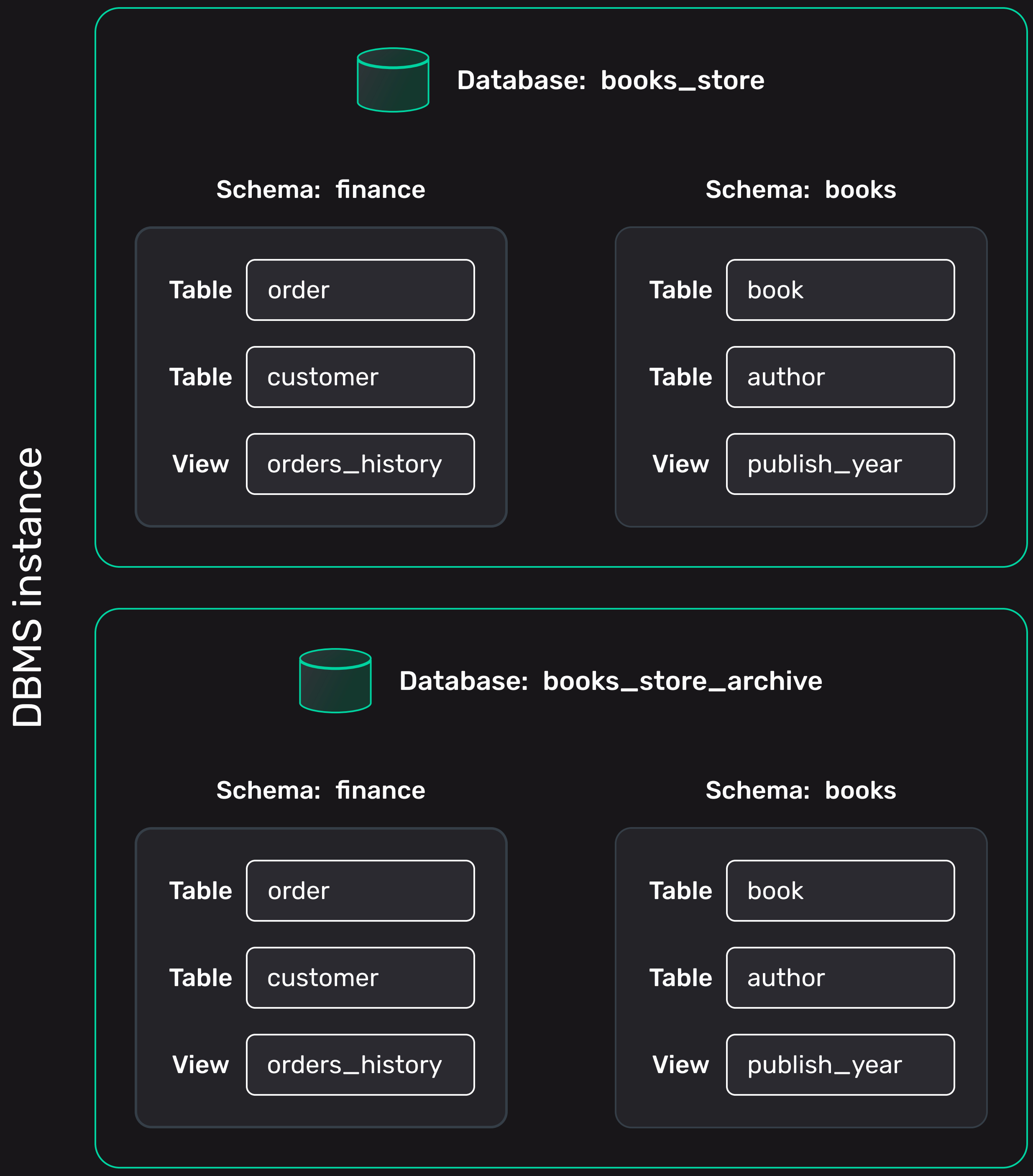
To Table of Contents

ADB is built on the open-source massively parallel Greengage DB DBMS that in turn is based on the PostgreSQL object-relational database. That is why most of the data model concepts that make up the core of PostgreSQL can be also applied to ADB.

The main data model components used in ADB are listed below:

- Database. This is a collection of data physically stored together. Every ADB cluster can contain one or more named databases. This is different from some other database management systems (e.g. Oracle) where the DBMS instance equals to the database. However, client programs can connect to and access only one ADB database at a time — cross-queries between different databases are impossible since databases are rigidly separated. Data transfer between several databases is also impossible by the DBMS functionality.
- Schema. This is a logical union of entities within a single database. Unlike databases, schemas are not physically separated: users can access multiple schemas at a time (if they have the necessary privileges).
- **Object**. This is any other entity that is defined within the database schema. For example, table, function, view, or materialized view, etc. To access objects, you can use the ANSI standard: `database.schema.object`. But the `schema.object` form is preferable (as you can access only the database defined in the current connection anyway). Notice that the same object name can be used in different schemas without conflict.

The picture below illustrates the hierarchy of the ADB data model components on the basis of a simple example. It includes the `books_store` and `books_store_archive` databases with two custom schemas, `finance` and `books`, inside each of them. Every schema in turn contains some tables and views. In this case, users can access both schemas at a time while working with any of the two databases specified in the connection request. But users cannot cross-query between databases.


DBMS instance
Database: books_store
Schema: finance
Table order
Table customer
View orders_history
Schema: books
Table book
Table author
View publish_year
Database: books_store_archive
Schema: finance
Table order
Table customer
View orders_history
Schema: books
Table book
Table author
View publish_year

Data model example

Hardware requirements

Daria Barysheva

Contents

To Table of Contents

ADB

ADB Control

ADBM

Arenadata Enterprise Tools

IMPORTANT

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

ADB

Coordinator host

- Physical server.
- **CPU:** 8 cores or more.
- **RAM:** 16 GB or more.
- **Disk storage:** separate physical unformatted RAID 10 array (5 GB or more).

Segment hosts

- Physical server.
- **CPU:** 16 cores or more.
- **RAM:** 32 GB or more.
- **Disk storage:** one or more physical unformatted RAID 10 arrays (10 GB or more per array).

ADB Control

NOTE

For more information on the ADB Control architecture, see Arenadata DB Control overview.

ADB Control host

The following are resources that are required on the host where ADB Control is installed.

Service	CPU cores	RAM	Disk storage	Note
Backend Server	6	16 GB	—	Requires additional monitoring of inner metrics queues
Scheduler	2	4 GB	—	Resource consumption may depend on the job configuration (like time periods, batch sizes, scheduling of heavy jobs to run at the same time)
Metrics DB (ClickHouse)	4	16 GB	100 GB	It is better to have the RAM amount that is equal to the compressed data in ClickHouse
Query DB (PostgreSQL)	4	8 GB	200 GB	Storage should be managed carefully so that it won't be exhausted between data cleanup events in ADB Control
Other components: Web UI & UI Backend Server, Service Registry	4	20 GB	10 GB	—
Total	20	64 GB	310 GB	—

ADB hosts

The following are resources that are required on each host of the ADB cluster where an ADB Control agent is installed:

- **CPU:** 4 cores or more.
- **RAM:** 4 GB or more.

ADBM

NOTE

For more information on the ADBM architecture, see Arenadata DB Backup Manager overview.

ADBM host

Arenadata DB Backup Manager can be installed on the same host as ADB Control or on a separate host. ADBM host requires the following minimal resources (for etcd, Backup Manager, Service Registry):

- **CPU:** 4 cores or more.
- **RAM:** 12 GB or more.
- **Disk storage:** 10 GB or more.

ADB hosts

On segment hosts, 30% CPU and 2 GB RAM are utilized by ADBM agent and pgbackrest instance for a one-threaded backup process.

ADBM agents on segment hosts require not less than 1 Gbps network to work. However, the recommended network performance is 10—25 Gbps and higher (the value depends on backup requirements).

Arenadata Enterprise Tools

If you use ET along with your product, take into account the following requirements:

- Virtual server.
- CPU: 4 cores or more.
- RAM: 8 GB or more.
- Disk: 50 GB or more.

Network requirements

Daria Barysheva

Contents

To Table of Contents

- Overview
- Ports
  - ADB ports
  - ADB Control ports
  - ADBM ports
  - Arenadata Enterprise Tools ports

Overview

The network requirements for Arenadata DB installation are listed below:

- One or more dedicated and separate peer-to-peer interconnect networks are required for internal cluster communication.
- All cluster hosts should be connected to interconnect networks.
- All cluster hosts should have IP addresses and the ability to communicate with each other at required speeds.
- `ping` access to any cluster host (ICMP) should be provided.
- The jumbo frame `MTU = 9000` should be specified in interconnect networks to allow packets formed by ADB (`gp_max_packet_size` + overhead) to fit into these frames.
- The Ethernet speed 1 Gbps or more should be provided (the standard value is 10 Gbps).
- There should be no firewall or other software that blocks or slows down the traffic (`firewalld`, `iptables`, `ufw`, etc.). If firewall software is not deactivated, you should configure your software to allow required communication between ADB hosts. For more information, see Greenplum documentation.
- From the outside, the ADB cluster should be available on the following ports:
  - `22` — SSH;
  - `5432` — incoming PostgreSQL interface.
- To access NTP (Network Time Protocol) servers, the coordinator host should be allowed to connect to it (by default, via the `123/udp` port).
- To send cluster component statuses to Arenadata Cluster Manager (ADCM), ADB hosts should be allowed to connect to it (by default, via the `8000/tcp` port). See the Install ADCM article for details on how to install ADCM.
- For offline installation of ADB, access to Arenadata Enterprise Tools repositories deployed on a dedicated host is required (by default, via the `81/tcp` and `443/tcp` ports). The Install Enterprise Tools cluster section contains instructions on how to install Arenadata Enterprise Tools.
- All ports that are listed below should be open on the respective hosts. If you use firewall, add ports to the exception list.

Ports

ADB ports

Coordinator/standby

Service	Port	Protocol	Note
ADB	5432	TCP	SQL client connection port on the ADB coordinator host. Supports clients using the PostgreSQL libpq API. Configurable
Standby coordinator replicator	1025-65535	TCP	Ports used to replicate the coordinator log to the standby coordinator

All cluster hosts

Service	Port	Protocol	Note
Secure shell (SSH): `gpssh`, `gpssh-exkeys`, `gppkg`	22	SSH	Many Greenplum utilities use Secure Copy Protocol (SCP) and Secure Shell (SSH) to manage a cluster and transfer files between hosts
Interconnect	10000-65535	UDP	The interconnect transports database tuples between ADB segments during query execution (for exact ports refer to the `gp_segment_configuration` system table)
Gpperfmon agents	8888	TCP	Connection port for `gpperfmon` agents (`gpmmon` and `gpsmon`) executing on ADB hosts. Configure by setting the `gpperfmon_port` parameter in *postgresql.conf* on coordinator and segment hosts
PXF	5888	TCP	The PXF Java service runs on the `5888` port on each ADB segment host
gpbackup	25 587	TCP	—

Cluster or external hosts

Service	Port	Protocol	Note
gpfdist	8080	HTTP	The `gpfdist` file serving utility can run on ADB hosts or external hosts. You can specify the port range with the `-p` (start port) and `-P` (end port) options when starting the server. The `8080` value is used as the start port by default
gpload	8000-9000	HTTPS	The `gpload` utility runs one or more instances of `gpfdist` with ports or port ranges specified in the configuration file. The `8000-9000` range is used by default and can be changed

ADQM/ClickHouse host

Service	Port	Protocol	Note
ADB ClickHouse Connector	8123	HTTP	All ADB nodes should have access to the `8123` HTTP port on all ADQM/ClickHouse nodes through which you plan to load data

> If you use ADB to ADB Connector, ensure that TCP ports that are used in connection are allowed on remote ADB cluster hosts (coordinator host, segment hosts, standby host). You can obtain the list of these ports using the following query on the remote ADB cluster:
>
> ```
> SELECT port FROM gp_segment_configuration;
> ```
>
> Add exceptions for the ports from the command output in case of using firewall.

ADB Control ports

> **NOTE**
>
> For more information on the ADB Control architecture, see Arenadata DB Control overview.

ADB Control host

Service	Port	Protocol	Note
Service Registry	6567	gRPC (HTTP 2.0)	gRPC endpoint of Service Registry instances
	8761	HTTP	—
UI Backend Server	6566	gRPC (HTTP 2.0)	—
	8890	HTTP	—
	81	HTTP	Entry to the ADB Control UI
Backend Server	6565	gRPC (HTTP 2.0)	—
	8081	HTTP	—
Scheduler	6572	gRPC (HTTP 2.0)	—
Query DB (PostgreSQL)	5433	TCP	The outside port is mapped to `5432` inside a container
Metrics DB (ClickHouse)	8123	HTTP	HTTP API port for HTTP requests. Used by JDBC, ODBC, and web interfaces
	9000	TCP	Native protocol port (also referred to as ClickHouse TCP protocol). Used by ClickHouse applications and processes like `clickhouse-server`, `clickhouse-client`, and native ClickHouse tools. Used for distributed queries via inter-server communication
Migration	8899	HTTP	—

ADB hosts

Service	Port	Protocol	Note
Agent	6581	TCP	Port for the utility service that is used to manage GUCs which are changed via ADB Control
Agent	5701	TCP	Port that is used as a part of the Hazelcast (distributed cache) cluster node address. The port should be opened on every ADB cluster host (with the ADB Control agent) since it is used for communication between Hazelcast cluster nodes
Agent	From `36001` to `36001 + <number of ADB hosts> - 1`	TCP	Range of allowed outbound ports for connections to other members or clients within a Hazelcast cluster. The ports should be opened on every ADB cluster host with the ADB Control agent

ADBM ports

> **NOTE**
>
> For more information on the ADBM architecture, see Arenadata DB Backup Manager overview.

ADBM host

Service	Port	Protocol	Note
Backup Manager	6570	TCP	—
	8870	TCP	—
	8988	TCP	The outside port is mapped to `8843` inside a container
Service Registry	6567	TCP	—
	8761	TCP	—
PostgreSQL	5432	TCP	The PostgreSQL database service is available on *localhost* and the default PostgreSQL port is `5432`
etcd	2391 2392 2393	TCP	The outside ports are mapped to `2379` inside each instance of etcd correspondingly

ADB hosts

Service	Port	Protocol	Note
Agent	6571	TCP	The port through which gRPC calls from ADBM to agents come

Arenadata Enterprise Tools ports

Port	Protocol	Note
443	TCP	Port that is used by Docker Registry
81	TCP	Port that is used to access repositories during offline installation

Software requirements

Daria Barysheva

Contents

To Table of Contents

Overview

Requirements for Astra Linux and Ubuntu

Operating system prerequisites

Overview

The software requirements for Arenadata DB installation are listed below:

1. ADB supports the following operating systems:
   - Ubuntu 22.04. See Requirements for Astra Linux and Ubuntu below.
   - AltLinux 10.2 SP (for the Enterprise edition of ADB).
   - RED OS 7.3 Certified edition (for the Enterprise edition of ADB).
   - Astra Linux 1.7.6 "Orel" SE and Astra Linux 1.7.6 "Voronezh" (for the Enterprise edition of ADB). See Requirements for Astra Linux and Ubuntu below.
2. If you use ClickHouse or PostgreSQL as external storage systems for ADB Control/ADBM (see Metrics DB and Query DB in Arenadata DB Control overview), the required versions of these databases are:
   - ClickHouse 22.8 — 25.9
   - PostgreSQL 12.20
3. ADB uses Docker as a containerization platform for some services (for example, ADB Control and ADBM) and Enterprise Tools. ADB supports Linux vendor-specific Docker engines. You can also use pre-installed Docker by setting the ***Use existing docker*** flag when configuring the ADB cluster.

> **IMPORTANT**
>
> Starting with ADB 6.27.1.56, ADB Control rewrites Docker settings in the */etc/docker/daemon.json* file for security purposes. Thus, if you need to modify this file, set the ***Use existing docker*** option when configuring ADB. This option will also allow you to use Docker in the rootless mode.

4. ADB needs Java 17 for some services (e.g. PXF, ADB Control, and ADBM). In all operating systems except Astra Linux, no additional Java installation is needed since Java comes with the main operating system repository. In Astra Linux, before ADB installation, you should install Java manually and fill in the Java path in the ***JAVA_HOME*** field when configuring the ADB cluster. This step is also necessary in other operating systems if you use Java 17 with a path different from the default one.
   The default Java paths for supported operating systems are listed below.

Ubuntu

/usr/lib/jvm/java-17-openjdk-amd64

RED OS

/usr/lib/jvm/jre-17-openjdk

AltLinux

/usr/lib/jvm/jre-17

> **IMPORTANT**
>
> - If you plan to use custom Java (such as Axiom), Java 17 is required for installation.
> - For Astra Linux, the ***JAVA_HOME*** field is mandatory starting with ADB 6.27.1.58.

5. The minimal ADCM version is 2.6.0.
6. If you use ADBM, its minimal version for ADB 7 is 2.8.0.
7. If you use ADB Control, its minimal version for ADB 7 is 4.15.0.
8. ADB requires the `en_US.UTF-8` operating system locale on all cluster hosts. Yet you do not need to select this locale as default. This locale is present by default in all supported operating systems except Astra Linux. In the case of Astra Linux, the locale is automatically added during ADB installation. To view available locales, run the following command:

```
$ localectl list-locales
```

9. It is recommended to switch off any antivirus software before ADB installation and during the ADB operation process. In case of using antivirus, please contact your antivirus software vendor to find out the settings necessary to allow ADB.
10. Only Latin characters and digits are allowed for ADB database objects. These characters are treated as reserved: `"`, `&`, `'`, ` ` (space), `(`, `)`, `;`, `<`, `>`, `\`, `` ` ``, `|`. If such characters are present, Arenadata DB Backup Manager will not be able to back up/restore a database.
11. During ADB installation, a bundle assigns the necessary permissions to the specific files and directories it creates, regardless of the `umask` value on cluster hosts.
12. During ADB installation, the following changes are made at the operating system level. Do not change any of the objects listed below.
    - The following system files are modified:
      - */etc/hosts* (optionally)
      - */etc/hostname* (optionally)
      - */etc/chrony.conf* (if you install Chrony)
      - */etc/ssh/sshd_config* (always)
      - */etc/security/limits.conf* (always)
    - A service file for configuring cgroup v2 for ADB is created at */etc/systemd/system/cgroup_v2_gpdb.service*;
    - New files are created in the */sys/fs/cgroup* directory.
13. The installation process creates a new file in the */var/spool/cron/* folder for the ADB system user (by default, `gpadmin`). All technical accounts should be able to change that file as well (via `crontab -e -u gpadmin`). ADB adds cron jobs to run the cluster maintenance scripts — their schedule and arguments should be modified using the corresponding section in the ADB configuration parameters.
14. For online installation of ADB, all cluster hosts should have access to Arenadata repositories specified on the Configuration tab in the ADB cluster menu available via ADCM UI (see ***Arenadata repo url*** and ***Monitoring repo url*** fields). For offline installation, provide access to the host where the Enterprise Tools cluster is installed.
15. For both online and offline installation, all cluster hosts should have access to the repositories that are listed below (or local mirrors should be created):
    - AltLinux: ftp.altlinux.org
    - Astra Linux: download.astralinux.ru
    - RED OS: repo.red-soft.ru and repo1.red-soft.ru
    - Ubuntu: archive.ubuntu.com/ubuntu

Requirements for Astra Linux and Ubuntu

1. ADB, like other analytical DBMS, is optimized to work with generic kernels. Using the lowlatency kernels may not provide optimal performance for tasks that are typical for analytical data processing. Due to this, for proper operation of ADB, it is recommended to use generic cores only.
2. You should also consider version compatibility.

> **NOTE**
>
> ADB 6 was built upon cgroup v1. Starting with version 7, ADB uses cgroup v2 that is the default in all supported operating systems.

Operating system prerequisites

Below is the list of prerequisites for your operating system environment to ensure proper ADB operation. You can either prepare the environment manually on all ADB hosts, or select the corresponding configuration parameters in the ADB service to automatically apply the recommended settings during ADB installation.

Before installing ADB, run the Precheck cluster action, which includes verification of the host operating system parameters.

Requirement	Name of ADB configuration parameter

The */etc/hosts* file contains hostnames of all cluster hosts: coordinator host, standby coordinator host, and segment hosts. For example:
`192.168.1.10 mdw.example.com`
`192.168.1.20 smdw.example.com`
`192.168.1.30 sdw1.example.com`
`192.168.1.40 sdw2.example.com`
Alternatively, DNS can be used for hostname resolution
	Edit /etc/hosts

Security Enhanced Linux (SELinux) is disabled	***Disable linux security module*** (requires host reboot)
IPv6 is disabled	***Disable IPv6***
The value of the `MaxStartups` parameter of the OpenSSH Daemon (sshd) is `1000:30:1022`	***Set up MaxStartups for sshd***
Transparent Huge Pages in the Linux kernel are disabled	***Set up GRUB for DBMS*** (requires host reboot)
The following limits are set for the default database user in the */etc/security/limits.conf* file: - The soft and hard limits for the number of open file descriptors are set to `524288`. - The soft and hard limits for the number of processes are set to `150000`.	***Set up ulimits for DBMS user***
The firewall is disabled on ADB hosts	***Disable firewall***
The recommended kernel parameters are as follows (use the `sysctl -a` command to view all kernel settings):	***Sysctl parameters***

```
    kernel.core_pipe_limit = 0
    kernel.core_uses_pid = 1
    kernel.msgmax = 65536
    kernel.msgmnb = 65536
    kernel.msgmni = 2048
    kernel.sem = 250 2048000 200 8192
    kernel.shmmni = 4096
    kernel.sysrq = 1
    net.core.netdev_max_backlog = 10000
    net.core.rmem_max = 2097152
    net.core.wmem_max = 2097152
    net.ipv4.conf.all.arp_filter = 1
    net.ipv4.conf.default.accept_source_route = 0
    net.ipv4.ip_local_port_range = 12000 65535
    net.ipv4.ipfrag_high_thresh = 41943040
    net.ipv4.ipfrag_low_thresh = 31457280
    net.ipv4.ipfrag_time = 60
    net.ipv4.tcp_max_syn_backlog = 4096
    net.ipv4.tcp_syncookies = 1
    vm.dirty_background_bytes = 1610612736
    vm.dirty_background_ratio = 0
    vm.dirty_bytes = 4294967296
    vm.dirty_expire_centisecs = 500
    vm.dirty_ratio = 0
    vm.dirty_writeback_centisecs = 100
    vm.overcommit_memory = 2
    vm.overcommit_ratio = 95
    vm.swappiness = 10
    vm.zone_reclaim_mode = 0
```

> **IMPORTANT**
>
> If you change the ***Disable linux security module*** and ***Set up GRUB for DBMS*** parameters, reboot the affected hosts. To do so:
>
> - To apply settings on the cluster hosts during installation — set the ***Reboot cluster servers after installation*** flag in the ***Install*** or ***Reinstall*** cluster action.
> - To apply settings on hosts that you add to the ADB service:
>   - Set the ***Reboot new segment host after installation*** flag in the ***Move Host Segment*** cluster action.
>   - Set the ***Reboot new servers after installation*** flag in the ***Expand*** or ***Init Standby Master*** cluster action.
>
> Otherwise, these settings will not apply to the hosts. If the ***Reboot cluster servers after installation*** flag was not set, it is recommended to stop the cluster and reboot the hosts manually to apply the specified settings.

Installation

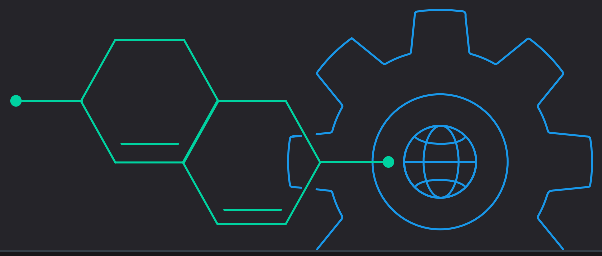
To Table of Contents

This tutorial guides you through the process of installing an Arenadata DB (ADB) 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 ADB via ADCM in two ways.

Online installation

on hosts with access to Internet



Offline installation

on hosts without access to Internet

Online installation

To Table of Contents

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

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

Step 1.

Install ADCM

Step 2.

Prepare hosts

ARENADATA
DB

Step 3.

Install ADB cluster

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
  - Online update
  - Offline update

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
```

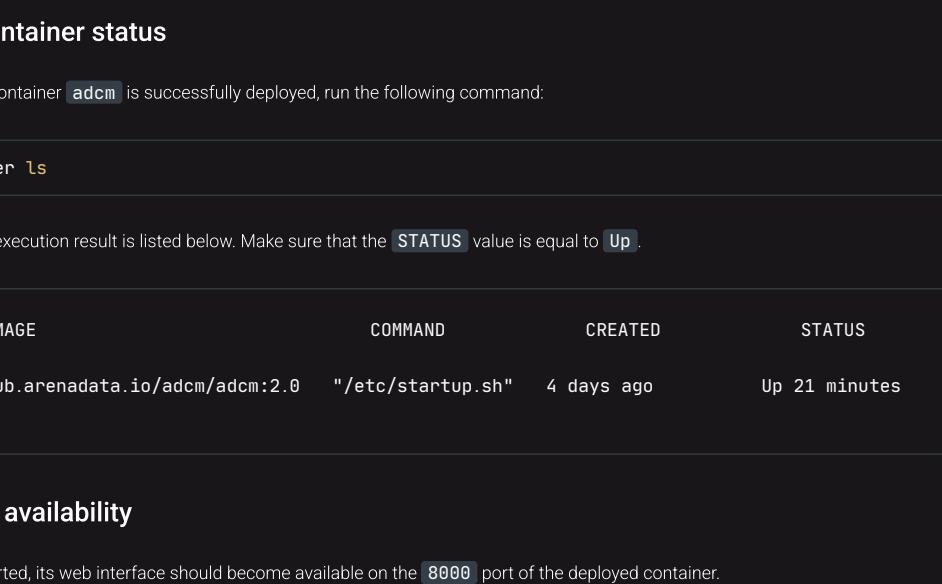
The example of the command execution result:

```
<!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:

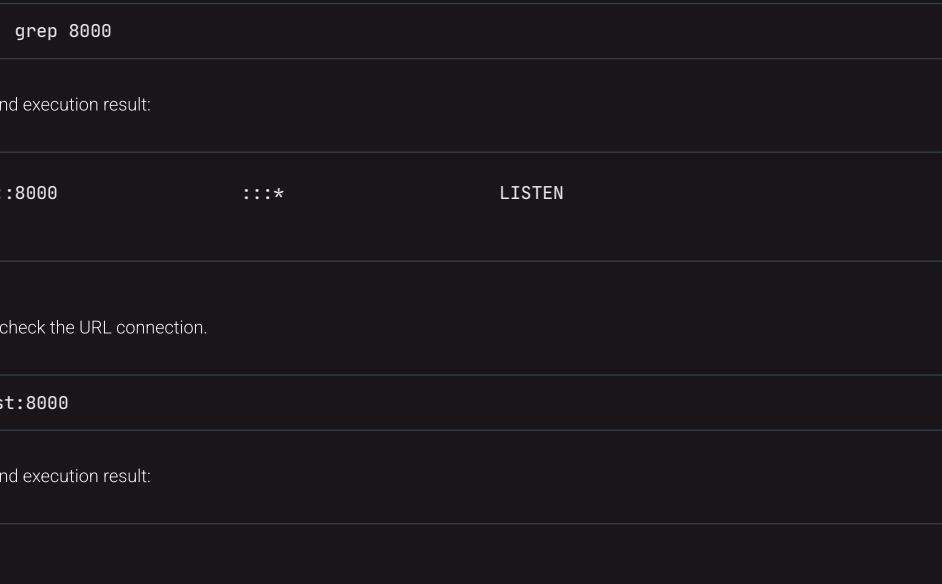
- ***User***: admin
- ***Password***: admin



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***.

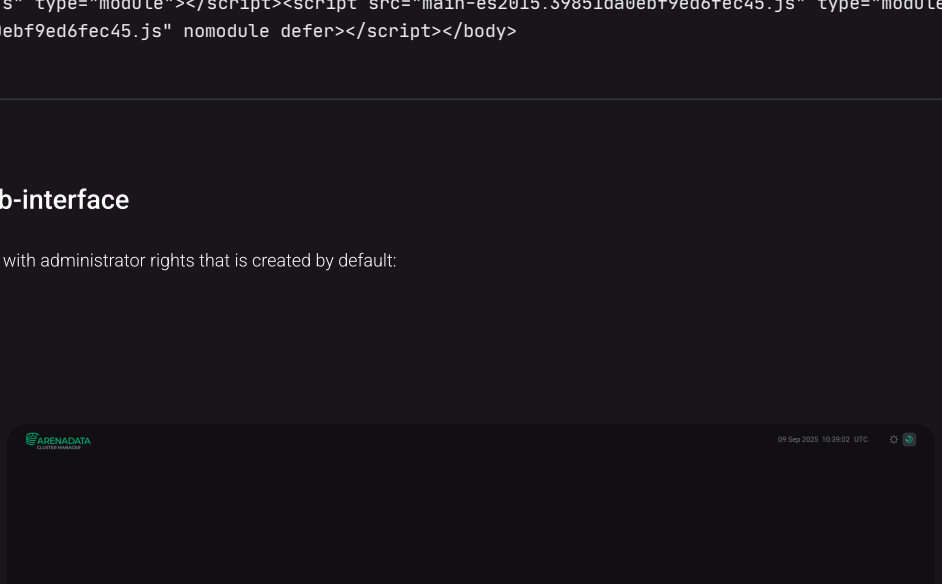


Change the admin password

Set the ADCM URL

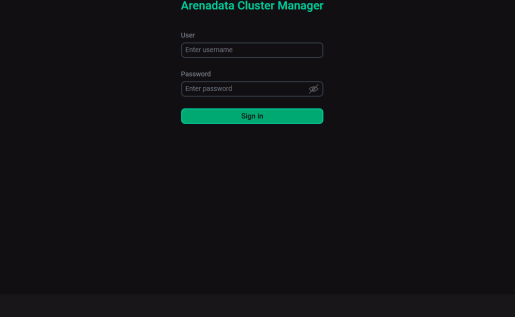
For successful communication between ADCM and product clusters deployed via ADCM, it is important to correctly define the external ADCM address. This address is used by cluster components for sending information about their state to ADCM. To specify the ADCM URL, use the environment variable `DEFAULT_ADCM_URL` 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.



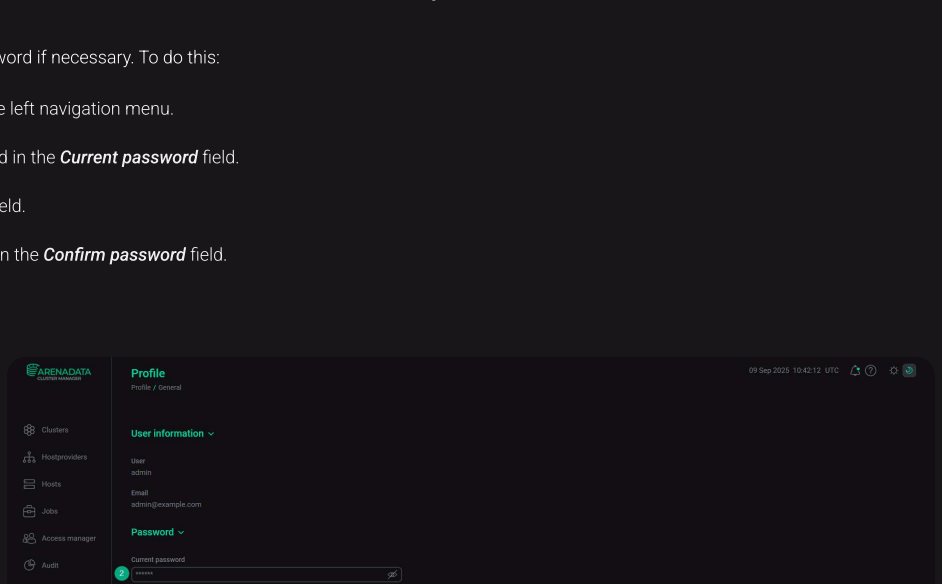
The Settings page

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



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).

Online update

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
```

Offline update

You can also try offline ADCM update in situations without Internet access.

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

To Table of Contents

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
- ADB Cloud Bundle

CAUTION

Starting with the 6.23.3.44 version, ADB supports both FQDN and short host names. For the previous ADB versions, it is not recommended to use FQDNs (as FQDNs cause errors during the ***Expand*** action and a workaround is needed).

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.

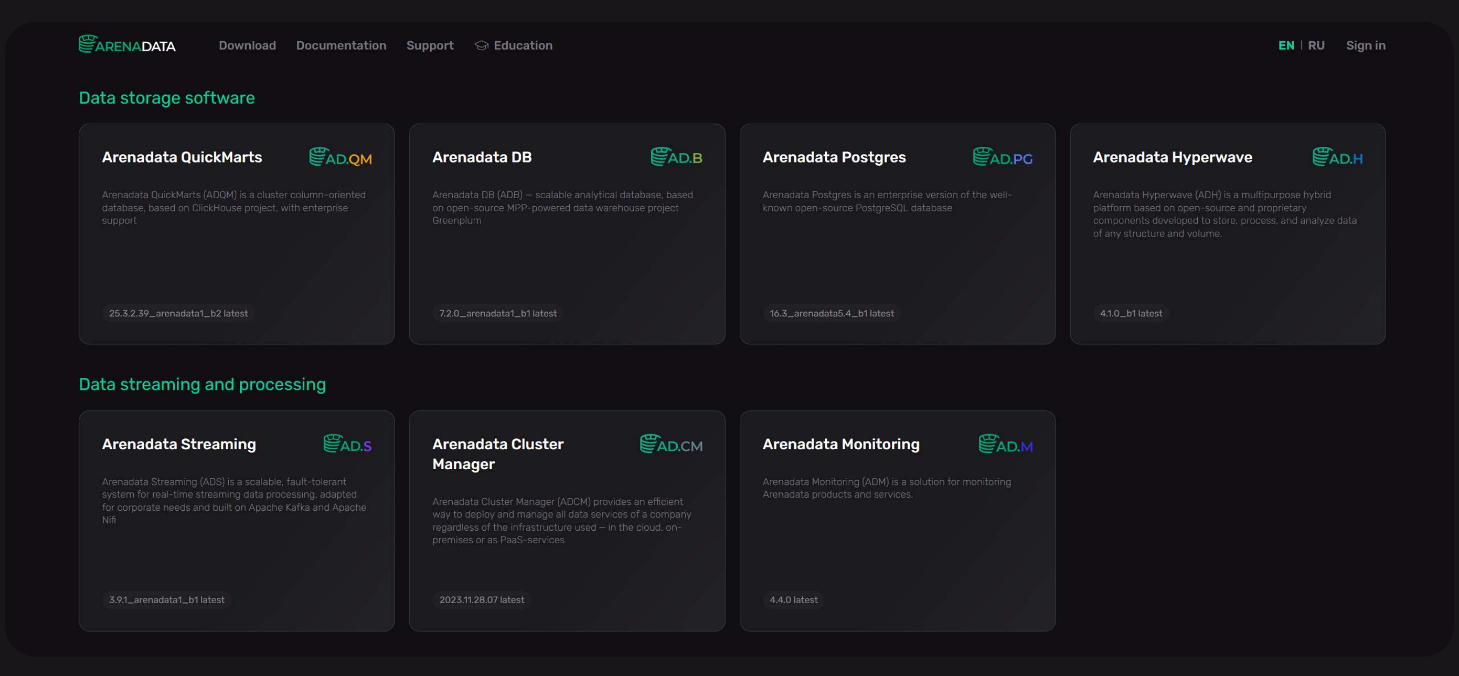
The steps for adding a cluster to ADCM are listed below.

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 DB***.



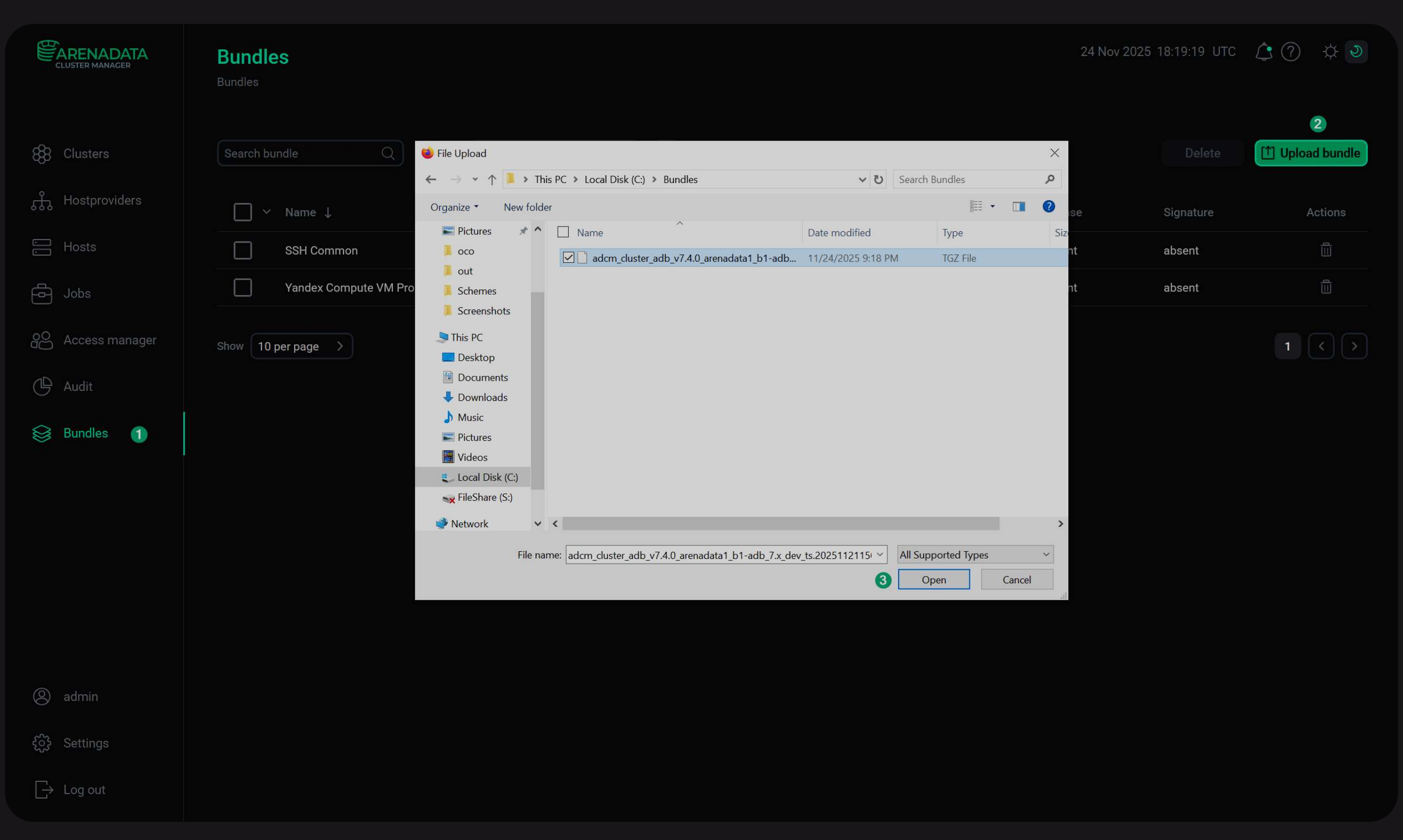
Switch to the bundle download page

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

Step 2. Upload a cluster bundle to ADCM

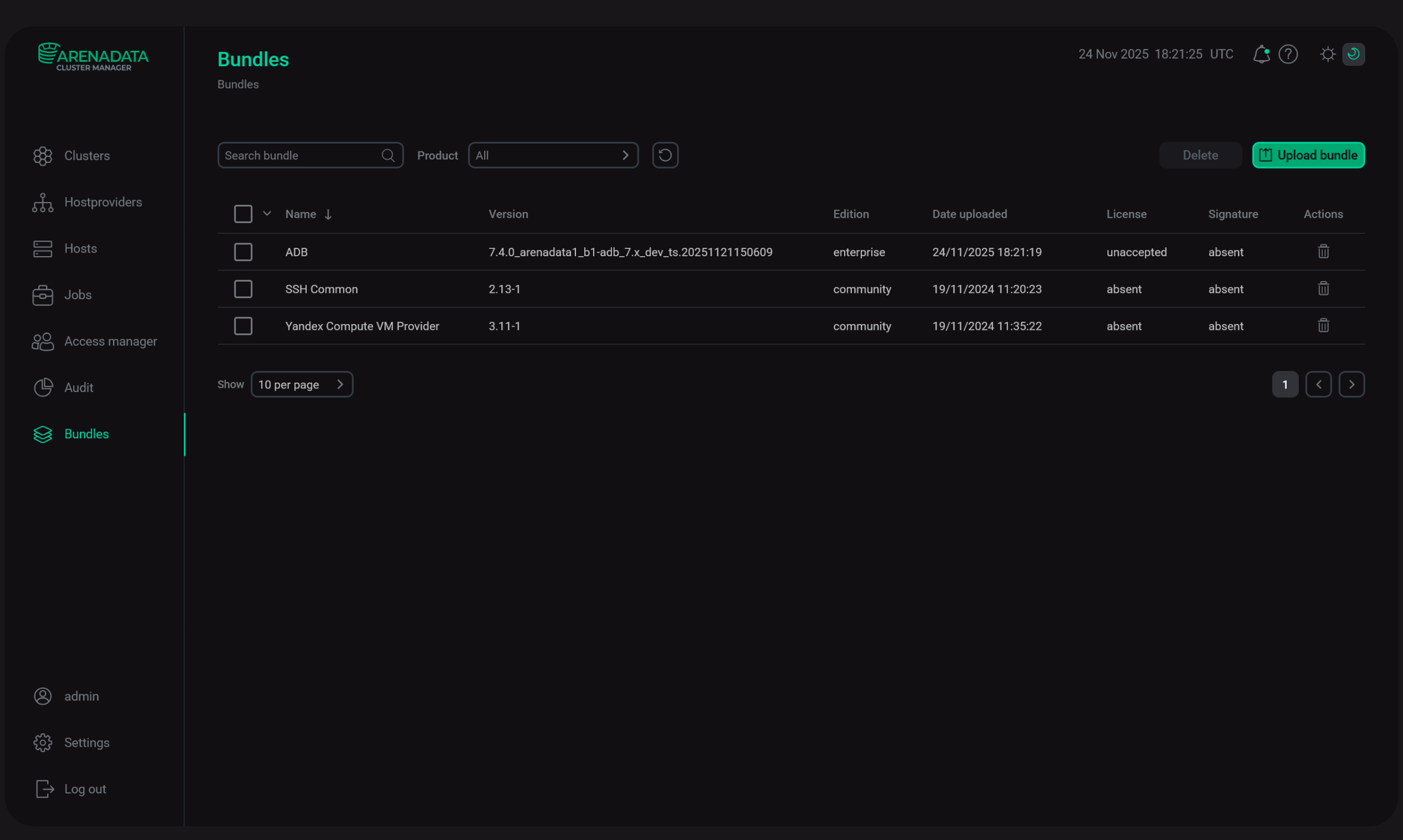
To upload a bundle to ADCM, follow the steps:

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

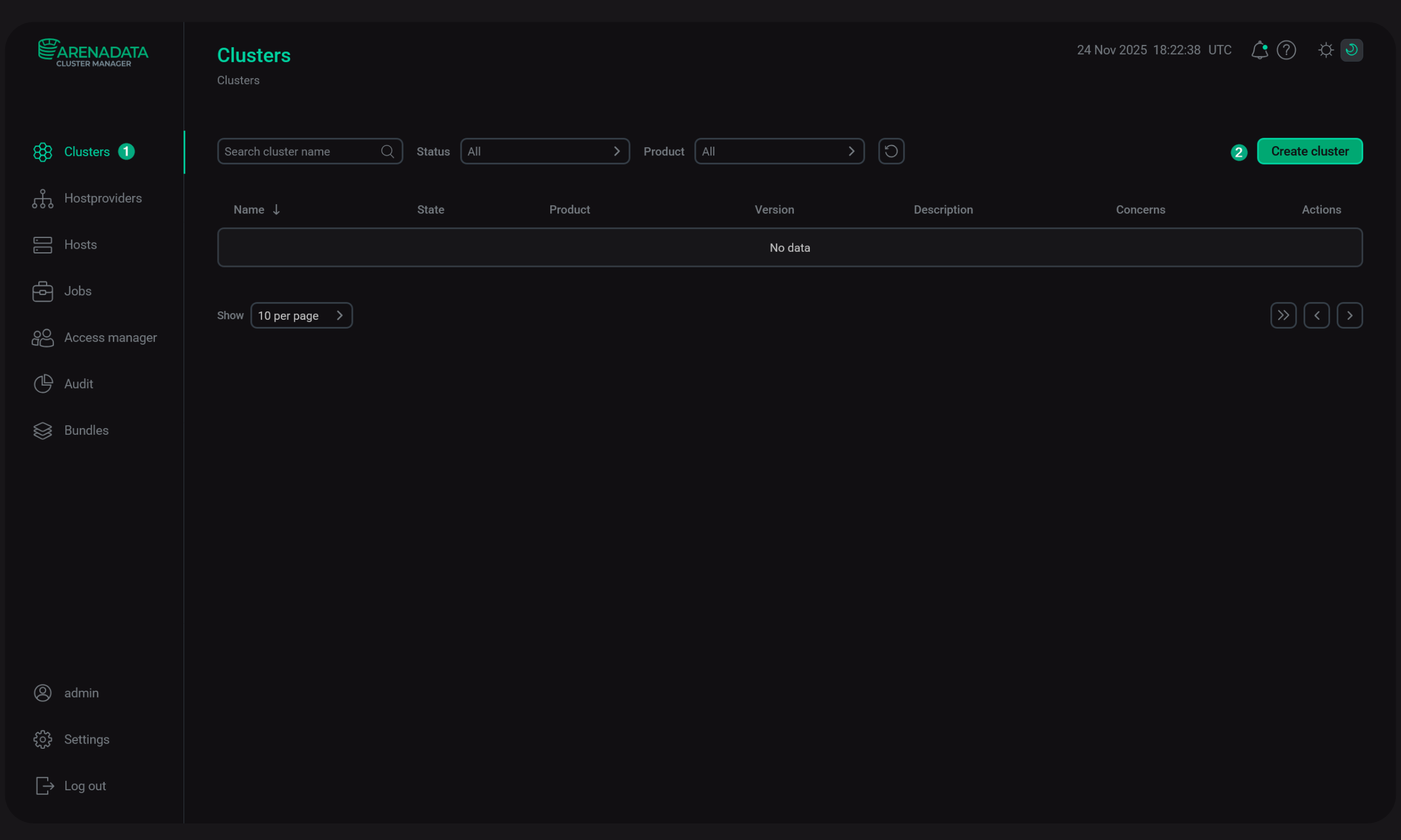
Step 3. Create a cluster based on the uploaded bundle

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.

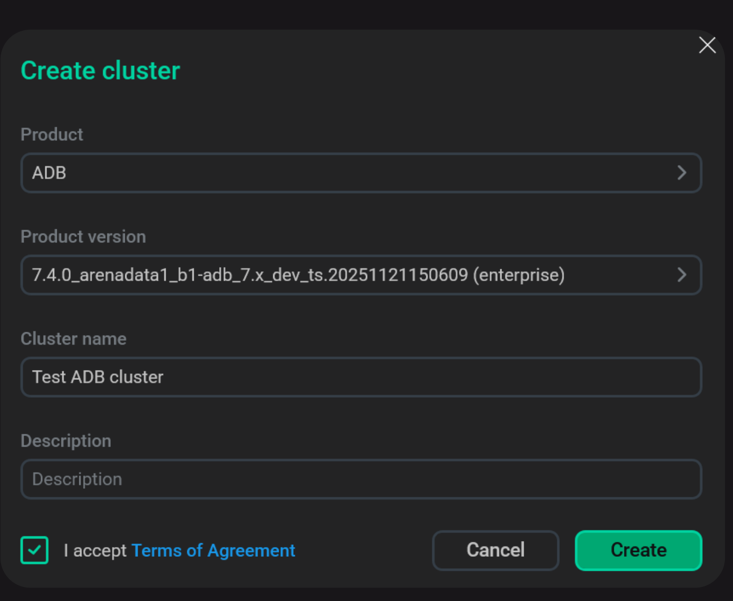
The steps for cluster creation are given below:

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: ADB

Product version: 7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609 (enterprise)

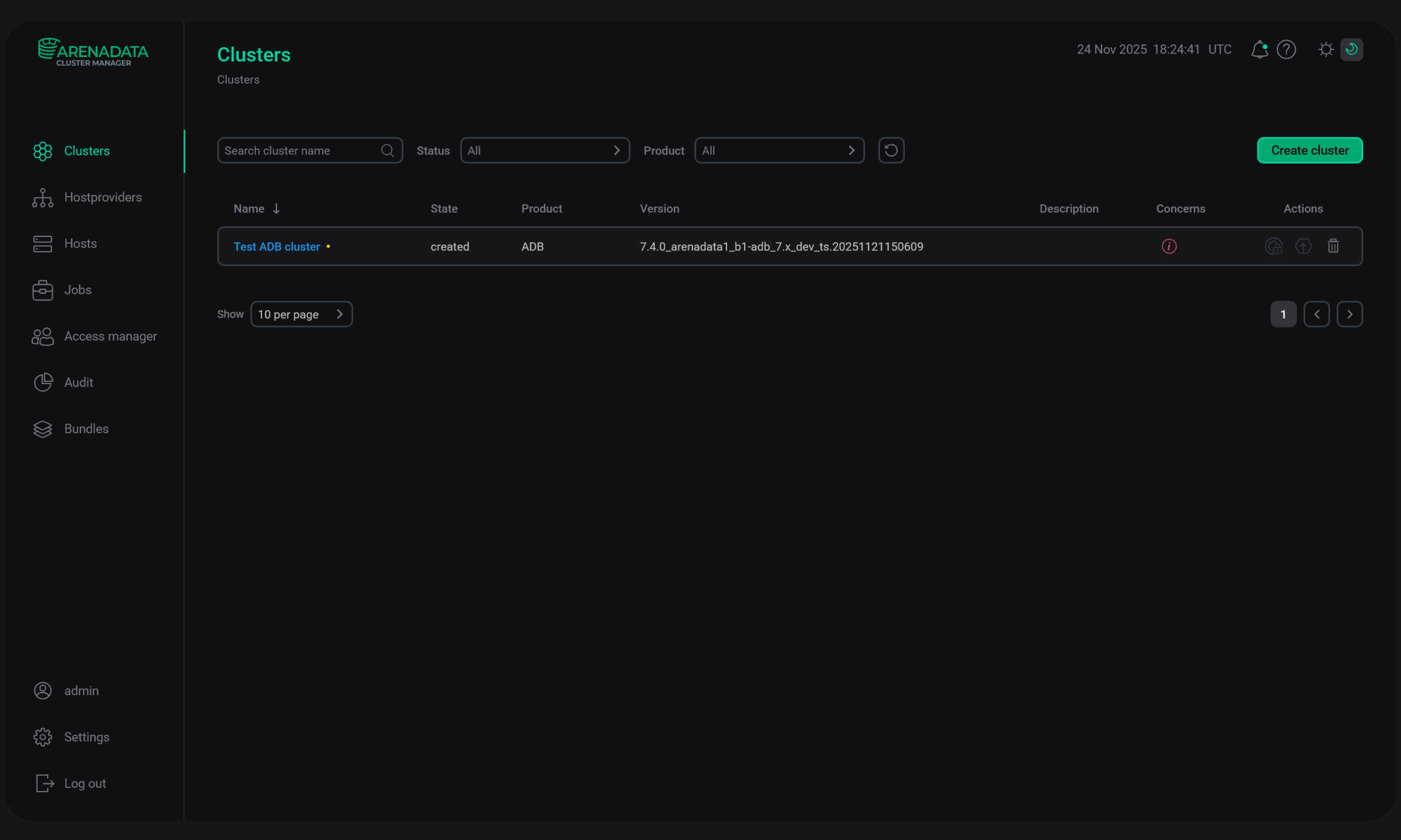
Cluster name: Test ADB cluster

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

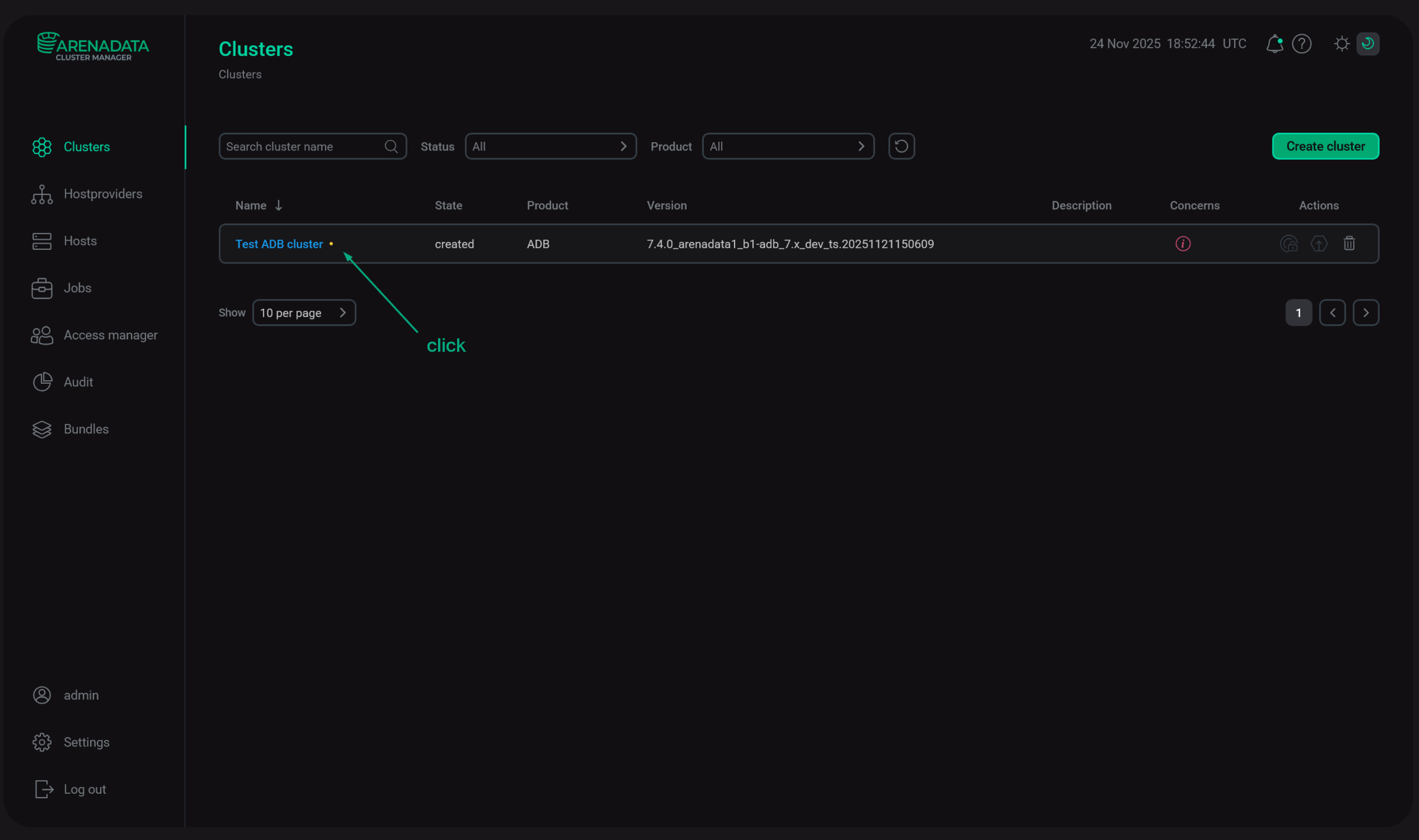
Add services

Daria Barysheva

To Table of Contents

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

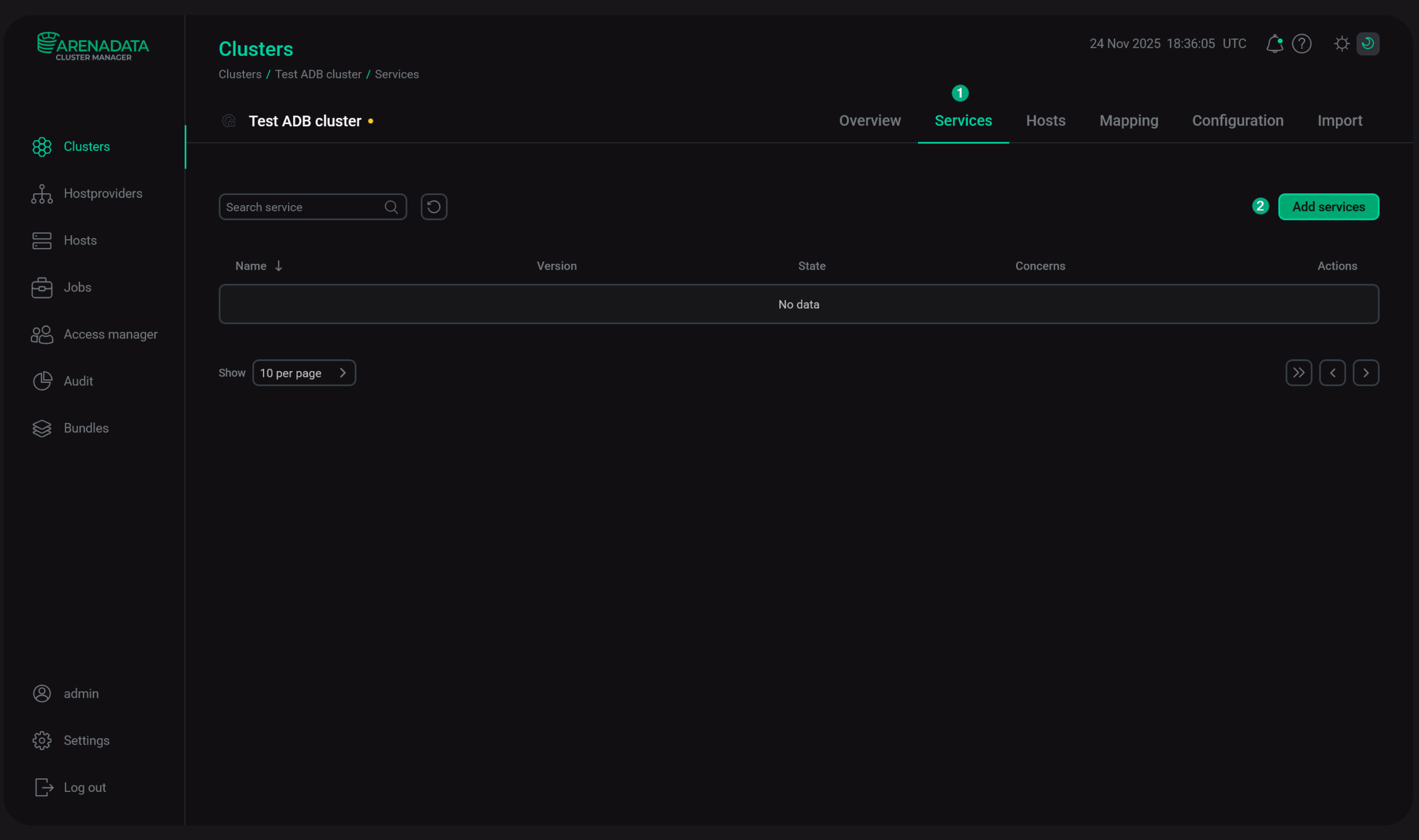
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters
24 Nov 2025 18:52:44 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1
click
admin
Settings
Log out

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

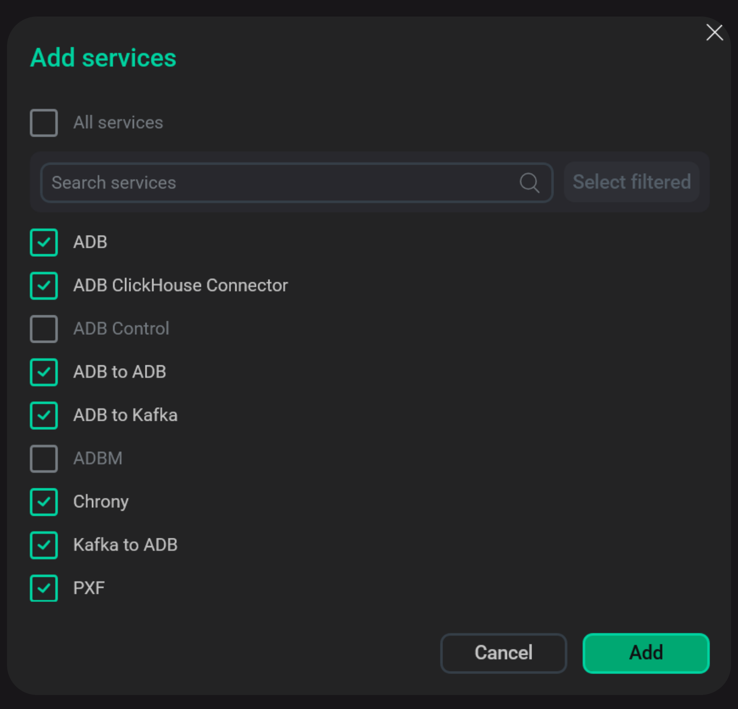
Select a cluster

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


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services
24 Nov 2025 18:36:05 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
2
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add services
Name
Version
State
Concerns
Actions
No data
Show
10 per page
admin
Settings
Log out

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

☑ ADB
☑ ADB ClickHouse Connector
☐ ADB Control
☑ ADB to ADB
☑ ADB to Kafka
☐ ADBM
☑ Chrony
☑ Kafka to ADB
☑ PXF

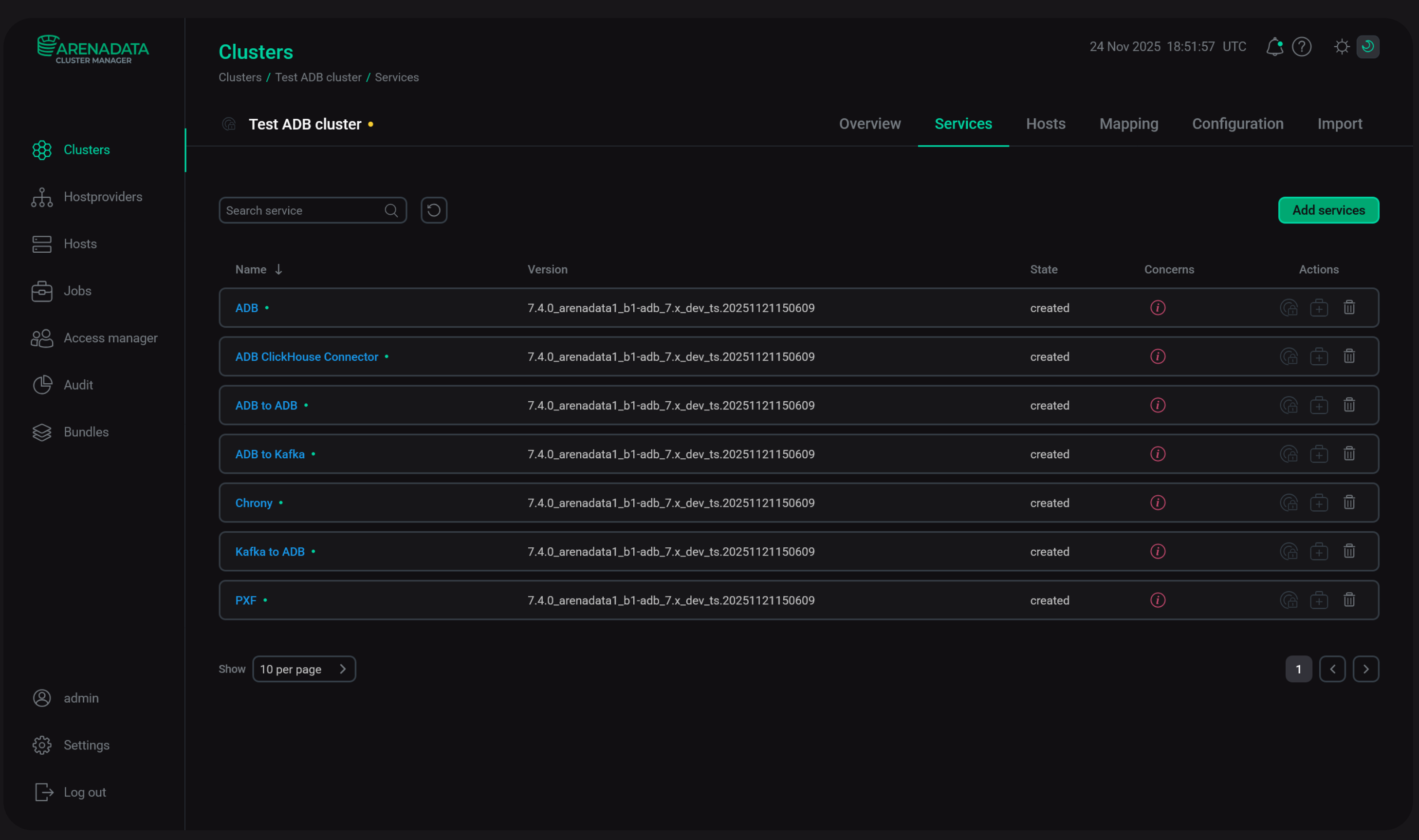
Select services

The brief description of available services is listed below. `CE` means the Common Edition of ADB, `EE` — the Enterprise Edition of ADB.

Services that can be added to the ADB cluster

Service	Mandatory	Edition	Purpose
ADB	Yes	CE/EE	A core service, corresponding to the massively parallel DBMS based on Greengage DB
Chrony	No	CE/EE	A time service. Provides time synchronization for the cluster hosts, that is necessary for the cluster to work correctly. If the time service is already configured on the hosts, there is no need to install Chrony
PXF	No	CE/EE	A service that provides ADB interactions with external data sources
ADB ClickHouse Connector	No	EE	ADB ClickHouse Connector — a service that provides two-way ADB interactions with ClickHouse. Requires PXF
ADB to Kafka	No	EE	ADB to Kafka Connector — a service that provides one-way ADB interactions with Kafka. Requires PXF
Kafka to ADB	No	EE	Kafka to ADB Connector — a service that provides one-way Kafka interactions with ADB
ADB to ADB	No	EE	ADB to ADB Connector — a service that provides two-way interactions between ADB clusters
ADBM	No	EE	Arenadata DB Backup Manager — a service for ADB binary backups management. Can be installed only offline
ADB Control	No	EE	Arenadata DB Control (ADB Control) — a service for monitoring, diagnostics, and management of queries, that run in the ADB cluster. Can be installed only offline

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


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services
24 Nov 2025 18:51:57 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add services
Show
10 per page
1
admin
Settings
Log out

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	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.

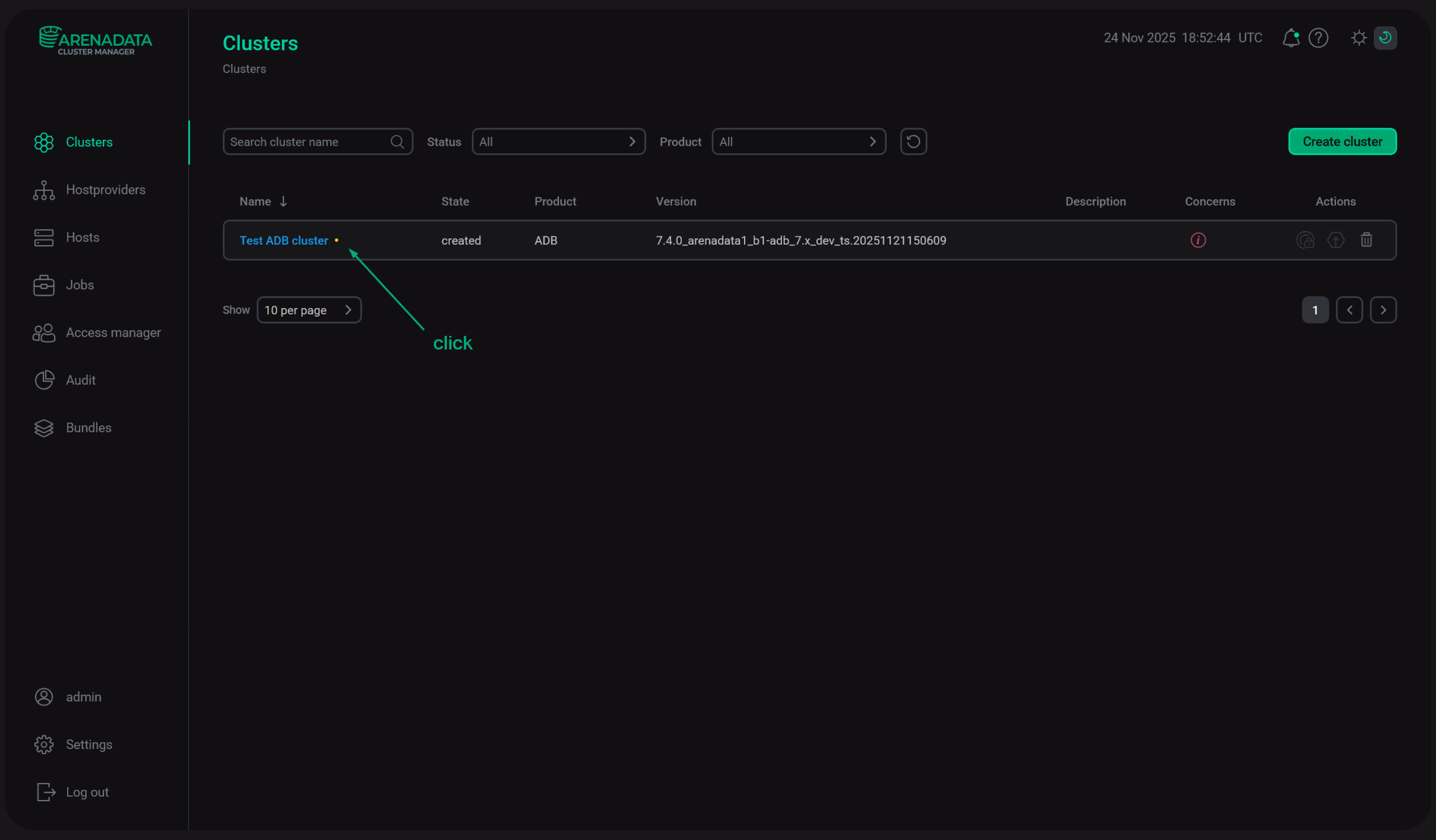
Add hosts to a cluster

Daria Barysheva

To Table of Contents

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

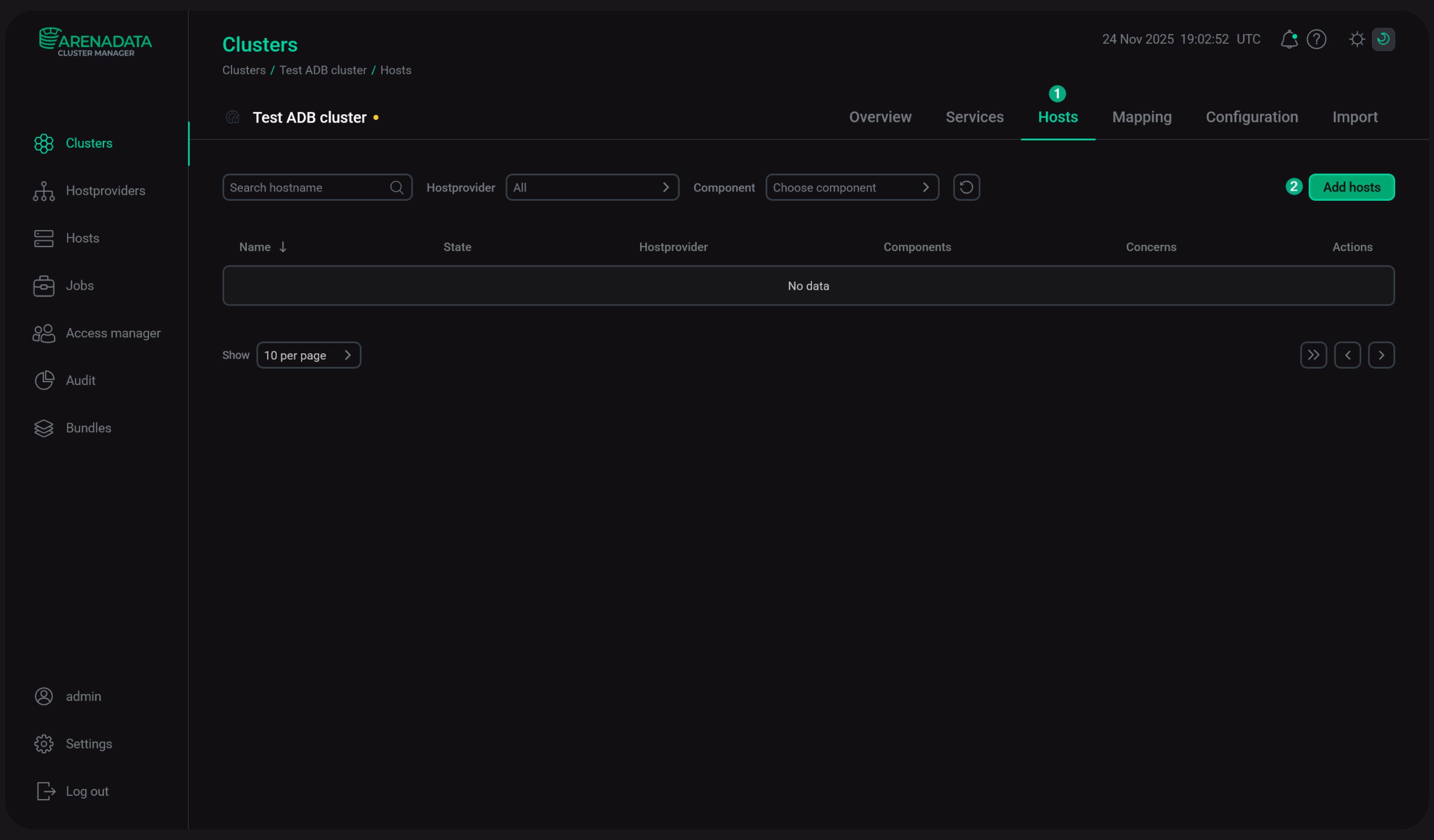
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
24 Nov 2025 18:52:44 UTC
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1
click

Name ↓	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

Select a cluster

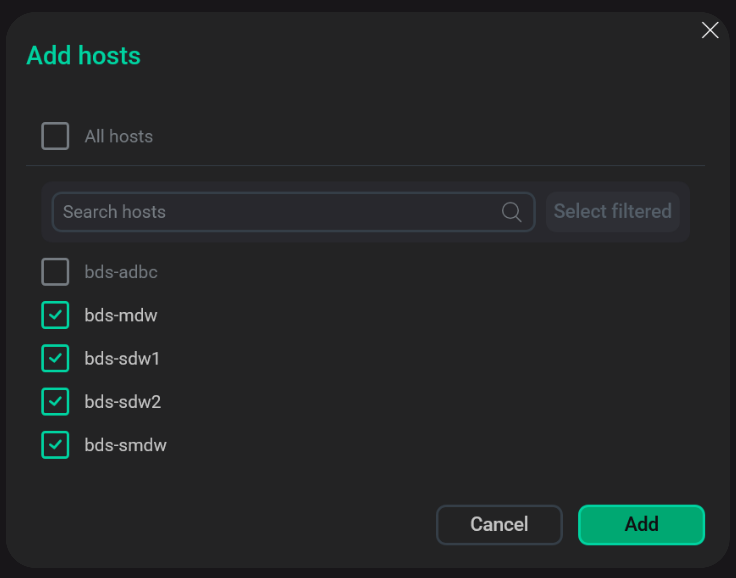
2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADB cluster / Hosts
24 Nov 2025 19:02:52 UTC
Test ADB cluster
Overview
Services
1
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Component
Choose component
2
Add hosts
Show
10 per page

Name ↓	State	Hostprovider	Components	Concerns	Actions
No data					

Switch to adding hosts

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

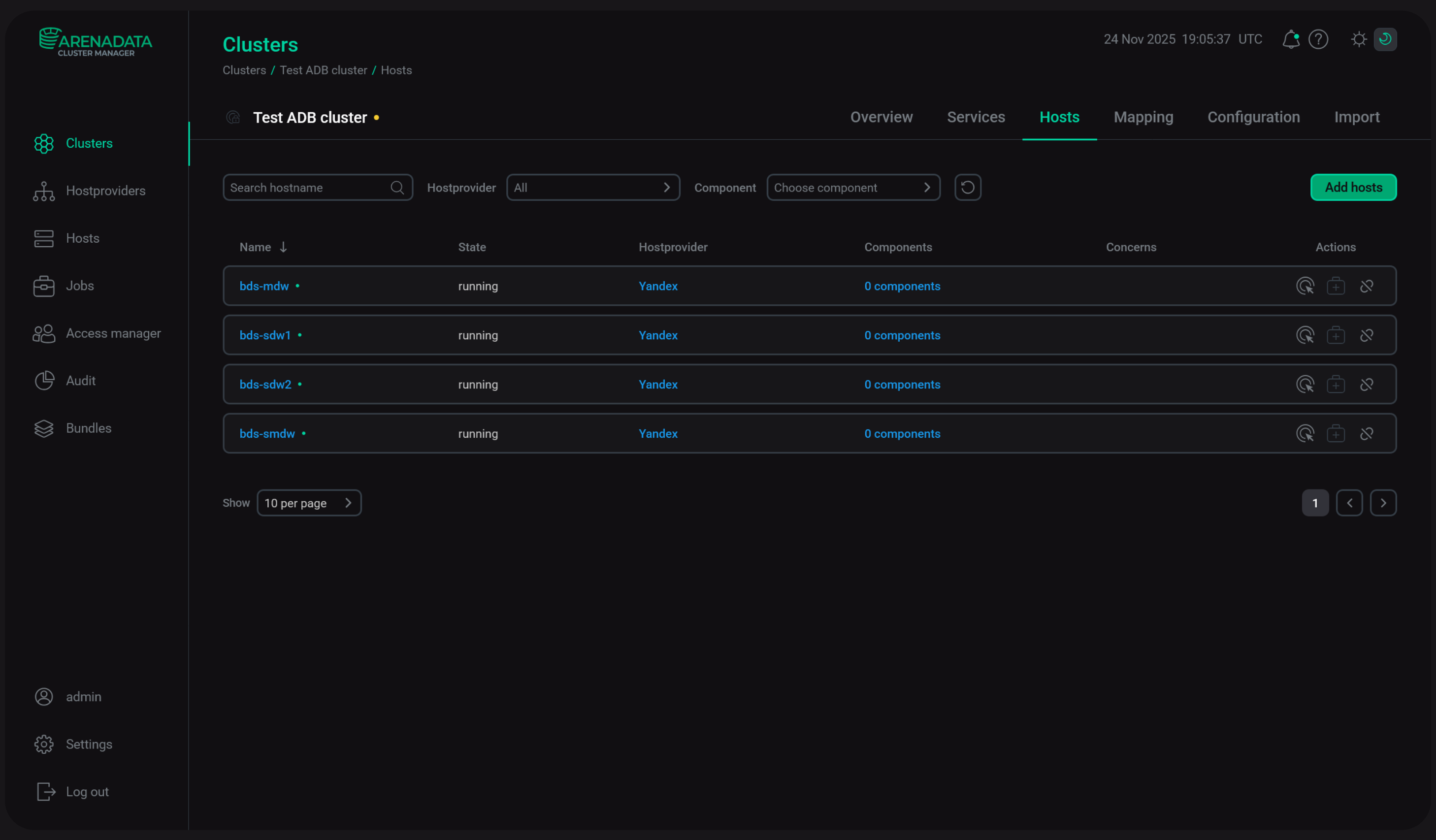

Add hosts
Search hosts
Select filtered
Cancel
Add

☐ All hosts

☐ bds-adbc
☑ bds-mdw
☑ bds-sdw1
☑ bds-sdw2
☑ bds-smdw

Select hosts

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


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADB cluster / Hosts
24 Nov 2025 19:05:37 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Component
Choose component
Add hosts
Show
10 per page
1

Name ↓	State	Hostprovider	Components	Concerns	Actions
bds-mdw	running	Yandex	0 components		
bds-sdw1	running	Yandex	0 components		
bds-sdw2	running	Yandex	0 components		
bds-smdw	running	Yandex	0 components		

The result of successful adding hosts to a cluster

Add components

Daria Barysheva

To Table of Contents

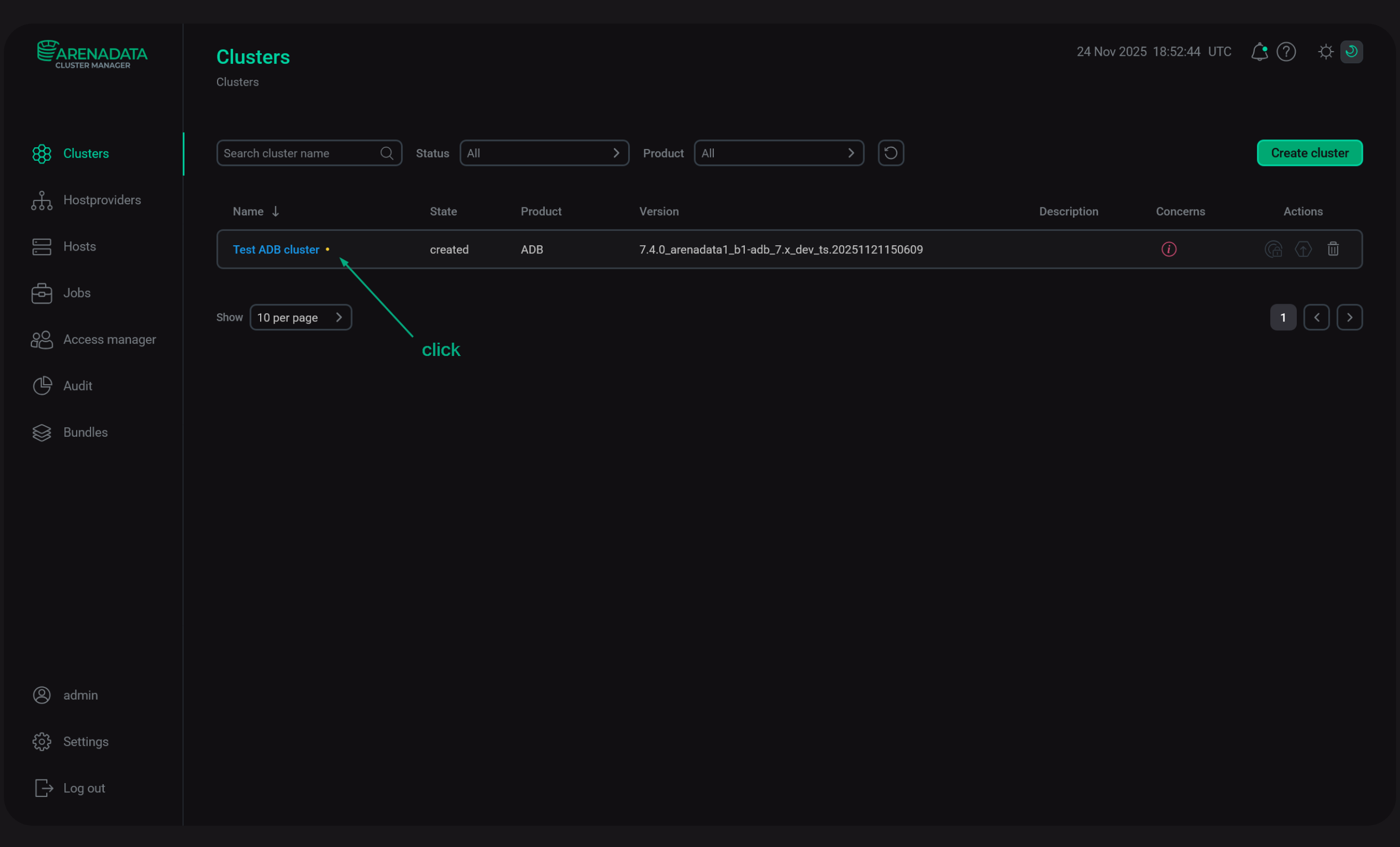
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts. For example, ADB service consists of such components as ADB Master, ADB Segment, ADB Standby, etc.

IMPORTANT

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

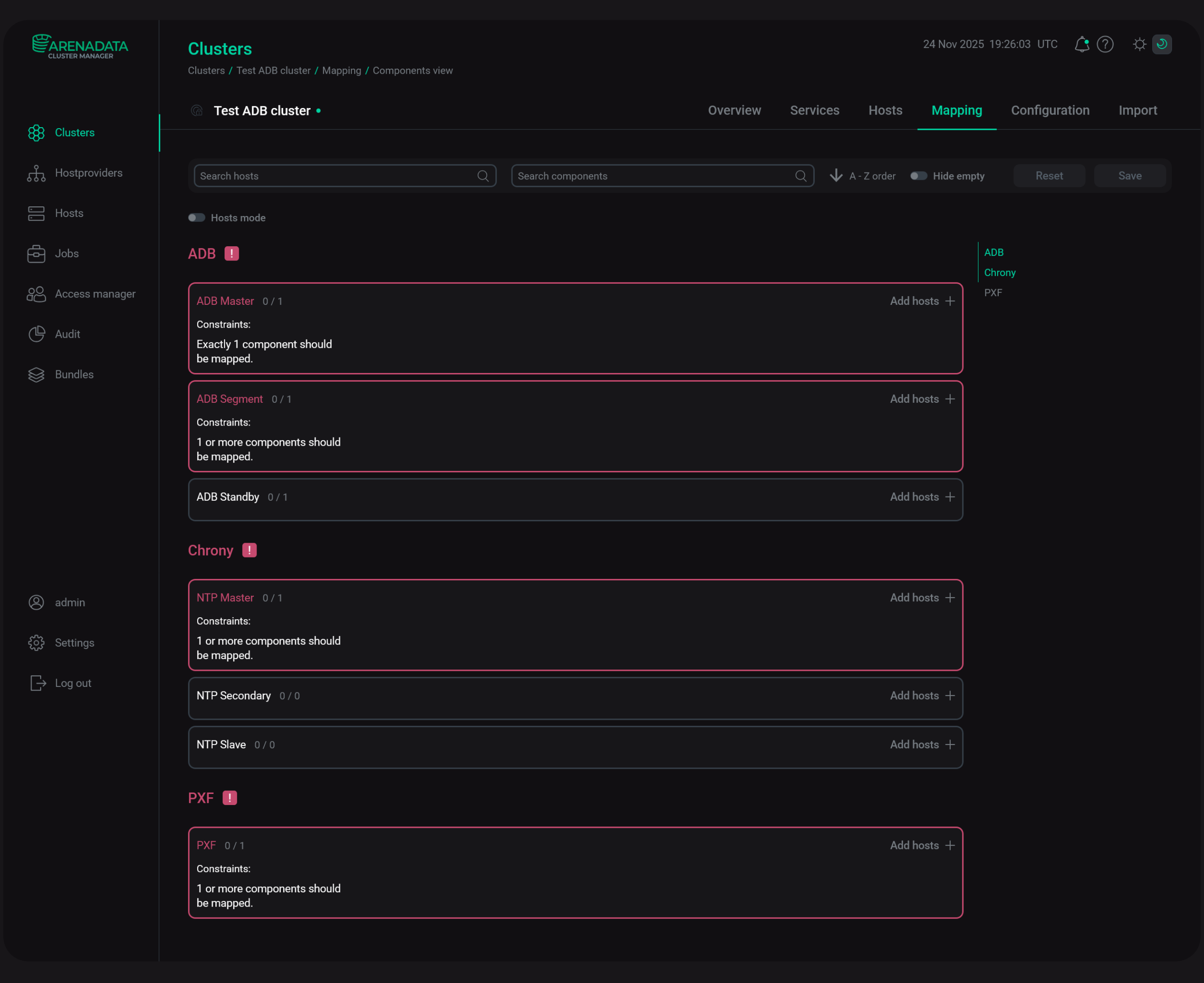
The steps for adding components to hosts are listed below:

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


ARENADATA CLUSTER MANAGER
Clusters
24 Nov 2025 18:52:44 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search cluster name
Status All
Product All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Test ADB cluster
created
ADB
7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609
Show 10 per page
click
admin
Settings
Log out

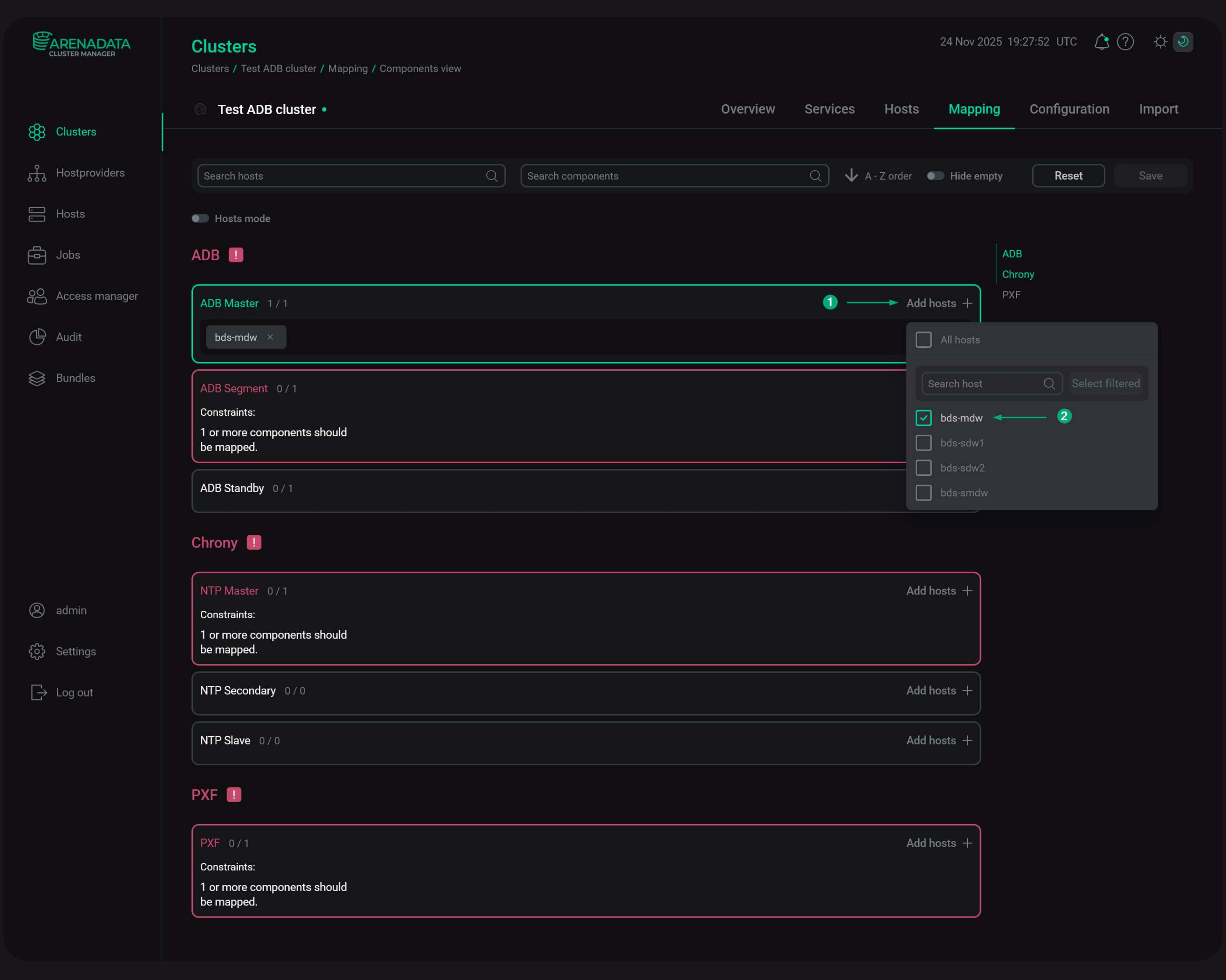
Select a cluster

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


Clusters
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
Reset
Save
Hosts mode
ADB
ADB Master 0 / 1
Constraints:
Exactly 1 component should be mapped.
ADB Segment 0 / 1
Constraints:
1 or more components should be mapped.
ADB Standby 0 / 1
Chrony
NTP Master 0 / 1
Constraints:
1 or more components should be mapped.
NTP Secondary 0 / 0
NTP Slave 0 / 0
PXF
PXF 0 / 1
Constraints:
1 or more components should be mapped.
Add hosts

Switch to mapping service components

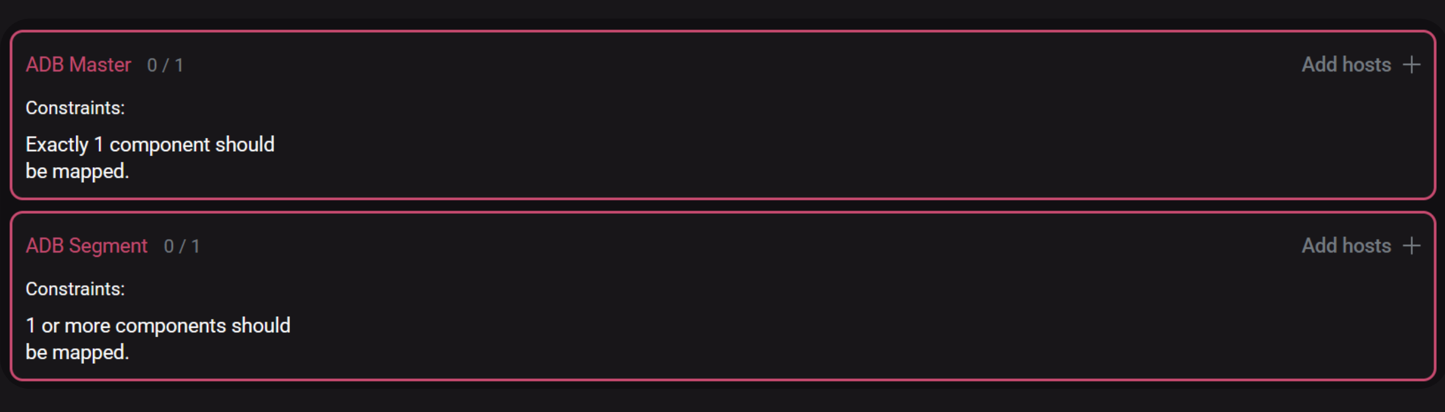
3. In the window that opens, map components to hosts. To assign hosts for each component, click ***Add hosts*** and select hosts in the pop-up window.


Clusters
Test ADB cluster
Mapping
ADB Master 1 / 1
bds-mdw
ADB Segment 0 / 1
ADB Standby
Chrony
NTP Master
NTP Secondary 0 / 0
NTP Slave 0 / 0
PXF
PXF 0 / 1
Constraints:
1 or more components should be mapped.
Add hosts
All hosts
bds-mdw

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.


ADB Master 0 / 1
Add hosts
Constraints:
Exactly 1 component should be mapped.
ADB Segment 0 / 1
Add hosts
Constraints:
1 or more components should be mapped.

Mandatory components

CAUTION

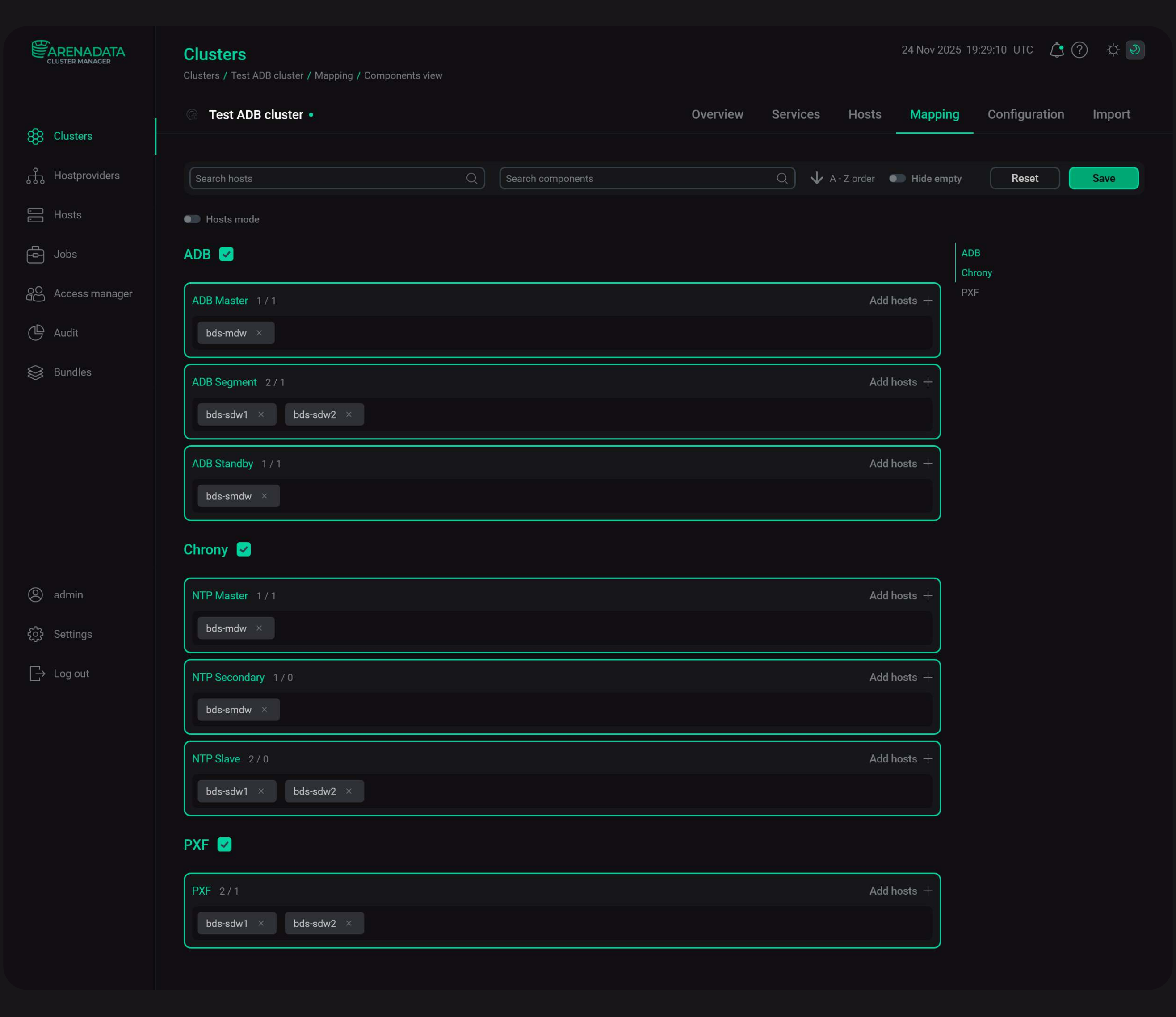
Without assigning a necessary number of hosts to mandatory components, the component mapping cannot be saved.

The typical scheme of mapping components in ADB clusters is described in the table below. It is based on the Enterprise Edition of ADB.

Example of mapping ADB components to hosts

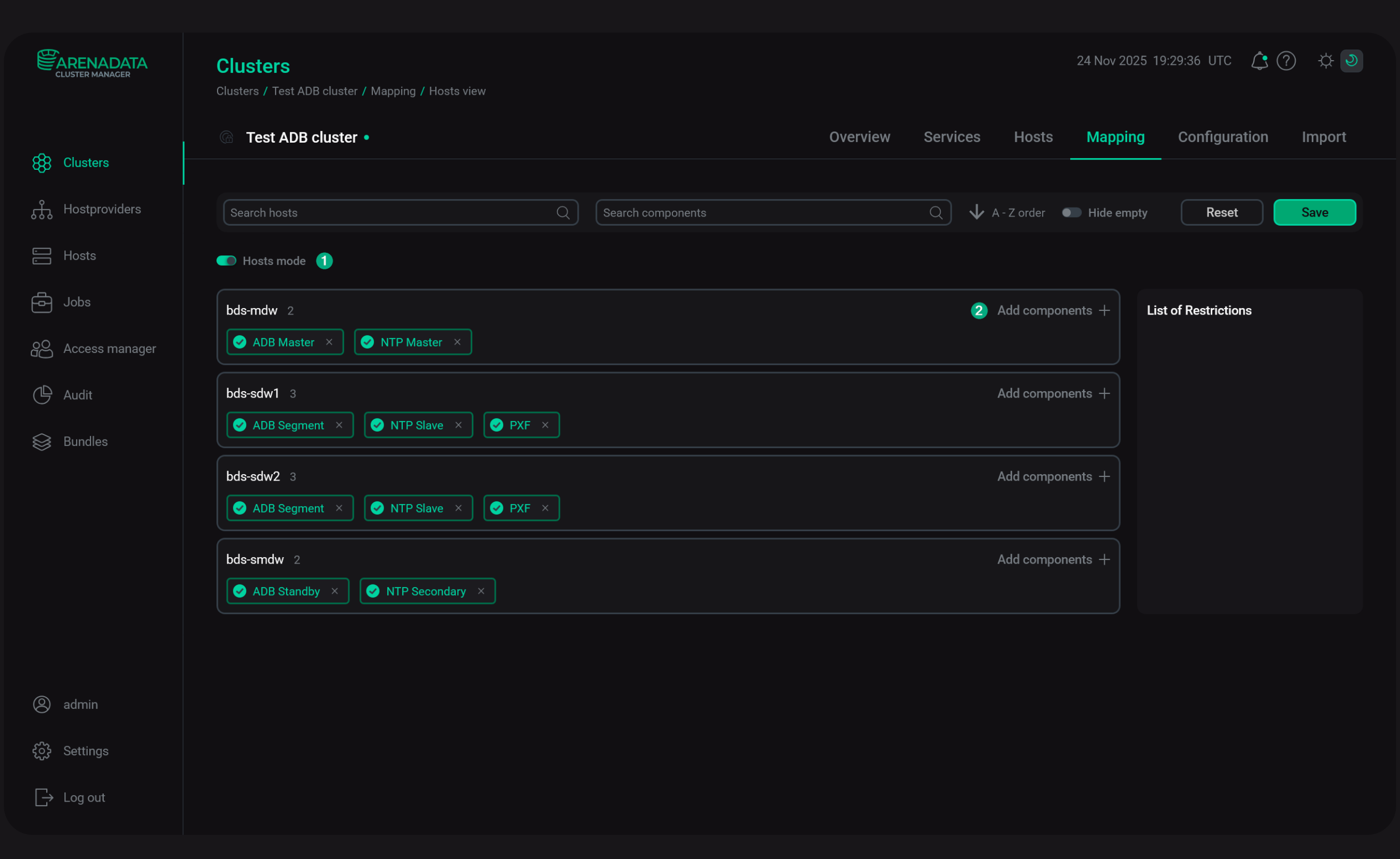
Host	Service	Component	Notes
Coordinator host	ADB	ADB Master	Should be installed on the coordinator host
	Chrony	NTP Master	Should be installed on the coordinator host
Secondary Master Host (Standby)	ADB	ADB Standby	Should be installed on the standby coordinator host
	Chrony	NTP Secondary	Should be installed on the standby coordinator host
Segment host (1..N)	ADB	ADB Segment	Should be installed on each segment host
	Chrony	NTP Slave	Should be installed on each segment host
	PXF	PXF	Should be installed on one or more segment hosts and, optionally, on the coordinator host

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


Clusters
24 Nov 2025 19:29:10 UTC
Clusters / Test ADB cluster / Mapping / Components view
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
ADB
ADB Master 1 / 1
bds-mdw
ADB Segment 2 / 1
bds-sdw1
bds-sdw2
ADB Standby
bds-smdw
Chrony
NTP Master
NTP Secondary
bds-smdw
NTP Slave 2 / 0
bds-sdw1
bds-sdw2
PXF
PXF 2 / 1
bds-sdw1
bds-sdw2
Add hosts

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.


Clusters
24 Nov 2025 19:29:36 UTC
Clusters / Test ADB cluster / Mapping / Hosts view
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hosts
Search components
A - Z order
Hide empty
Reset
Save
Hosts mode
1
bds-mdw 2
2
Add components
ADB Master
NTP Master
bds-sdw1 3
ADB Segment
NTP Slave
PXF
bds-sdw2 3
bds-smdw 2
ADB Standby
NTP Secondary
List of Restrictions

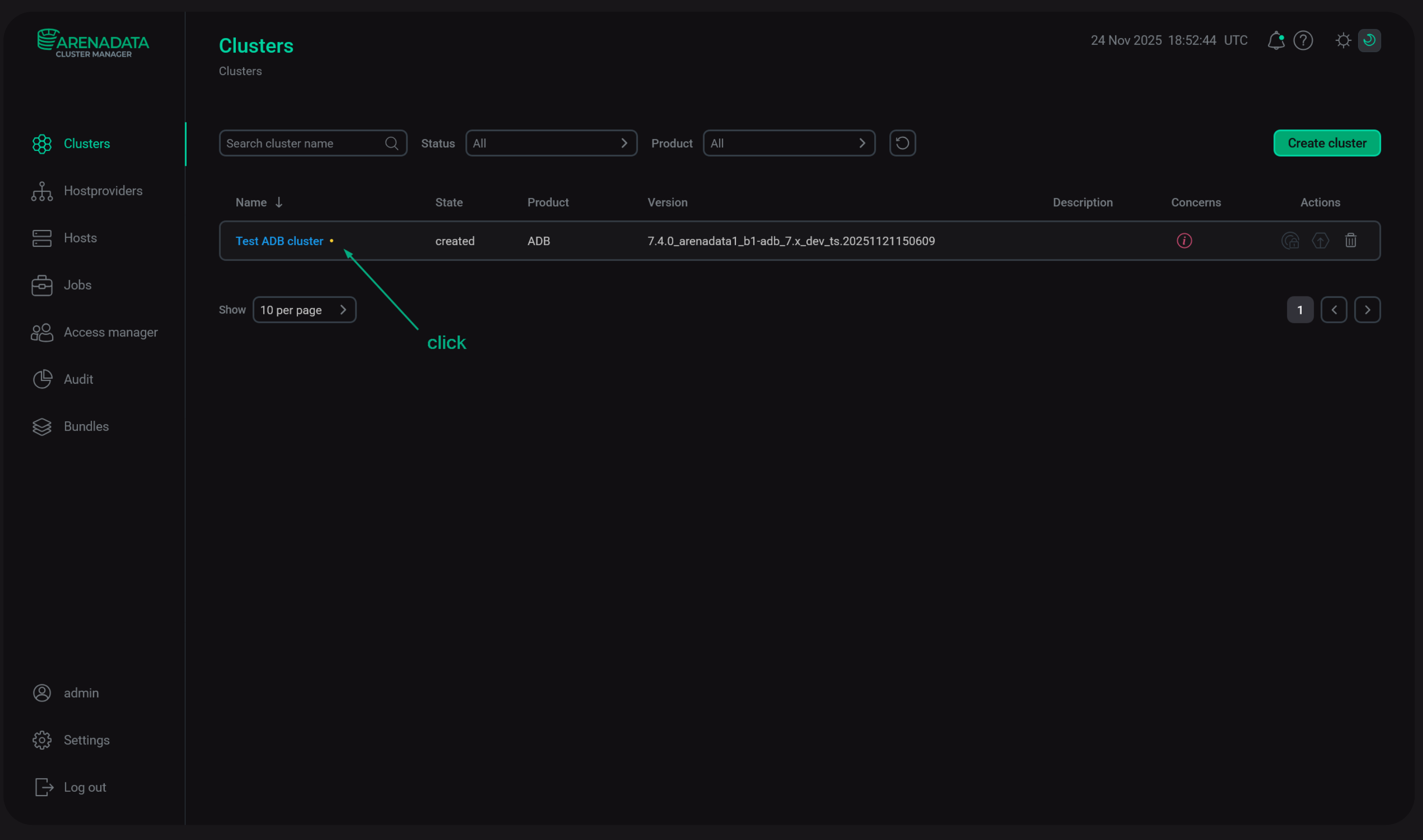
Use mapping by hosts

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.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
24 Nov 2025 18:52:44 UTC
Search cluster name
Status All
Product All
Create cluster
click
Show 10 per page

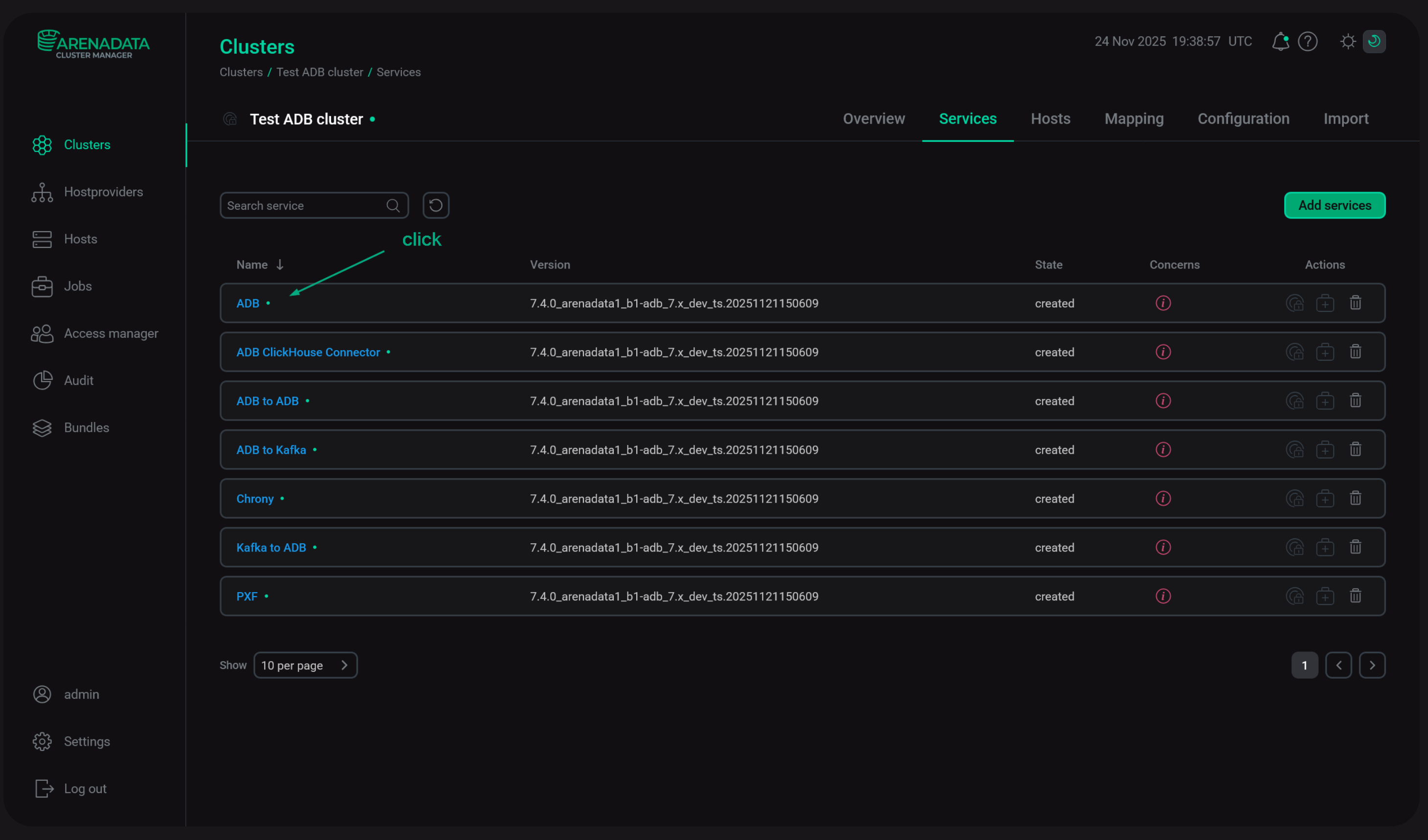
Clusters

Clusters

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

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.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
24 Nov 2025 19:38:57 UTC
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services
click
Show 10 per page

Clusters

Clusters / Test ADB cluster / Services

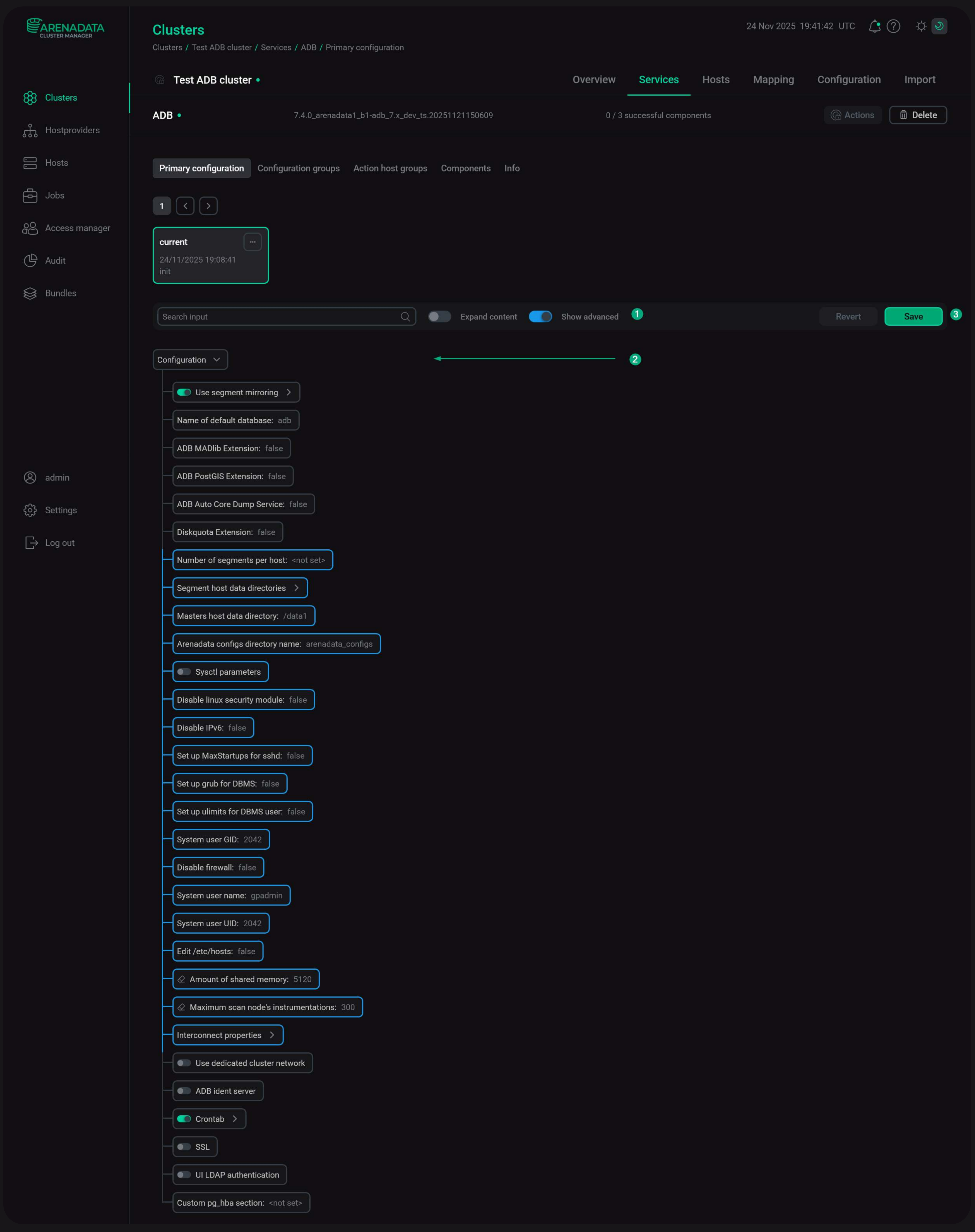
Test ADB cluster

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	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***.


ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
24 Nov 2025 19:41:42 UTC
Overview
Services
Hosts
Mapping
Configuration
Import
0 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
Search input
Expand content
Show advanced
①
Revert
Save
③
②
Configuration

Clusters

Clusters / Test ADB cluster / Services / ADB / Primary configuration

Test ADB cluster

ADB 7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609

current
24/11/2025 19:08:41
init

- ● Use segment mirroring
- Name of default database: adb
- ADB MADlib Extension: false
- ADB PostGIS Extension: false
- ADB Auto Core Dump Service: false
- Diskquota Extension: false
- Number of segments per host: <not set>
- Segment host data directories
- Masters host data directory: /data1
- Arenadata configs directory name: arenadata_configs
- ○ Sysctl parameters
- Disable linux security module: false
- Disable IPv6: false
- Set up MaxStartups for sshd: false
- Set up grub for DBMS: false
- Set up ulimits for DBMS user: false
- System user GID: 2042
- Disable firewall: false
- System user name: gpadmin
- System user UID: 2042
- Edit /etc/hosts: false
- Amount of shared memory: 5120
- Maximum scan node's instrumentations: 300
- Interconnect properties
- ○ Use dedicated cluster network
- ○ ADB ident server
- ● Crontab
- ○ SSL
- ○ UI LDAP authentication
- Custom pg_hba section: <not set>

Configure a service

   Pay attention to the following parameters:

   - ***ADB*** **→** ***Edit /etc/hosts***. Set to `true` to let ADB modify the */etc/hosts* file to include hostnames of all cluster hosts. Otherwise, you should modify this file manually or configure a DNS server. Using the ADB service settings, you can also adjust other parameters of the cluster hosts to align them with recommended values.
   - ***ADB*** **→** ***Disable linux security module*** and ***ADB*** **→** ***Set up GRUB for DBMS***. Changing these parameters requires host reboot, so it is recommended to set them to `true` before cluster installation and then set ***Reboot cluster servers after installation*** to `true` when launching the Install cluster action to reboot hosts automatically. If you change these parameters after installation, you will have to stop the cluster and reboot hosts manually.
   - ***Chrony*** **→** ***NTP servers***. Though this parameter is optional, we recommend to set it to the nearest NTP server, e.g. the one that your company uses.

NOTE

For more information about services configuration parameters, see Configuration parameters.

Configure a cluster

Daria Barysheva

Contents

To Table of Contents

Primary configuration

Configuration groups

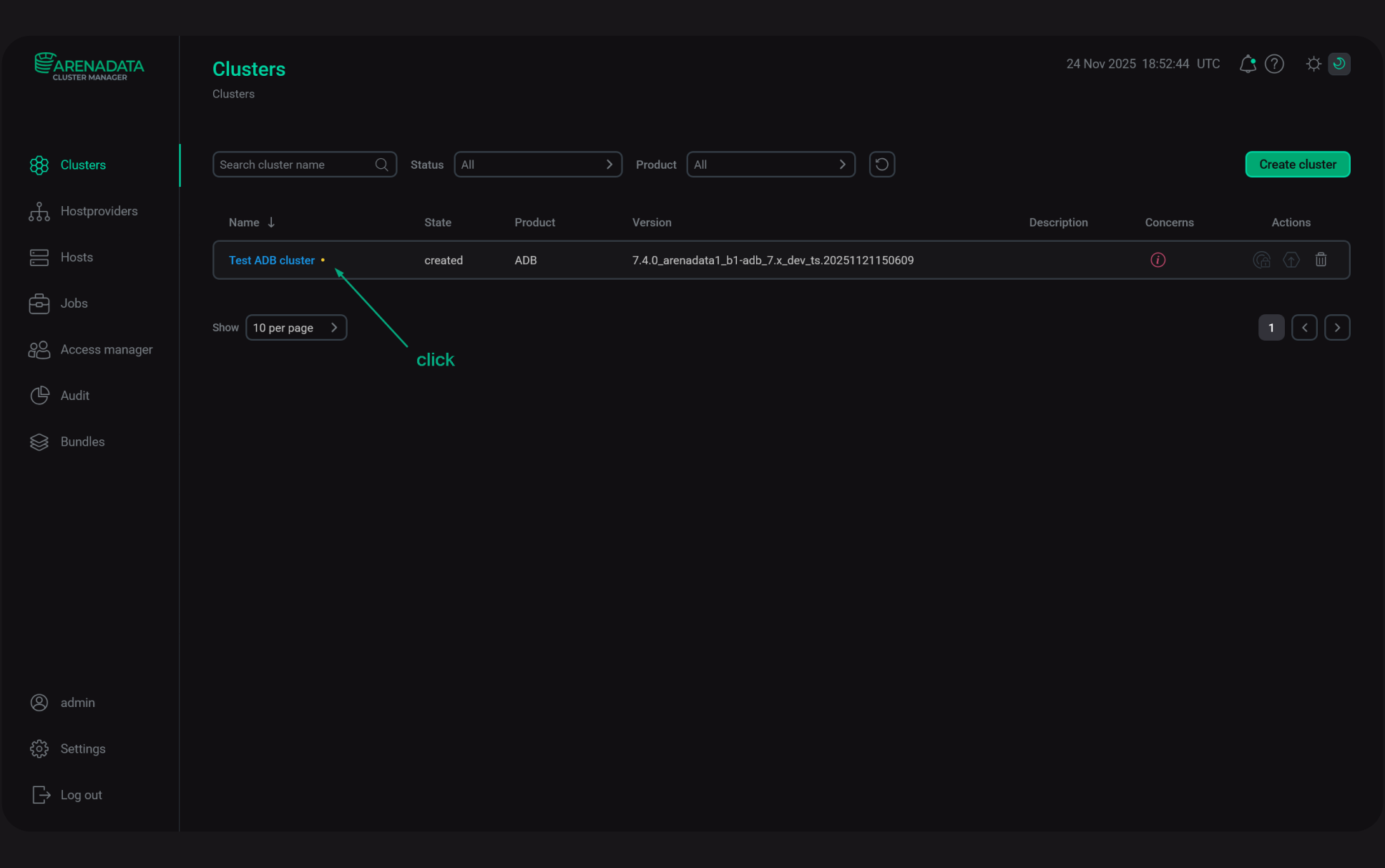
Ansible settings

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.



Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

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.

CAUTION

Use all custom installations (***Use existing docker***, ***Use custom registry***, ***Use custom JAVA_HOME for cluster***) on your own risk.

Available configuration parameters are listed below:

- ***Cluster UUID*** — the unique cluster identifier, which is generated automatically during ADB installation.

- ***Timeout to unlock package manager*** — parameters of this section are intended to set the timeout of the DNF software package manager unlocking wait. In Ubuntu, after the host reboot, automatic update starts, which locks DNF.

Name	Description	Default
retries	The maximum number of retries	3
delay	The time delay between retries (in seconds)	10

- Paths to the repositories that are used during the installation process depending on the cluster operating system (AltLinux, Astra Linux, RED OS, and Ubuntu). Only the marked repositories are registered on the cluster hosts.

Altlinux 10 repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

Astra linux repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

redos 7 repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

Ubuntu 22.04 repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

- ***Use existing docker*** — a flag that indicates whether to use the existing Docker during the installation of ADB services (e.g. ADB Control, ADBM, Monitoring, Enterprise Tools). If this flag is set, only containers will be deployed, while Docker and Docker Compose installation will be skipped. For more information on Docker requirements, see Software requirements.

IMPORTANT

Starting with ADB 6.27.1.56, ADB Control rewrites Docker settings in the */etc/docker/daemon.json* file for security purposes. Thus, if you need to modify this file, set the ***Use existing docker*** option when configuring ADB. This option will also allow you to use Docker in the rootless mode.

- ***Use custom registry*** — a switcher that determines whether to use a custom registry. When the switcher is activated, the following parameters are available.

Parameter	Description
registry	A custom registry URL in the following form: `<host>:<port>`
prefix	An optional subdirectory name that will be added to the custom registry URL: `<registry>/<prefix>`
Use self-signed certificate	Whether or not to use a self-signed certificate
Self-signed certificate	A self-signed certificate text

- ***Use custom JAVA_HOME for cluster*** — a switcher that indicates whether to use a custom Java path for ADB services (e.g. PXF, ADB Control, ADBM). Displayed if the ***Show advanced*** flag is set. When the switcher is activated, the ***JAVA_HOME*** field becomes available. Enter a work Java path in that field. The selected path should contain a *bin* folder with a Java binary file, for example */usr/lib/jvm/java-17-openjdk-amd64*. Installation will fail if there is no Java at the specified path. For more information on Java requirements, see Software requirements.
  If the ***JAVA_HOME*** value is empty, one of the following default paths is used (depending on the operating system).

Ubuntu

/usr/lib/jvm/java-17-openjdk-amd64

RED OS

/usr/lib/jvm/jre-17-openjdk

AltLinux

/usr/lib/jvm/jre-17

IMPORTANT

- If you plan to use custom Java (such as Axiom), Java 17 is required for installation.
- For Astra Linux, the ***JAVA_HOME*** field is mandatory starting with ADB 6.27.1.58.

- ***Parameter archiving*** — a switcher that indicates whether to export ADB settings according to the specified schedule. When the switcher is activated, the ***Parameter archiving job setting*** field becomes available. Enter a schedule for an export job in that field. The field value should be a valid `cron` expression. After you edit and save the ***Parameter archiving job setting*** value, run the cluster action Reconfigure parameter archiving.
  Each time a `cron` job is run, ADB settings are exported to the following directory:

```
/home/gpadmin/arenadata_configs/settings/<timestamp>
```

where `<timestamp>` is a timestamp of the export job launch in the `YYYY-MM-DD_HH-mm-ss` format.

A job execution log is saved to the *parameter_archiving.log* file.

Currently, the following settings are offloaded.

Service	Exported parameters	Relative path
ADB	GUC values for coordinator/segment	./<YYYY-MM-DD_HH-mm-ss>_guc_config.json
	Linux Greenplum-related parameters that are exported from */etc/sysctl.conf*	./<YYYY-MM-DD_HH-mm-ss>_os_config.json
	Resource group parameters	./<YYYY-MM-DD_HH-mm-ss>_rg_config.json
ADB Control	Configuration parameters of ADB Control components	./adcc/*
ADBM	Configuration parameters of ADBM components	./adbm/*
PXF	PXF settings that are exported from: */etc/pxf/conf/pxf-application.properties* */etc/pxf/conf/pxf-profiles.xml* *$PXF_BASE/servers/**	./pxf_config/*

- ***Vault integration*** — a switcher that indicates whether to use HashiCorp Vault to store secrets of ADB services. When the switcher is activated, the following parameters become available.

Name	Description	Default
Address of the Vault server	The Vault server address (e.g. `192.168.1.120` or `vault.domain.com`)	—
Port of the Vault server	The Vault server port. Should be specified if the service is published on a port other than HTTPS (443)	443
Internal CA certificate in PEM format (optional)	The internal CA certificate in PEM format that signed the Vault server's certificate	—
Use PEM CA for vault	Defines whether to use the CA certificate in PEM format. If you set this parameter to `true`, you also need to fill in ***Internal CA certificate in PEM format (optional)***. To hide the truststore password in Vault, set ***Use PEM CA for vault*** to `true`	false
Vault namespace (optional)	The Vault namespace	—

Writer authentication method for the Vault server	The authentication method that is used to connect to the Vault server with write access.
	Possible values:
	`token`
	`approle`
		approle

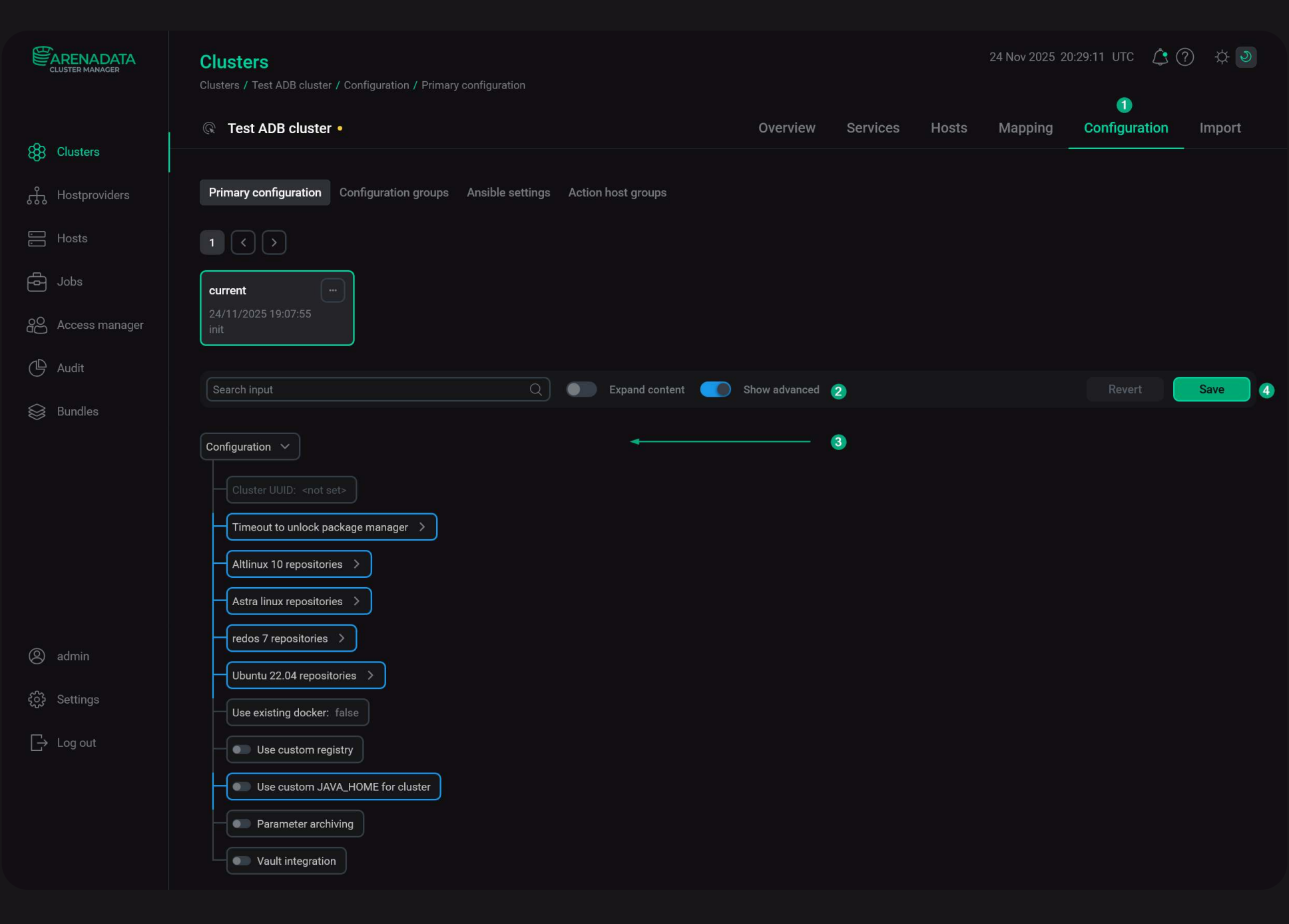
Writer Vault token	The application token that is used to connect to the Vault server with write access. Required only if the ***Writer authentication method for the Vault server*** parameter is set to `token`	—
Writer Vault role ID	The role identifier provided for applications to connect to the Vault server with write access. Required only if the ***Writer authentication method for the Vault server*** parameter is set to `approle`	—
Writer Vault secret ID	The secret identifier associated with the ***Writer Vault role ID*** role that is used by applications to connect to the Vault server with write access. Required only if the ***Writer authentication method for the Vault server*** parameter is set to `approle`	—

Reader authentication method for the Vault server	The authentication method that is used to connect to the Vault server with read access.
	Possible values:
	`token`
	`approle`
		approle

Reader Vault token	The application token that is used to connect to the Vault server with read access. Required only if the ***Reader authentication method for the Vault server*** parameter is set to `token`	—
Reader Vault role ID	The role identifier provided for applications to connect to the Vault server with read access. Required only if the ***Reader authentication method for the Vault server*** parameter is set to `approle`	—
Reader Vault secret ID	The secret identifier associated with the ***Reader Vault role ID*** role that is used by applications to connect to the Vault server with read access. Required only if the ***Reader authentication method for the Vault server*** parameter is set to `approle`	—
Mount point for ADB cluster secrets	A mount point that is accessible for read/write operations with the provided ADB cluster authentication parameters. Each service will store secrets on the Vault server at the following path: *<Mount point for ADB cluster secrets>/adb/<cluster_name>/service/<service_name>/<component>*	—
Login mount point	A custom authentication mount point. If empty, equals to the authentication method selected in the ***Writer authentication method for the Vault server*** (write) or ***Reader authentication method for the Vault server*** (read) field	—

IMPORTANT

- Before editing Vault integration parameters, ensure that Vault is installed and configured.
- Each time you edit and save parameters in the ***Vault integration*** section, run the cluster action Reconfigure Vault integration.
- Currently, Vault can be used to store authentication parameters of ADB Control and ADBM.
- If you need to store credentials for the ADB Control Query DB and Metrics DB in Vault, set up these databases as external.



Cluster configuration window

Configuration groups

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

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

Daria Barysheva

Contents

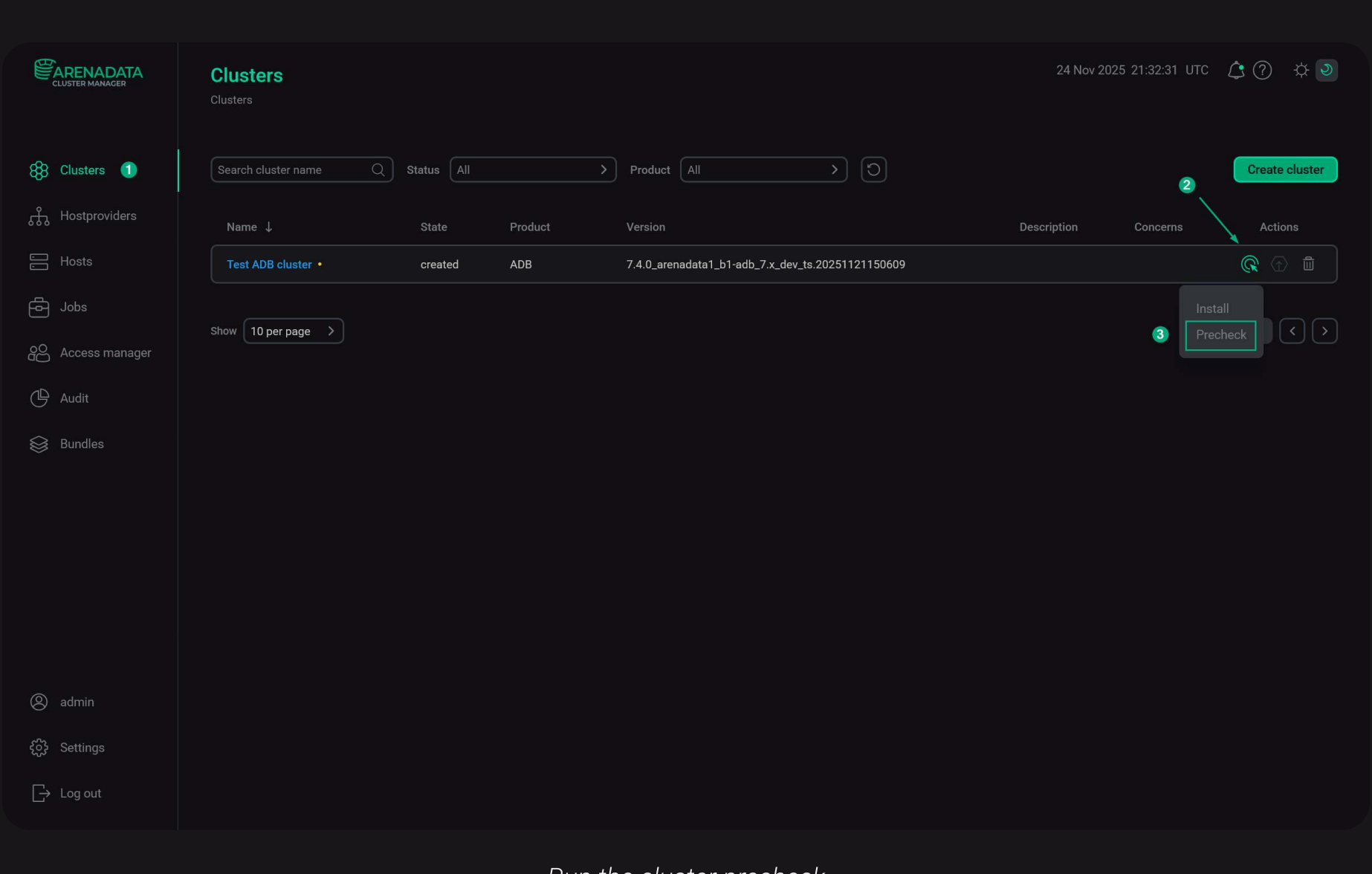
To Table of Contents

Step 1. Precheck
Step 2. Run installation
  Whole cluster
  Single services
Step 3. Check the cluster state after installation
Step 4. Check the default database

Step 1. Precheck

Before installing a new cluster it is recommended to run the ***Precheck*** action. By doing this, you make sure that all necessary repositories are available, the cluster does not contain hosts with the same IP addresses, and all other prerequisites are met. Do the following:

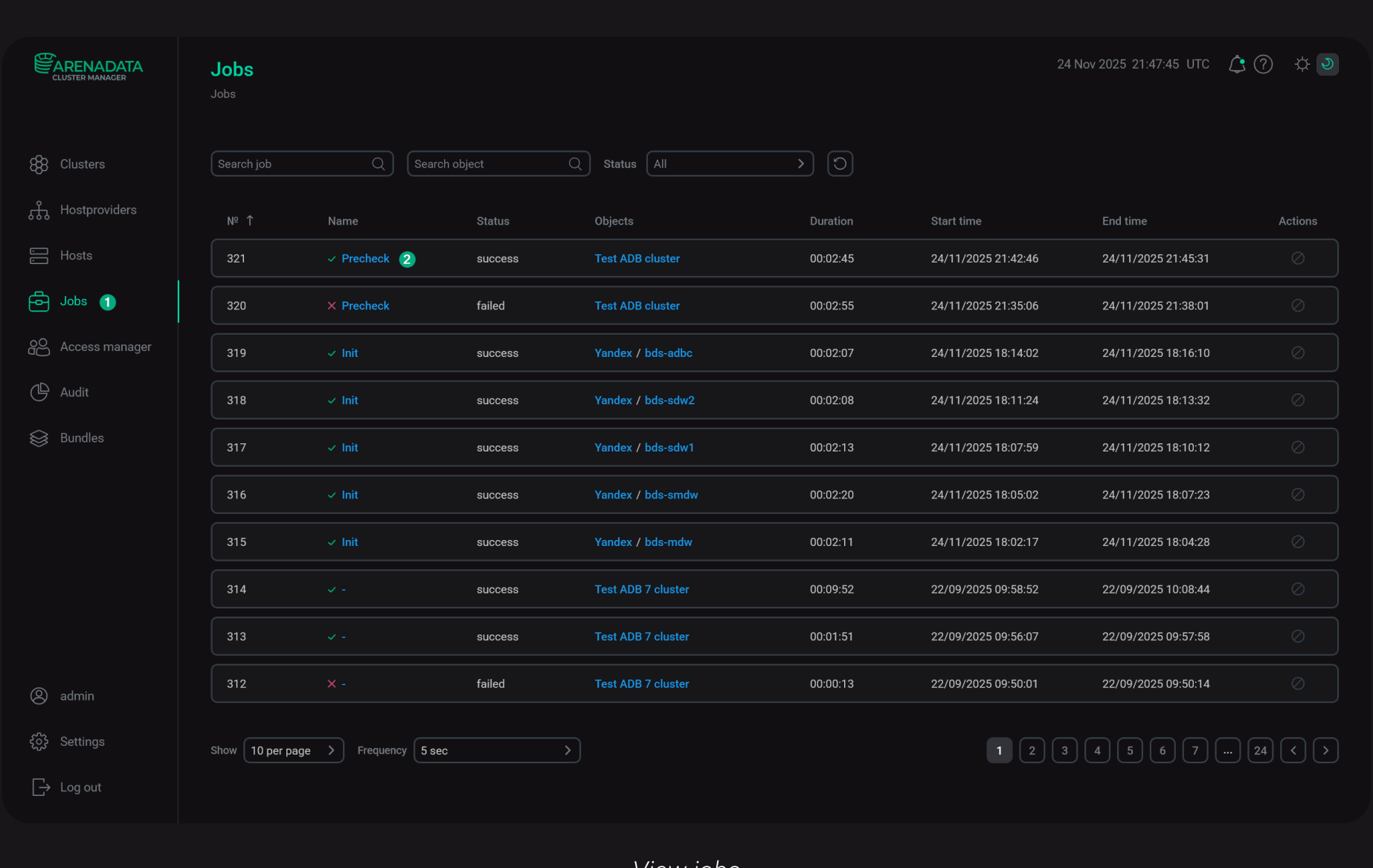
1. Apply the ***Precheck*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.



Name	State	Product	Version
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609

Run the cluster precheck

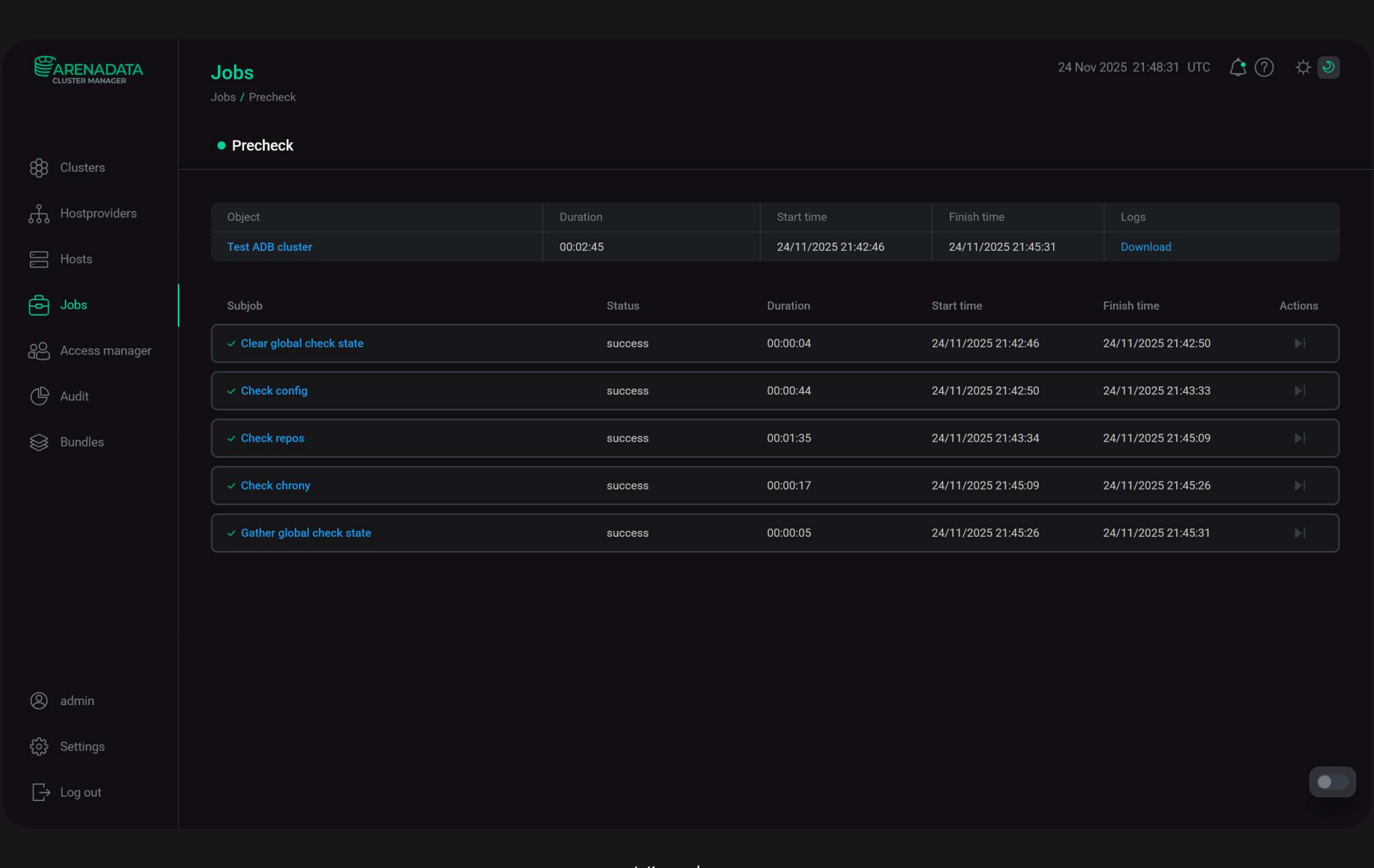
2. Confirm the requested action in the window that opens.
3. Open the ***Jobs*** page and wait until the running job is completed. After this, click the first item in the job list.



№	Name	Status	Objects	Duration	Start time	End time
321	Precheck	success	Test ADB cluster	00:02:45	24/11/2025 21:42:46	24/11/2025 21:45:31
320	Precheck	failed	Test ADB cluster	00:02:55	24/11/2025 21:35:06	24/11/2025 21:38:01
319	Init	success	Yandex / bds-adbc	00:02:07	24/11/2025 18:14:02	24/11/2025 18:16:10
318	Init	success	Yandex / bds-sdw2	00:02:08	24/11/2025 18:11:24	24/11/2025 18:13:32
317	Init	success	Yandex / bds-sdw1	00:02:13	24/11/2025 18:07:59	24/11/2025 18:10:12
316	Init	success	Yandex / bds-smdw	00:02:20	24/11/2025 18:05:02	24/11/2025 18:07:23
315	Init	success	Yandex / bds-mdw	00:02:11	24/11/2025 18:02:17	24/11/2025 18:04:28
314	-	success	Test ADB 7 cluster	00:09:52	22/09/2025 09:58:52	22/09/2025 10:08:44
313	-	success	Test ADB 7 cluster	00:01:51	22/09/2025 09:56:07	22/09/2025 09:57:58
312	-	failed	Test ADB 7 cluster	00:00:13	22/09/2025 09:50:01	22/09/2025 09:50:14

View jobs

4. View logs in the window that opens. To download logs, click ***Download***.



Object	Duration	Start time	Finish time	Logs
Test ADB cluster	00:02:45	24/11/2025 21:42:46	24/11/2025 21:45:31	Download

Subjob	Status	Duration	Start time	Finish time
Clear global check state	success	00:00:04	24/11/2025 21:42:46	24/11/2025 21:42:50
Check config	success	00:00:44	24/11/2025 21:42:50	24/11/2025 21:43:33
Check repos	success	00:01:35	24/11/2025 21:43:34	24/11/2025 21:45:09
Check chrony	success	00:00:17	24/11/2025 21:45:09	24/11/2025 21:45:26
Gather global check state	success	00:00:05	24/11/2025 21:45:26	24/11/2025 21:45:31

View logs

If the logs contain errors related to host settings (for example, regarding IPv6 or sysctl parameters), make sure the cluster hosts meet operating system prerequisites.

Step 2. Run 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.

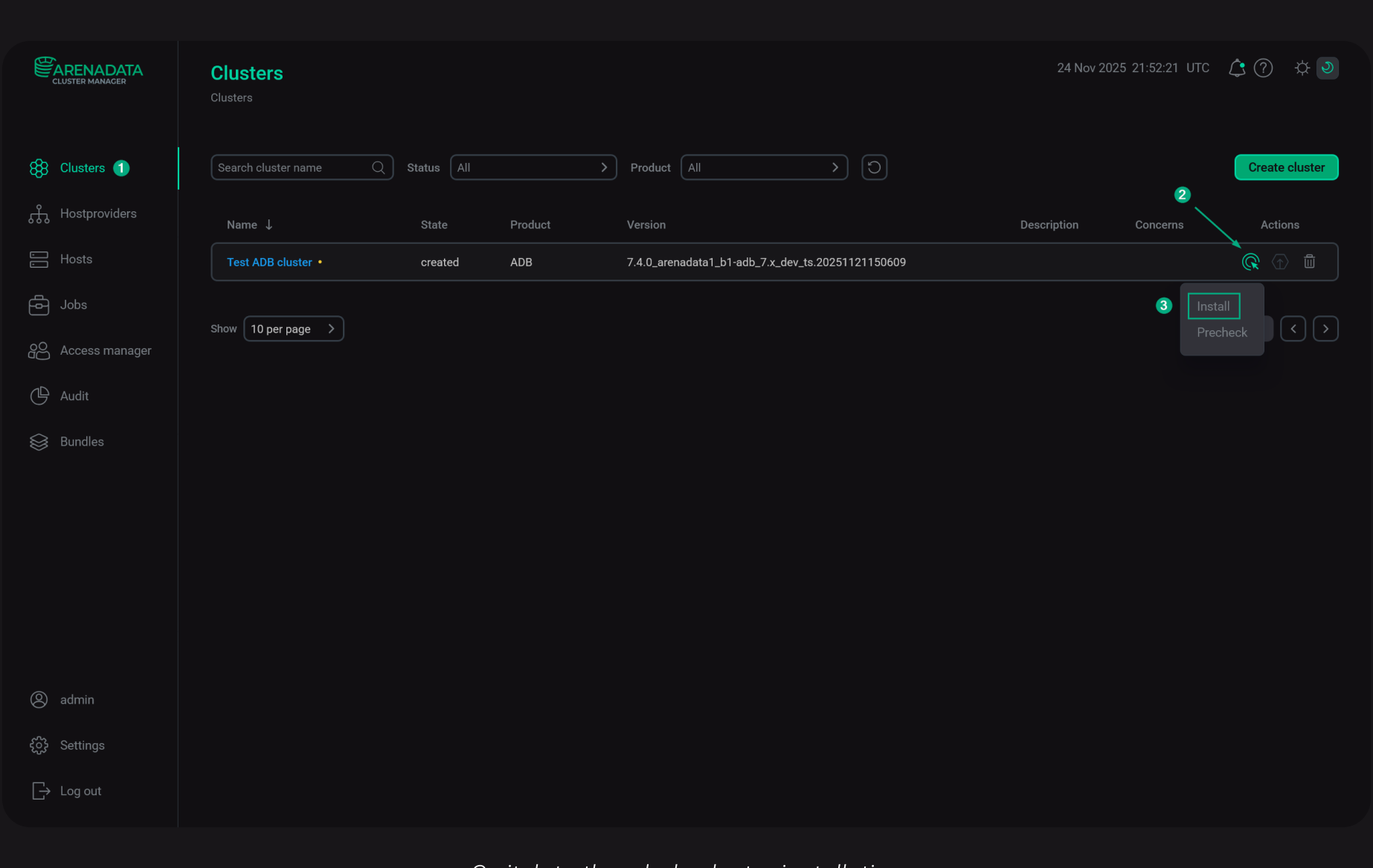
In both cases each service starts automatically after installation.

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

Whole cluster

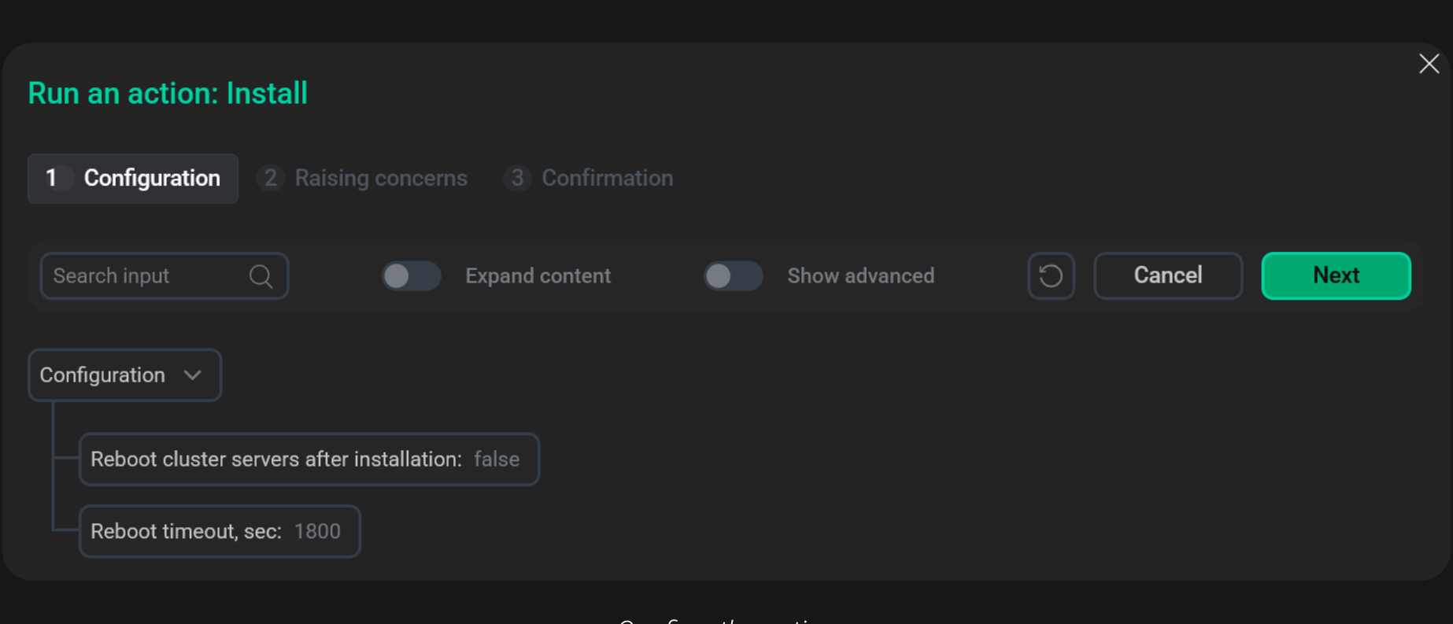
To install all cluster services within a single action, follow the steps:

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



Switch to the whole cluster installation

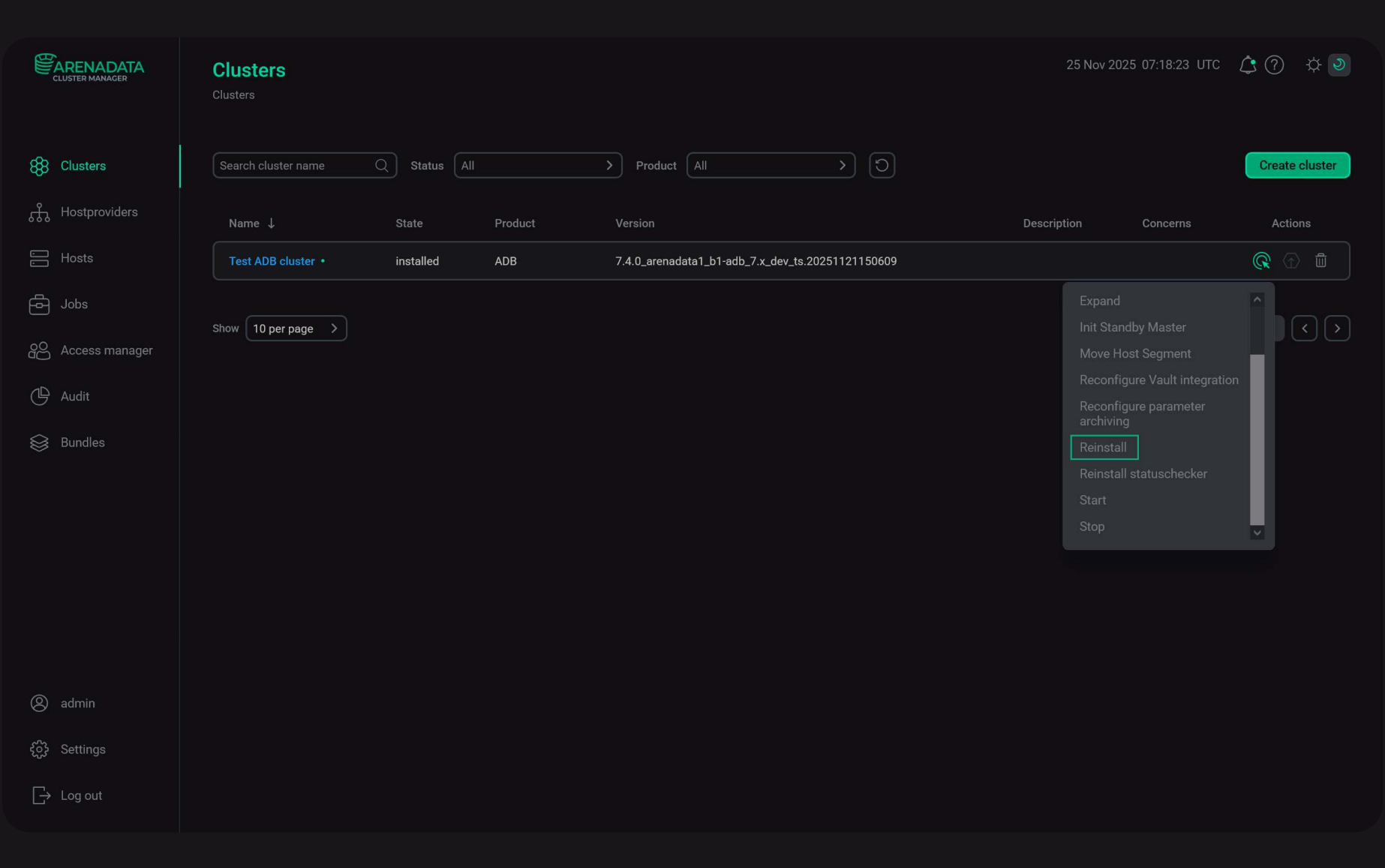
2. Verify the requested action in the opened window. Notice that setting the ***Reboot cluster servers after installation*** flag leads to the hosts being rebooted after the installation is completed. Otherwise, the reboot should be done manually.



Configuration
Reboot cluster servers after installation: false
Reboot timeout, sec: 1800

Confirm the action

3. Wait until the job is completed.
4. If some service fails during installation, but the preceding services are installed successfully, you can try to fix errors and restart installation of all failed services. To do this, select the ***Reinstall*** action in the ***Actions*** column on the ***Clusters*** page.



Installation restart

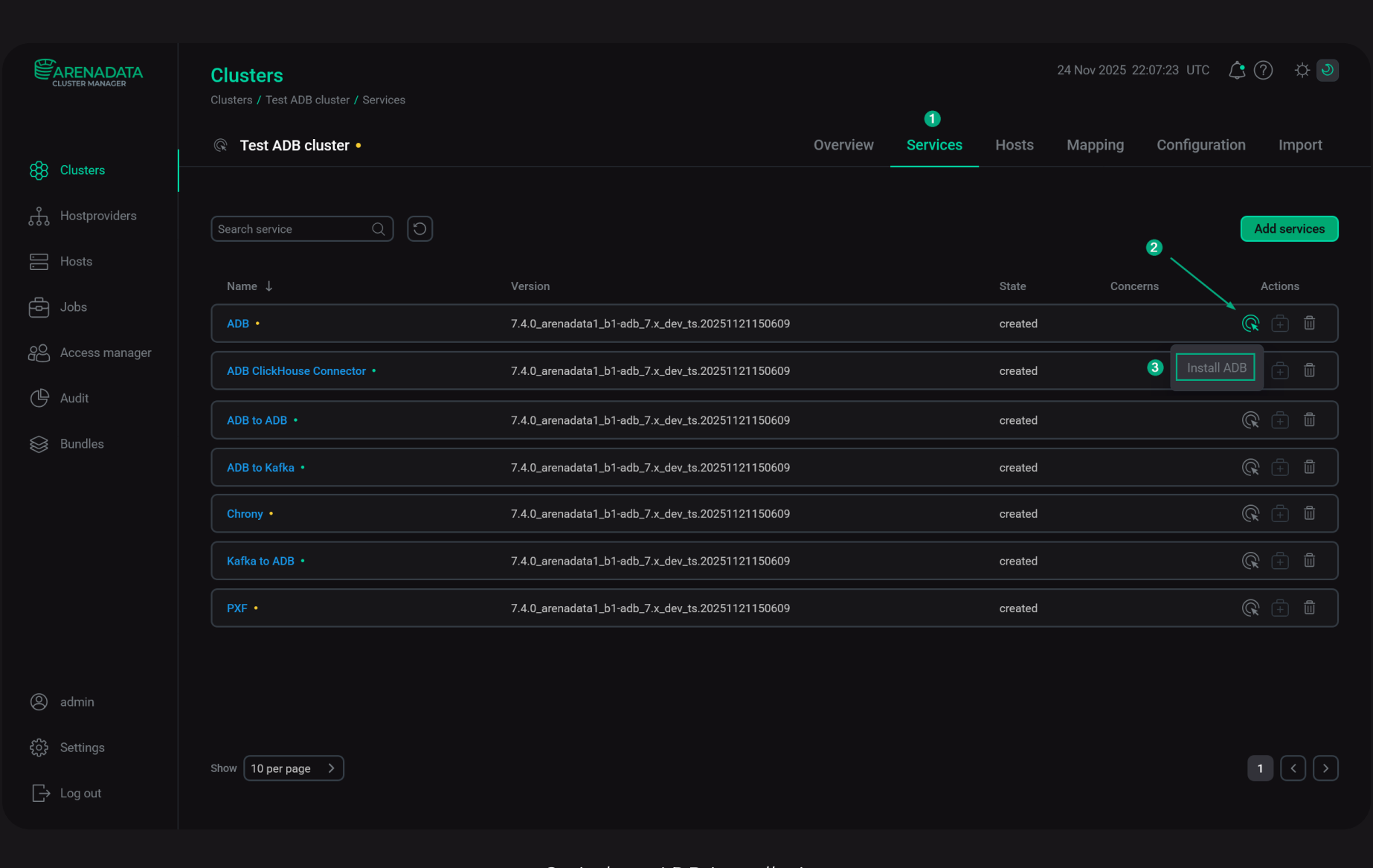
Single services

To install one or several cluster services manually, add these services to the cluster and then perform the actions described in the following subsections. The process of installing the ADB service differs from all other services.

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

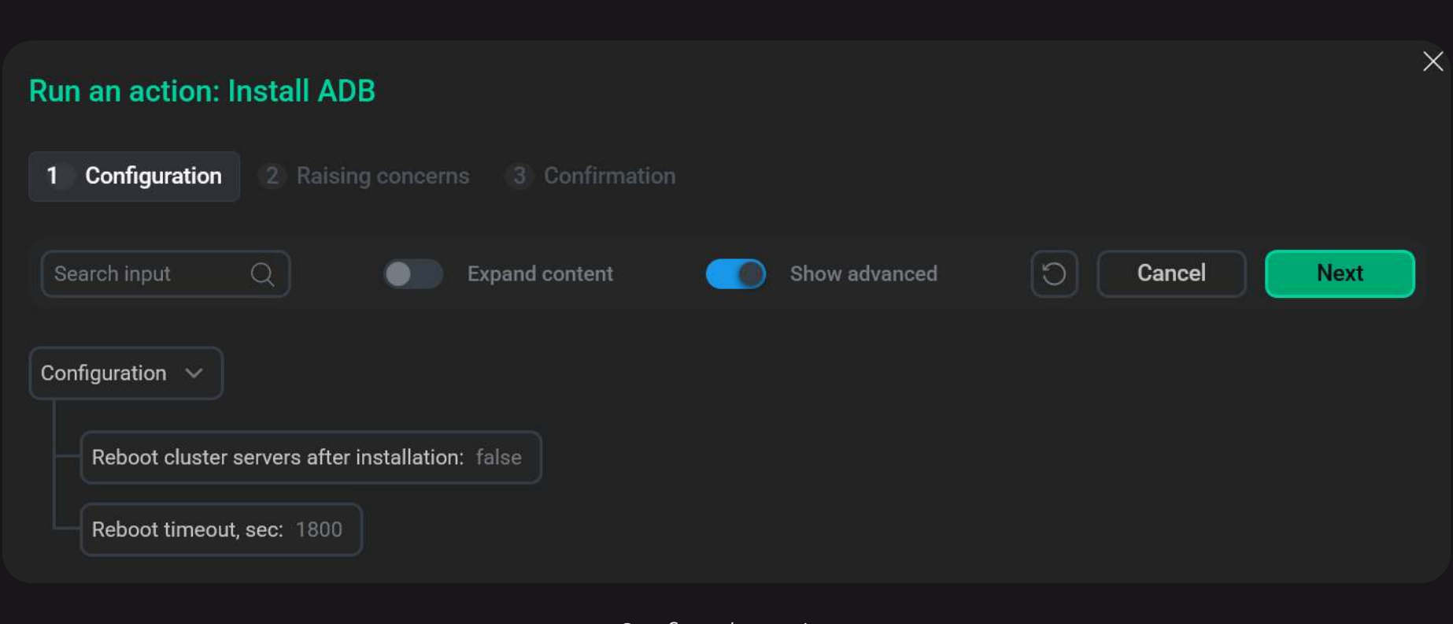
ADB service

1. Open the ***Services*** tab on the cluster page. Find the row corresponding to the ADB service, click the icon in the ***Actions*** column, and select the ***Install ADB*** action.



Switch to ADB installation

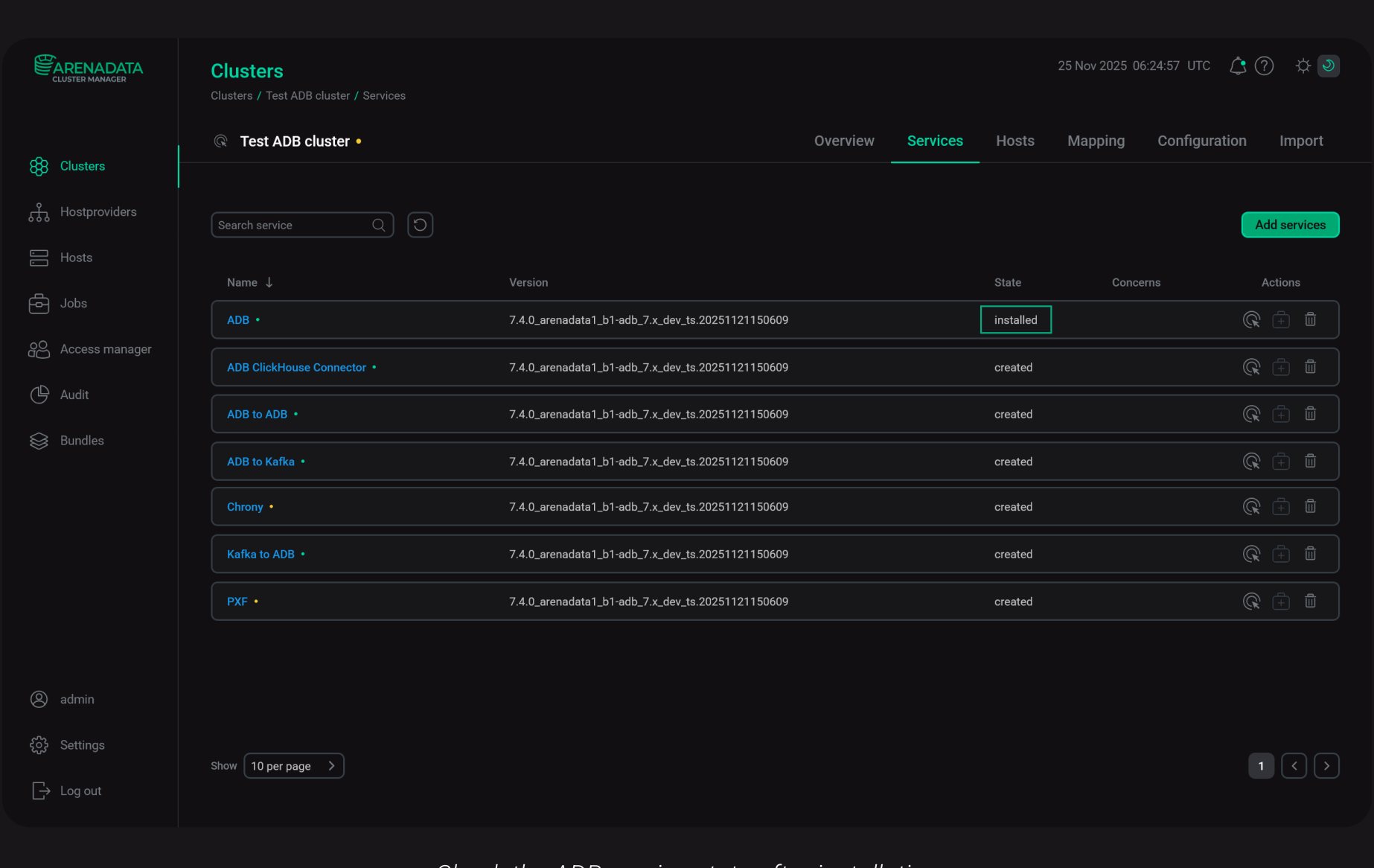
2. Confirm the action in the window that opens.



Configuration
Reboot cluster servers after installation: false
Reboot timeout, sec: 1800

Confirm the action

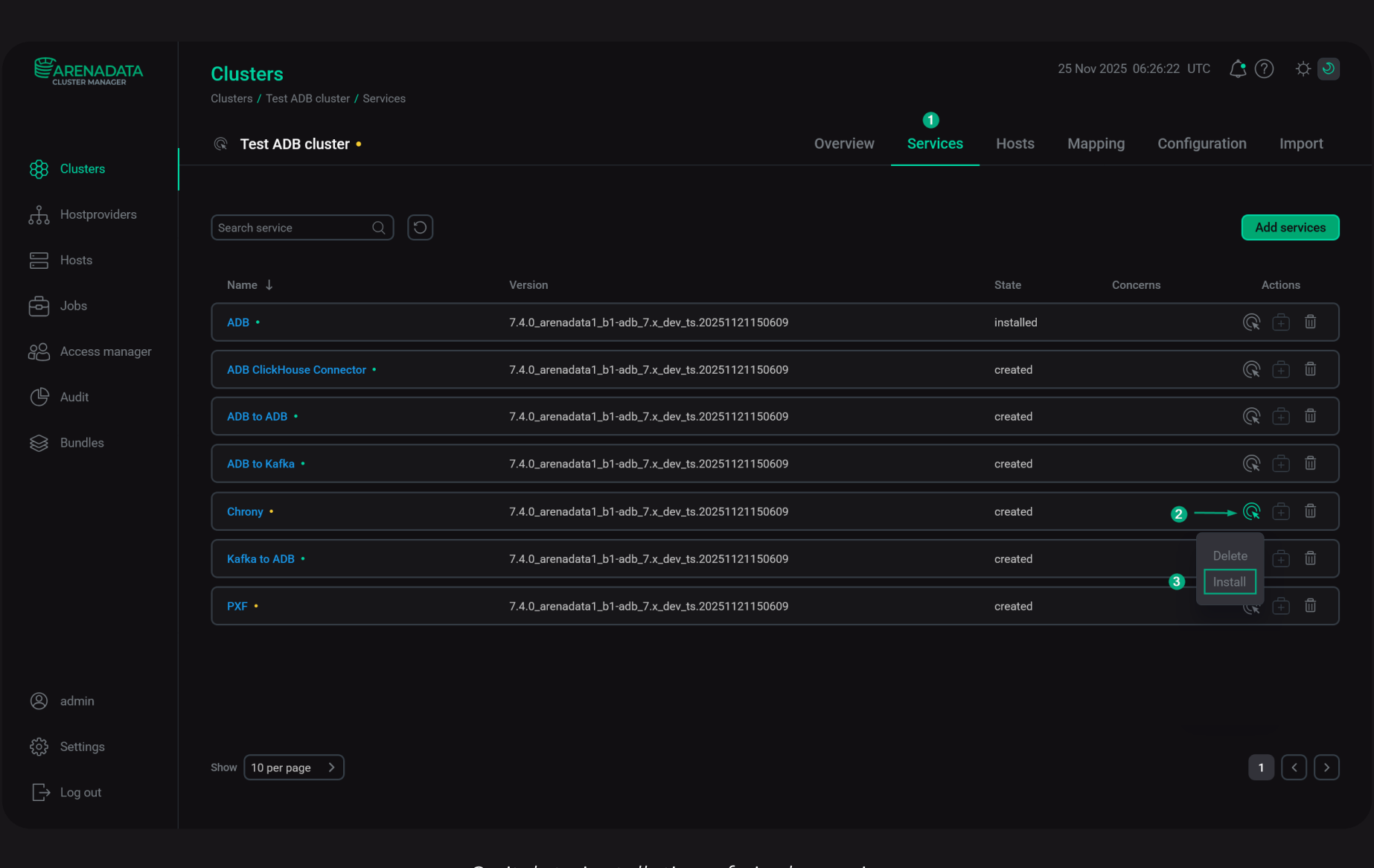
3. Wait until the job is completed. Then check that the ADB service state is changed from `created` to `installed`.



Check the ADB service state after installation

Other services

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.



Switch to installation of single services

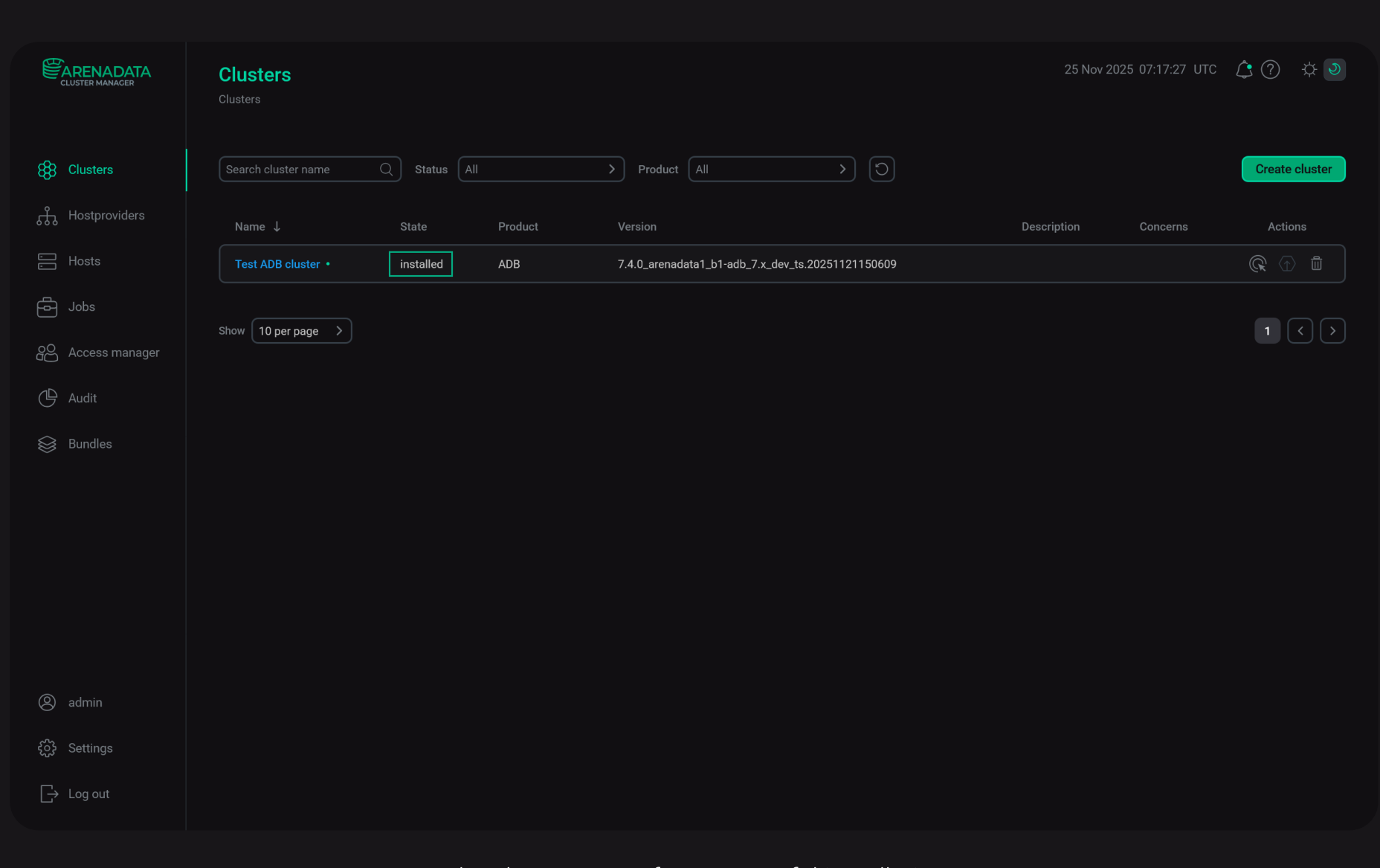
2. Confirm the action in the window that opens.
3. Wait until the job is completed. Then check that the service state has changed from `created` to a new value that is shown in the New states of ADB services after installation table.
4. Repeat the previous actions for other services that should be installed.

> **IMPORTANT**
>
> When you install services manually, one by one, use the following order:
>
> 1. ADB
> 2. Chrony
> 3. ADB to ADB
> 4. PXF
> 5. ADB ClickHouse Connector
> 6. ADB to Kafka
> 7. Kafka to ADB

Step 3. Check the cluster state after 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`.



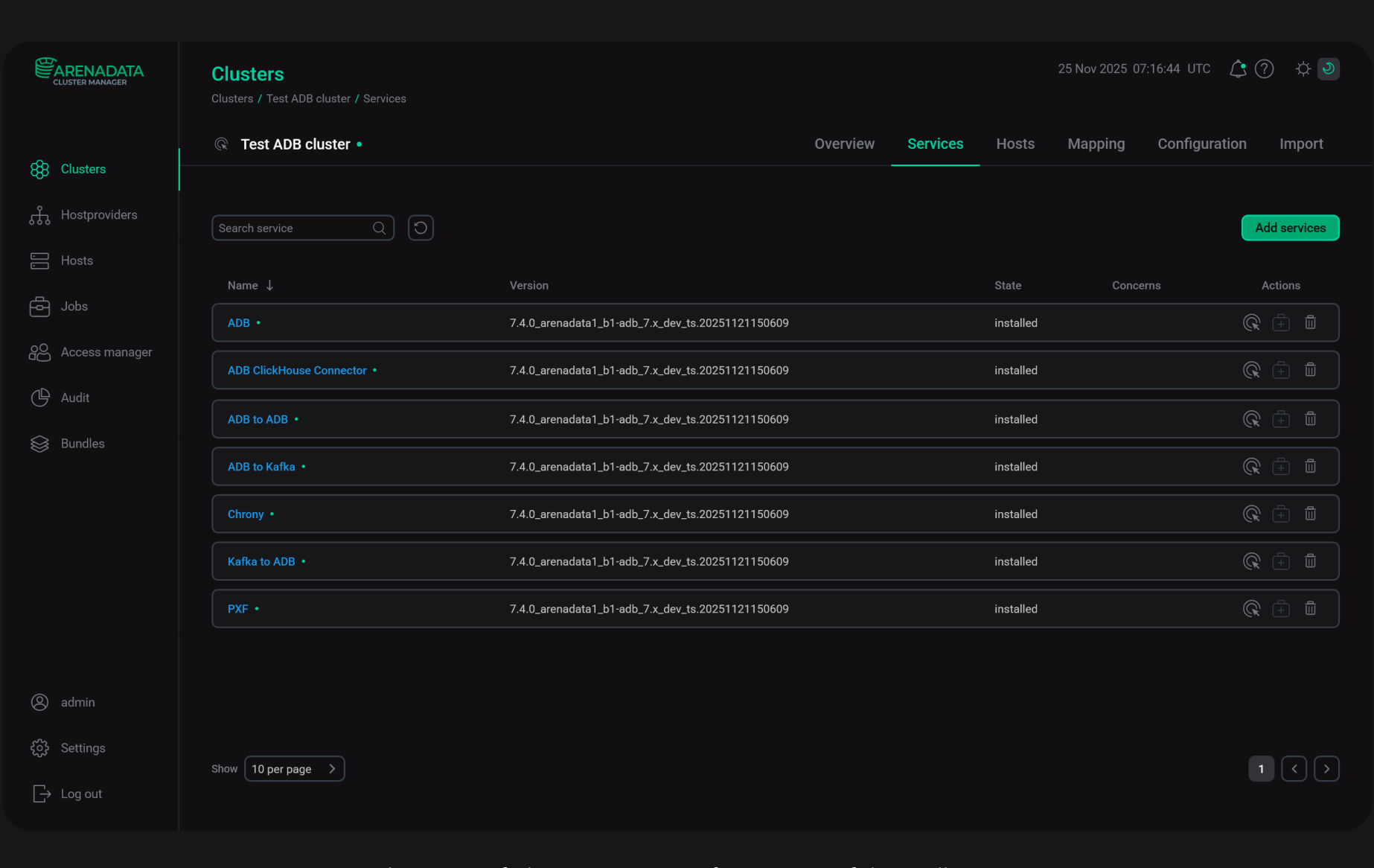
Name	State	Product	Version
Test ADB cluster	installed	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609

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 a new value that is shown in the table below.

New states of ADB services after installation

Service	New state
ADB	installed
Chrony	installed
PXF	installed
ADB ClickHouse Connector	installed
ADB to ADB	installed
ADB to Kafka	installed
Kafka to ADB	installed



Name	Version	State
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed

The state of cluster services after successful installation

Step 4. Check the default database

Regardless of the installation method, the default database is created during the ADB service installation. This database is used as default for client connections (for example, during the installation of extensions via ADB service actions). The database name can be specified in the ***Name of default database*** configuration parameter of the ADB service before installation. The default name is `adb`.

To ensure that the default database is successfully created, perform the following steps:

1. Connect to the ADB coordinator host (via SSH, etc.) and log in under the default user `gpadmin`.

```
$ sudo su - gpadmin
```

2. List all ADB databases.

```
psql -l
```

The command output contains the `adb` database:

```
                              List of databases
   Name    |  Owner  | Encoding |   Collate   |    Ctype    |  Access privileges
-----------+---------+----------+-------------+-------------+---------------------
 adb       | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/gpadmin        +
           |         |          |             |             | gpadmin=CTc/gpadmin
 postgres  | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/gpadmin         +
           |         |          |             |             | gpadmin=CTc/gpadmin
 template1 | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/gpadmin         +
           |         |          |             |             | gpadmin=CTc/gpadmin
(4 rows)
```

Offline installation

To Table of Contents

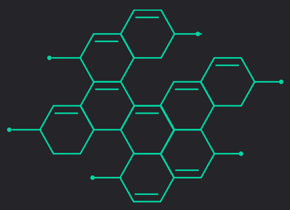
This topic describes **offline** installation of ADB. It can be applied to the hosts without access to Internet.

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 ADB.
- This is the only possible installation way for:
  - Arenadata DB Control (ADB Control) — a service that is responsible for monitoring, diagnostics, and queries management in the ADB clusters.
  - Arenadata DB Backup Manager (ADBM) — a service that is responsible for ADB binary backups management.



Step 1.

Install ADCM

Step 2.

Prepare hosts

Step 3.

Install ET cluster

Step 4.

Install ADB cluster

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
  - Online update
  - Offline update

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
```

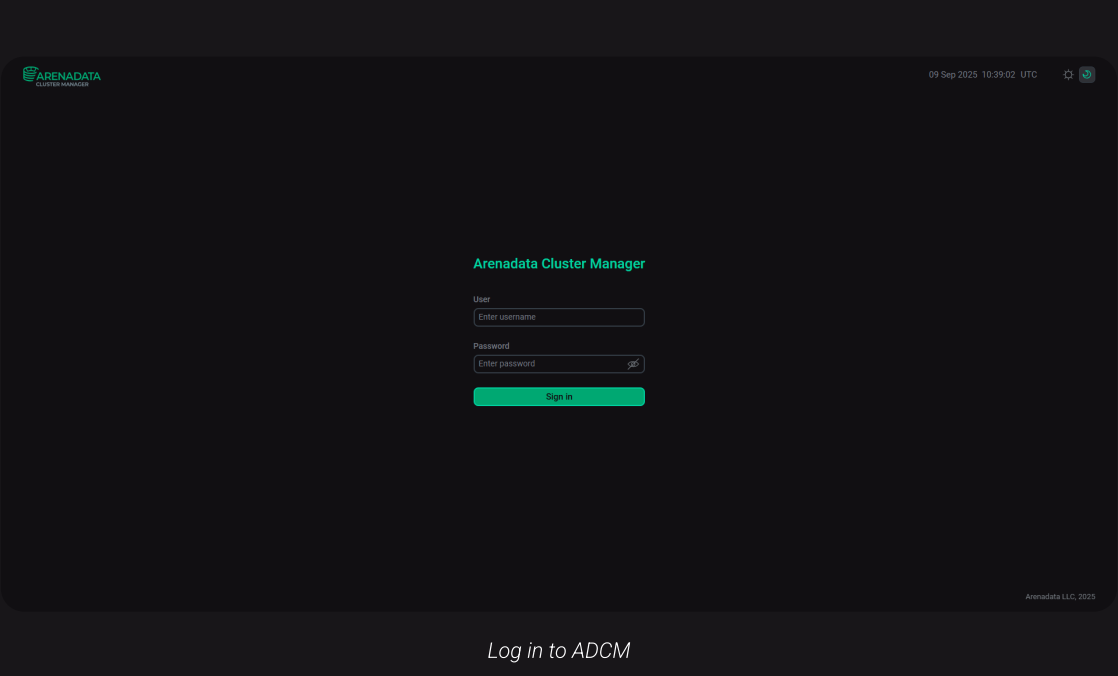
The example of the command execution result:

```
<!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:

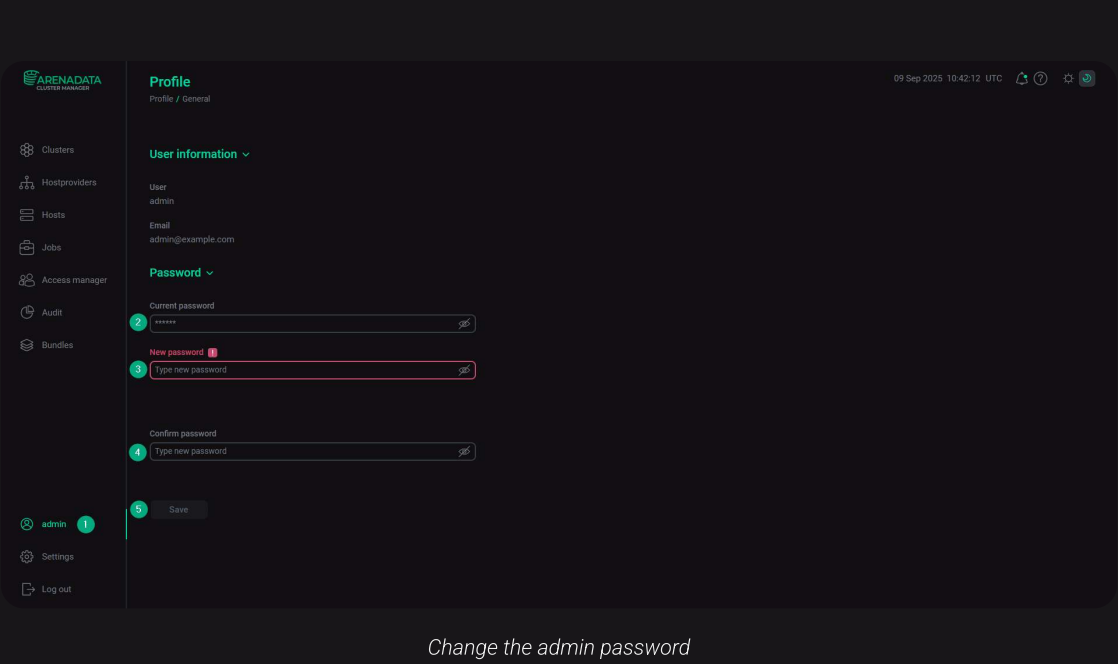
- ***User***: admin
- ***Password***: admin



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***.

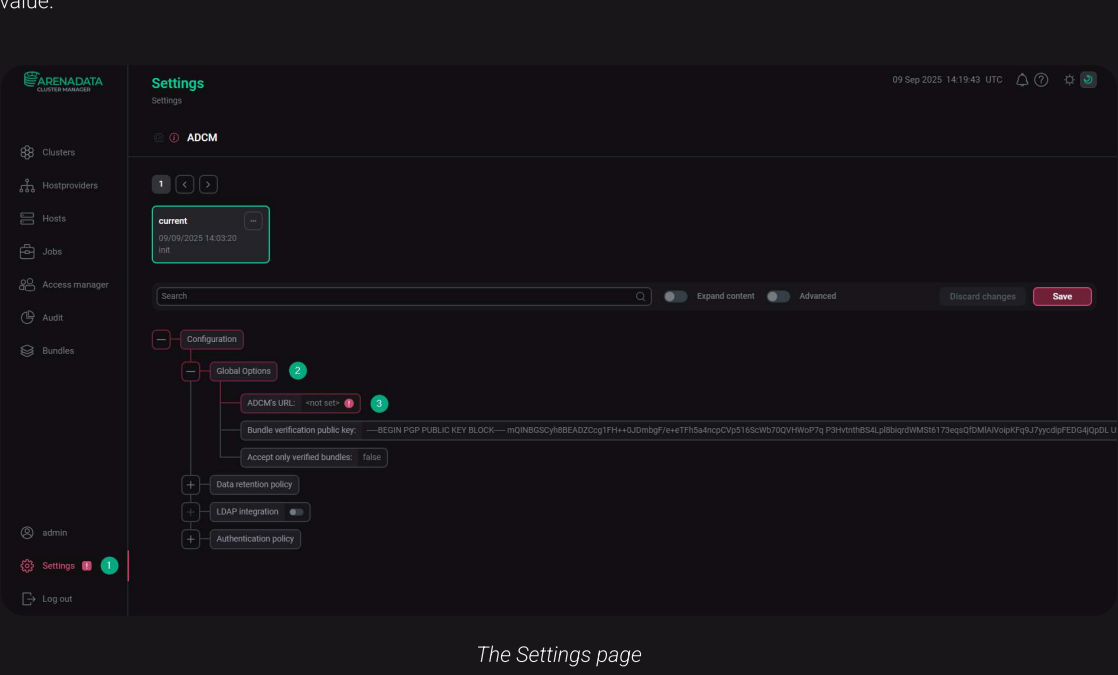


Change the admin password

Set the ADCM URL

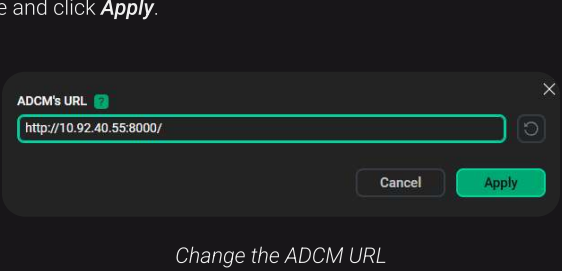
For successful communication between ADCM and product clusters deployed via ADCM, it is important to correctly define the external ADCM address. This address is used by cluster components for sending information about their state to ADCM. To specify the ADCM URL, use the environment variable `DEFAULT_ADCM_URL` 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.



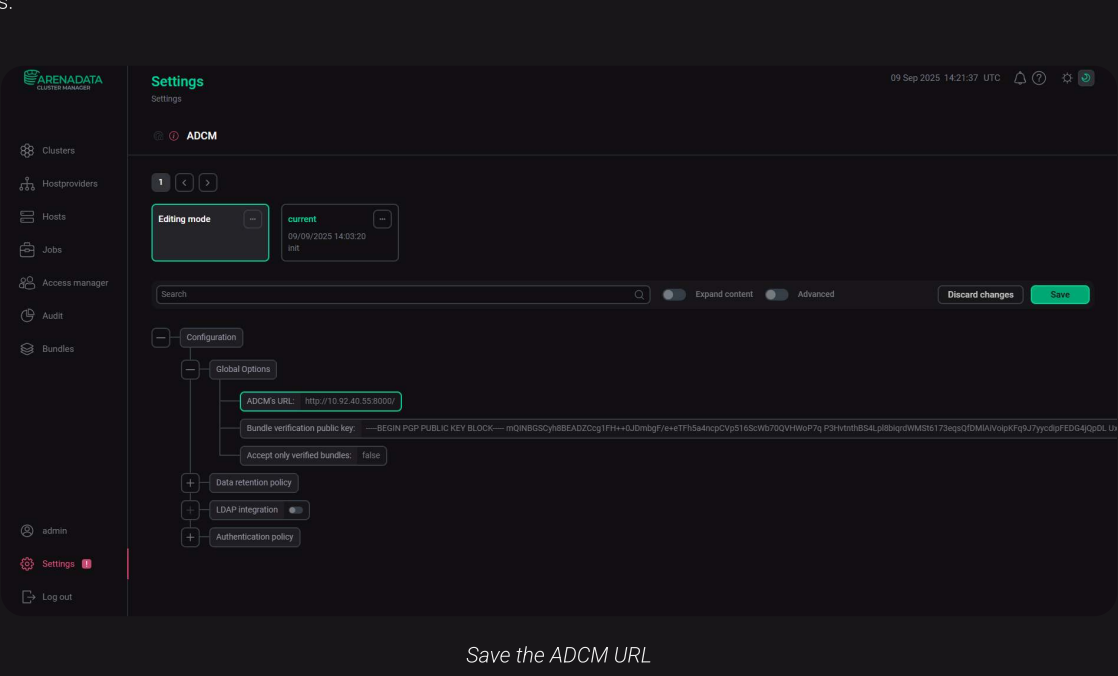
The Settings page

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).

Online update

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
```

Offline update

You can also try offline ADCM update in situations without Internet access.

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

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
- ADB Cloud Bundle

For offline installation, the SSH hostprovider is recommended.

CAUTION

Starting with the 6.23.3.44 version, ADB supports both FQDN and short host names. For the previous ADB versions, it is not recommended to use FQDNs (as FQDNs cause errors during the ***Expand*** action and a workaround is needed).

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.

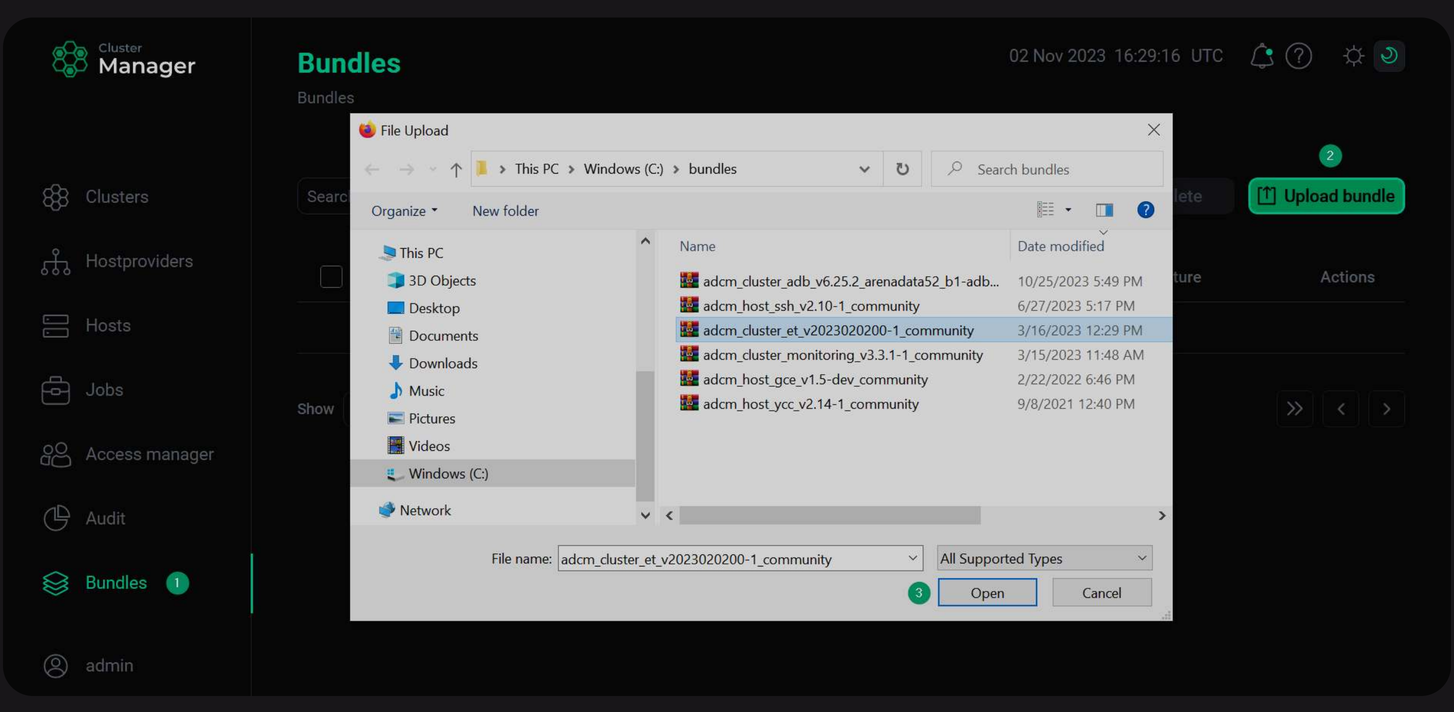
Step 1. Get the Enterprise version of a cluster bundle

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
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
Open
Cancel
3

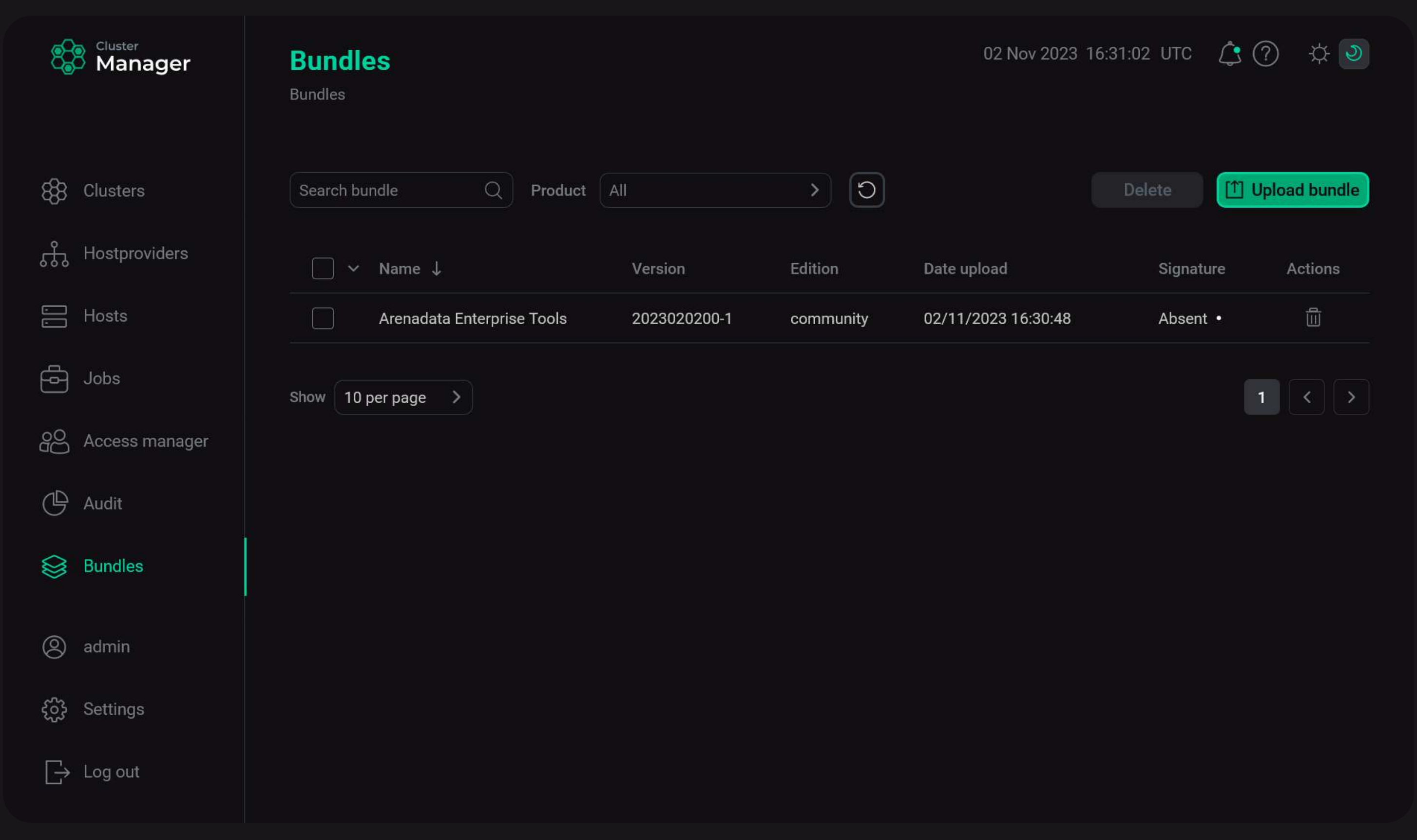
Name	Date modified
adcm_cluster_adb_v6.25.2_arenadata52_b1-adb...	10/25/2023 5:49 PM
adcm_host_ssh_v2.10-1_community	6/27/2023 5:17 PM
adcm_cluster_et_v2023020200-1_community	3/16/2023 12:29 PM
adcm_cluster_monitoring_v3.3.1-1_community	3/15/2023 11:48 AM
adcm_host_gce_v1.5-dev_community	2/22/2022 6:46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

File name: adcm_cluster_et_v2023020200-1_community

All Supported Types

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

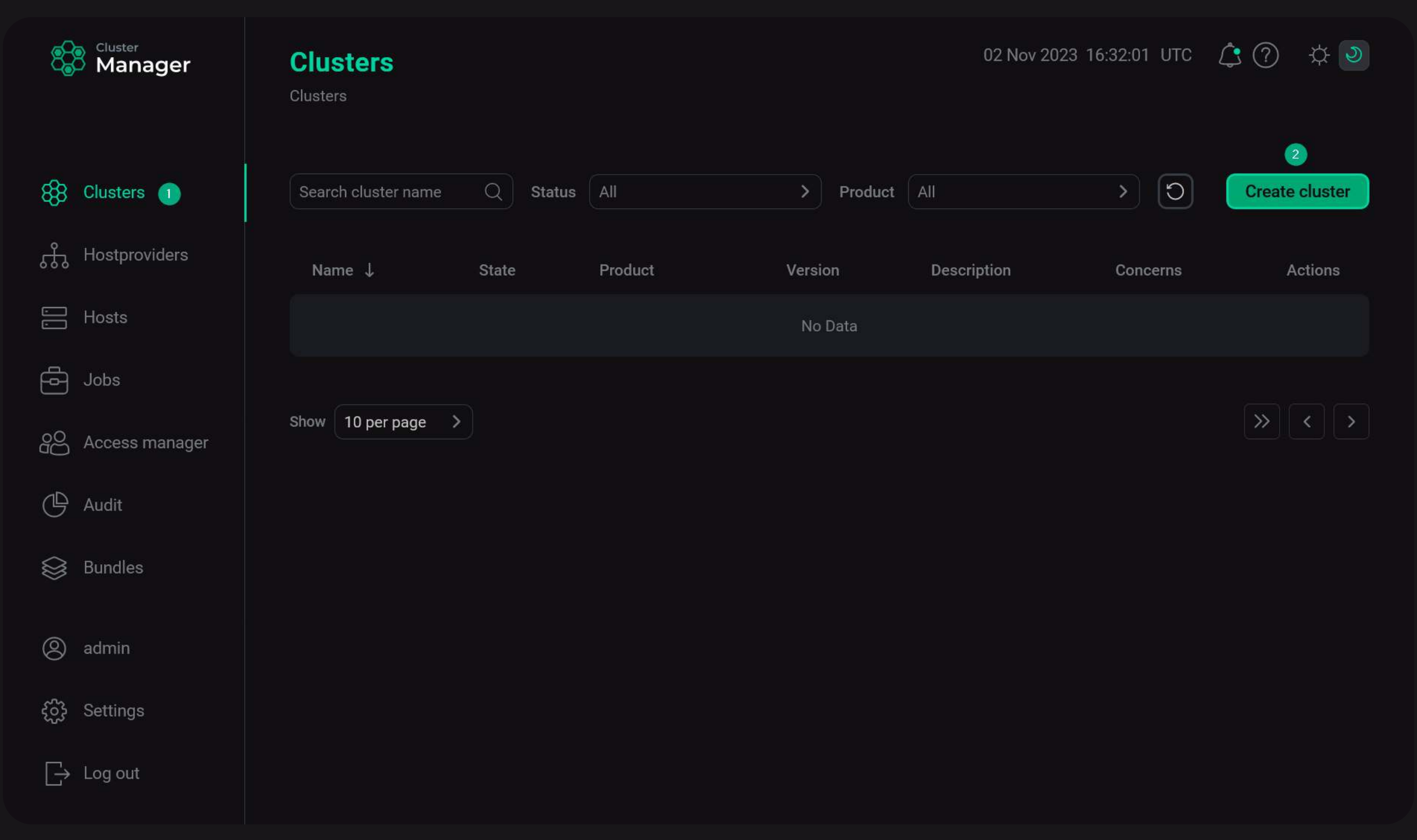
Step 3. Create a cluster based on the uploaded bundle

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***.



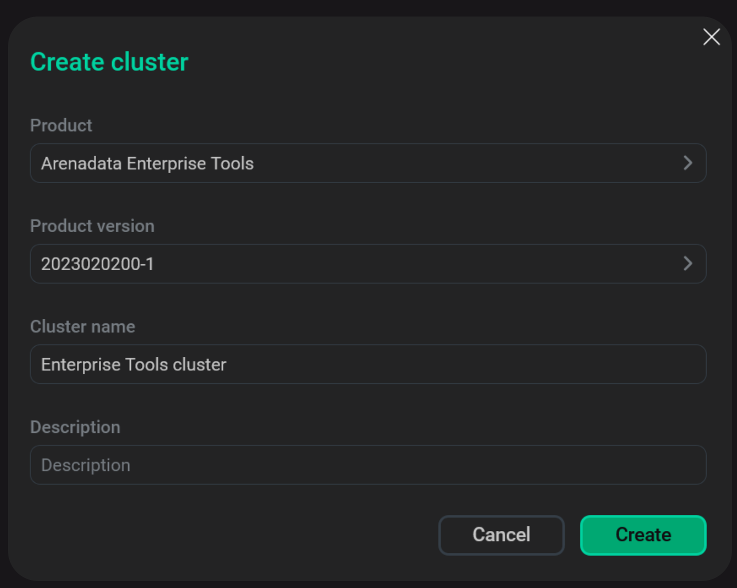
Cluster Manager
Clusters
Clusters
02 Nov 2023 16:32:01 UTC
Clusters
1
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status
All
Product
All
2
Create cluster
Show
10 per page

Name	State	Product	Version	Description	Concerns	Actions

No Data

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
Cancel
Create

Product: Arenadata Enterprise Tools

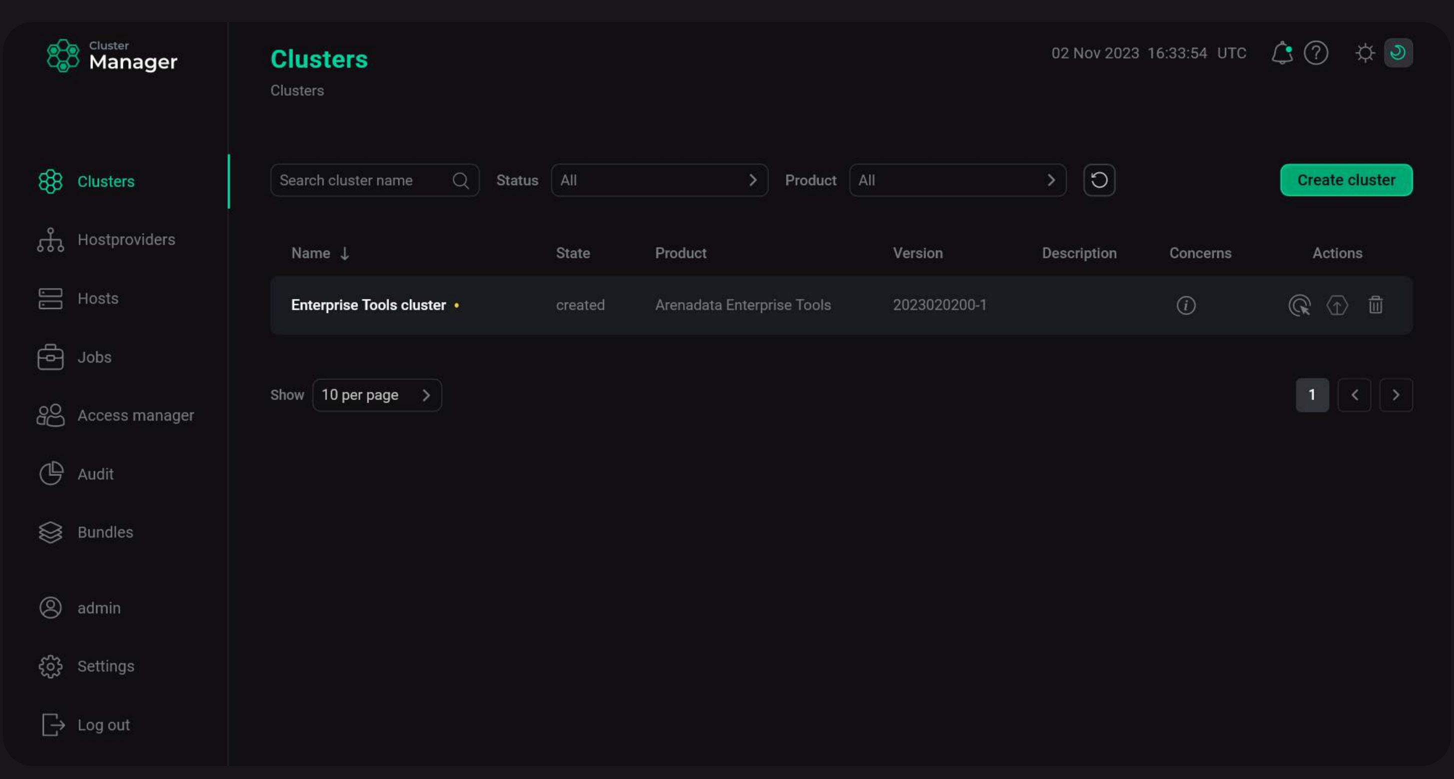
Product version: 2023020200-1

Cluster name: Enterprise Tools cluster

Description: Description

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

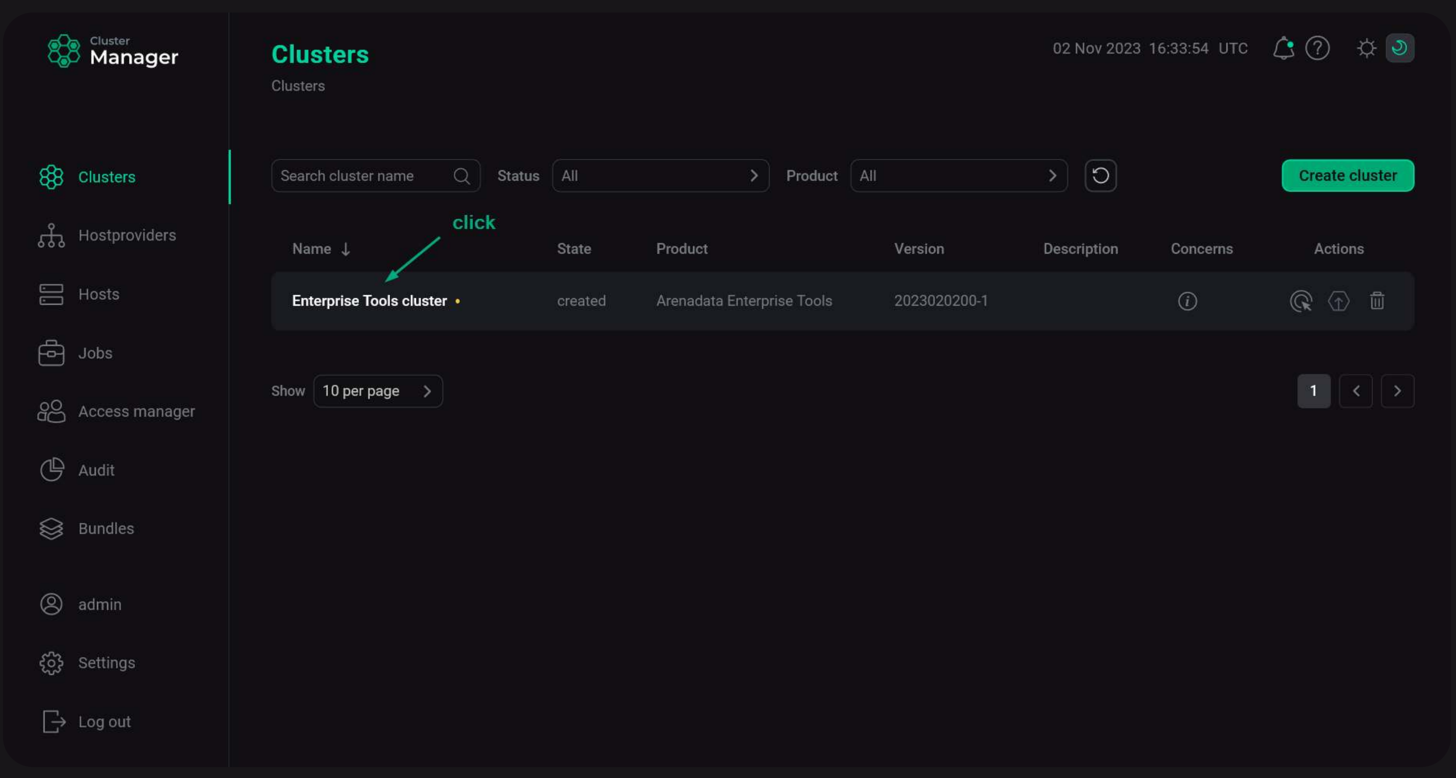
Add services

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:

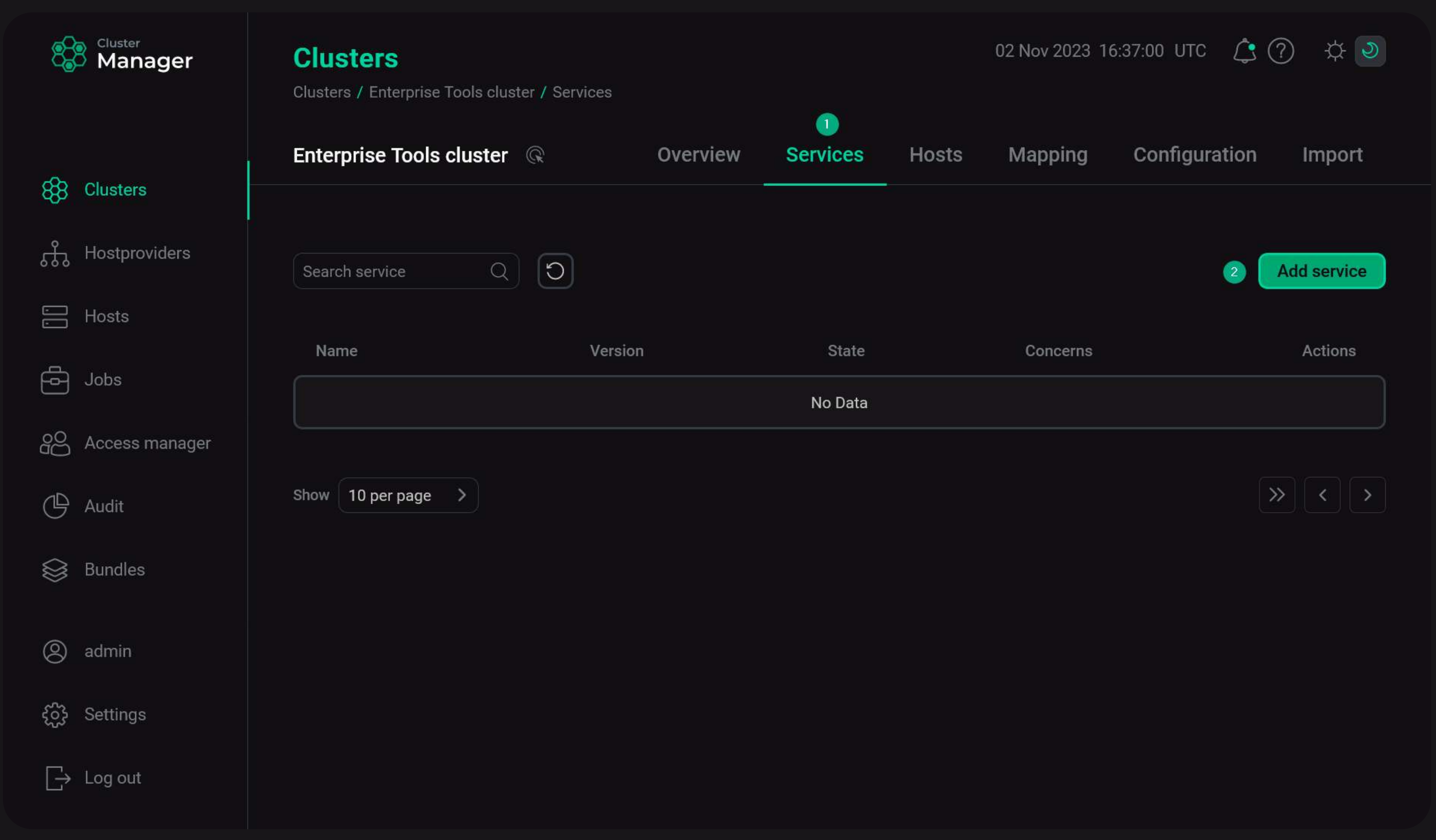
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
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
click
Show
10 per page
1

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

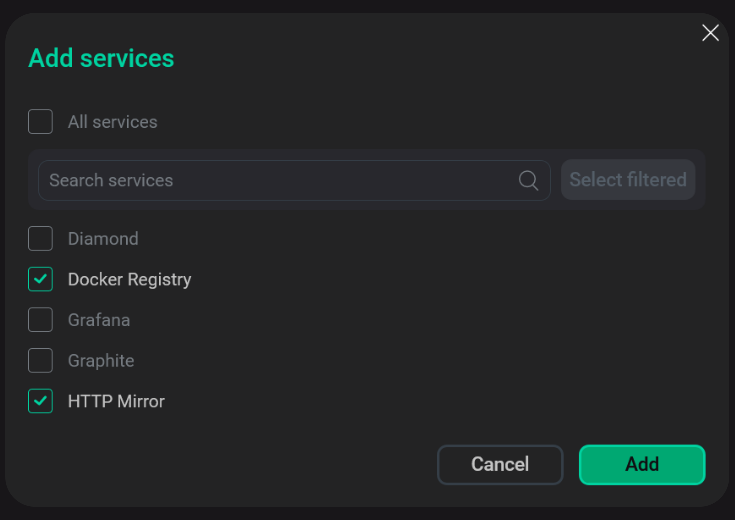
Select a cluster

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


Cluster Manager
Clusters
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 16:37:00 UTC
Enterprise Tools 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

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

☐ Diamond

☑ Docker Registry

☐ Grafana

☐ Graphite

☑ HTTP Mirror

Select services

The brief description of available services is listed below.

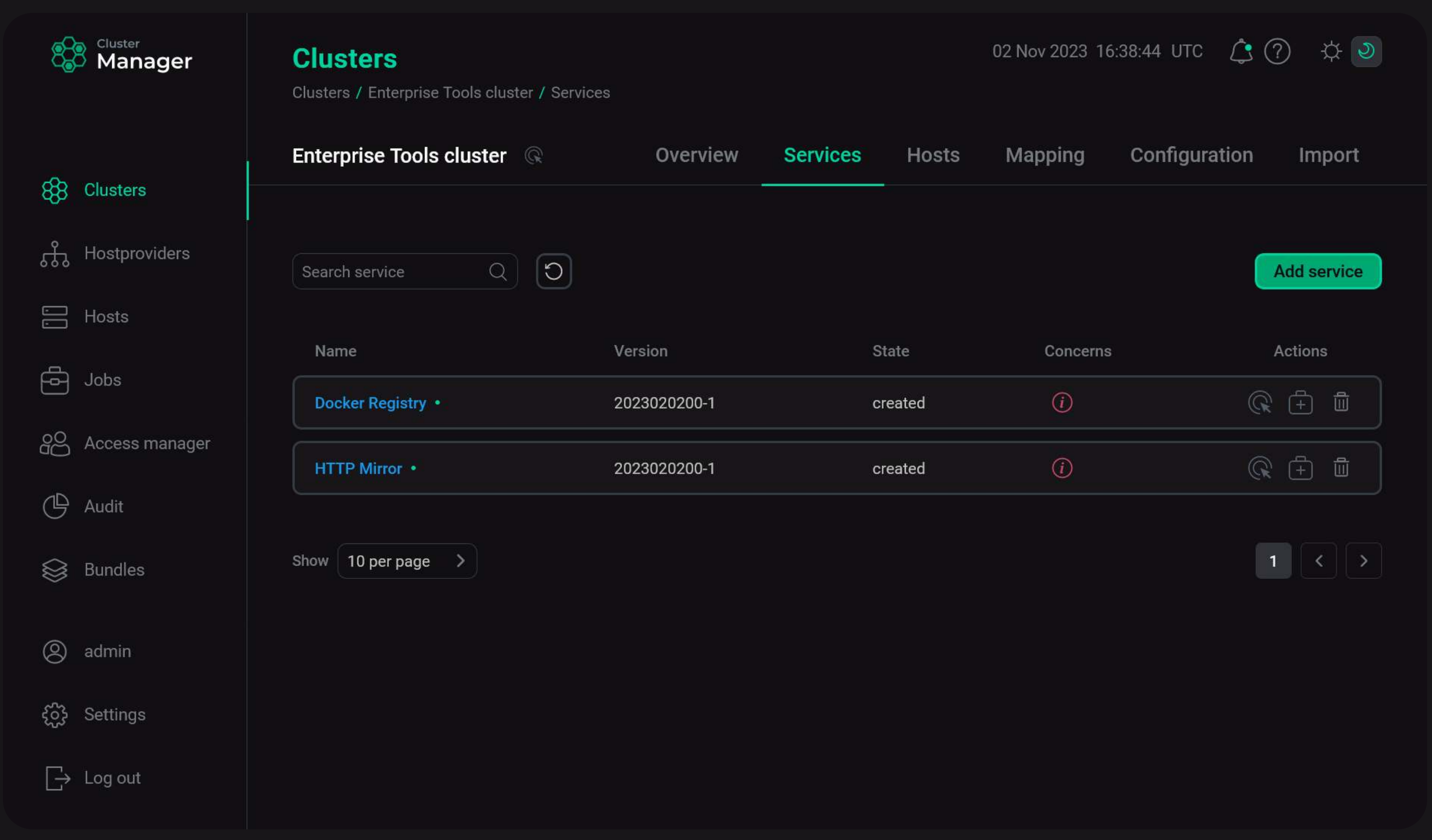
Services that should be added to the Enterprise Tools cluster

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
Clusters / Enterprise Tools cluster / Services
02 Nov 2023 16:38:44 UTC
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

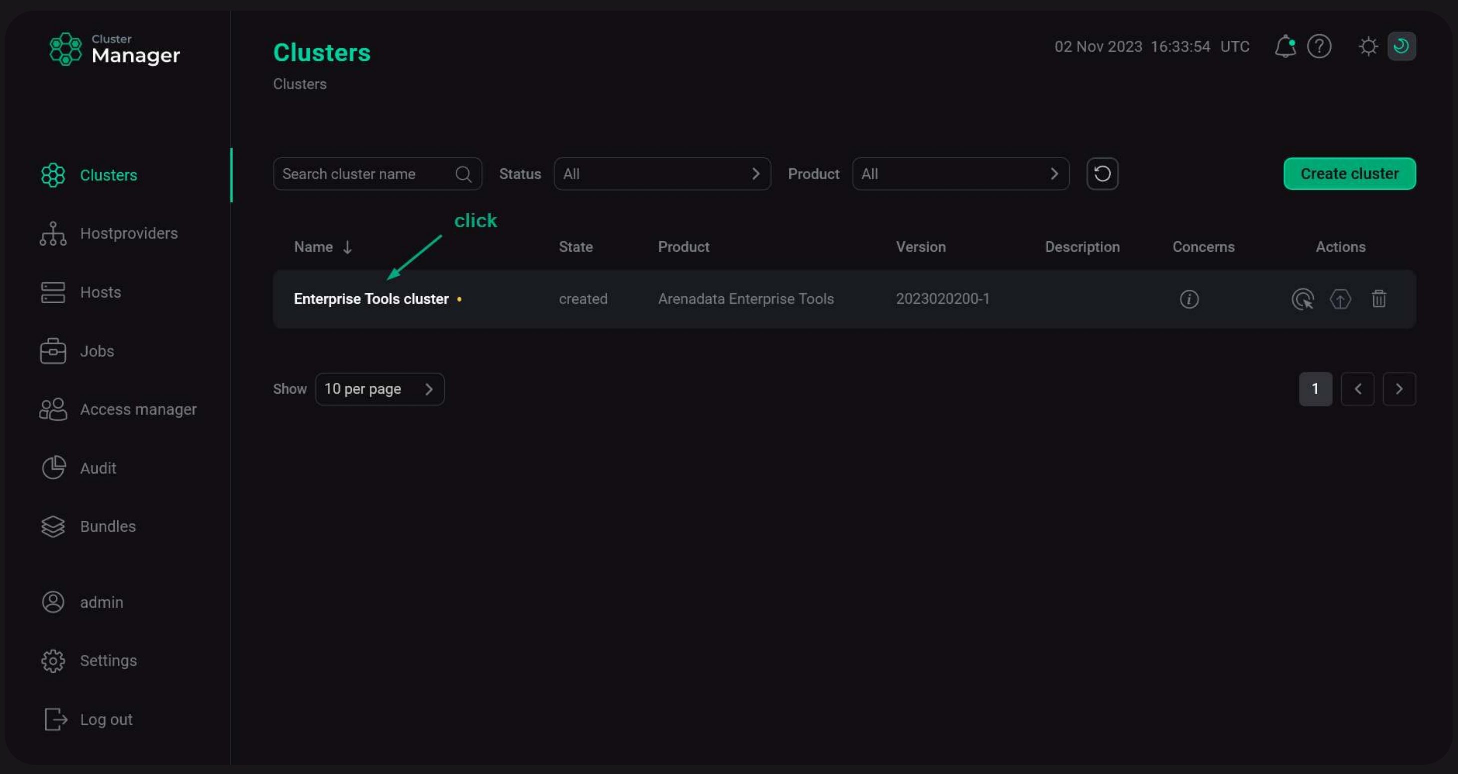
Add hosts to a cluster

Daria Barysheva

To Table of Contents

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

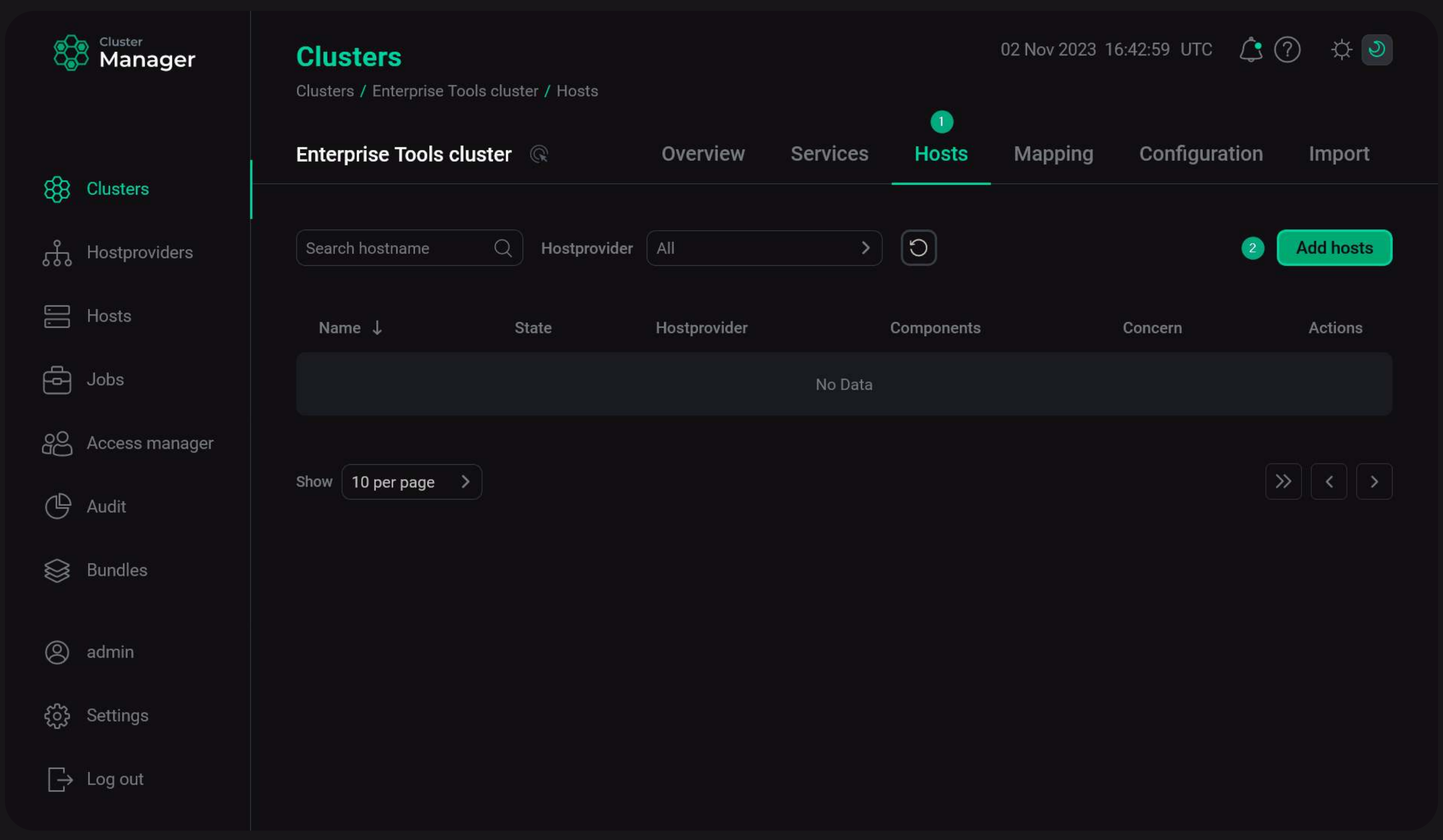
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


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

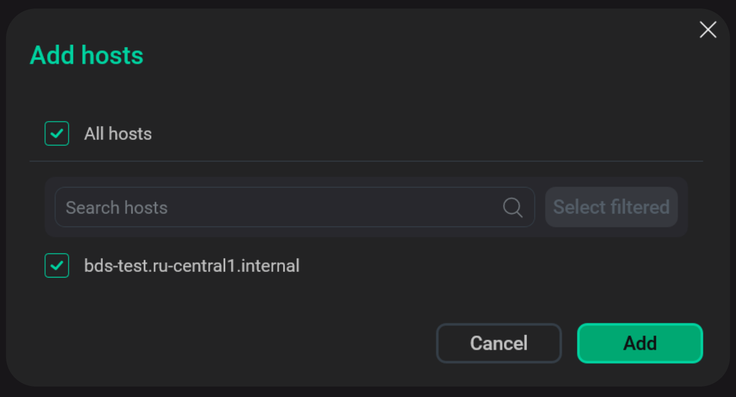
2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.


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
Show 10 per page

Name	State	Hostprovider	Components	Concern	Actions
No Data					

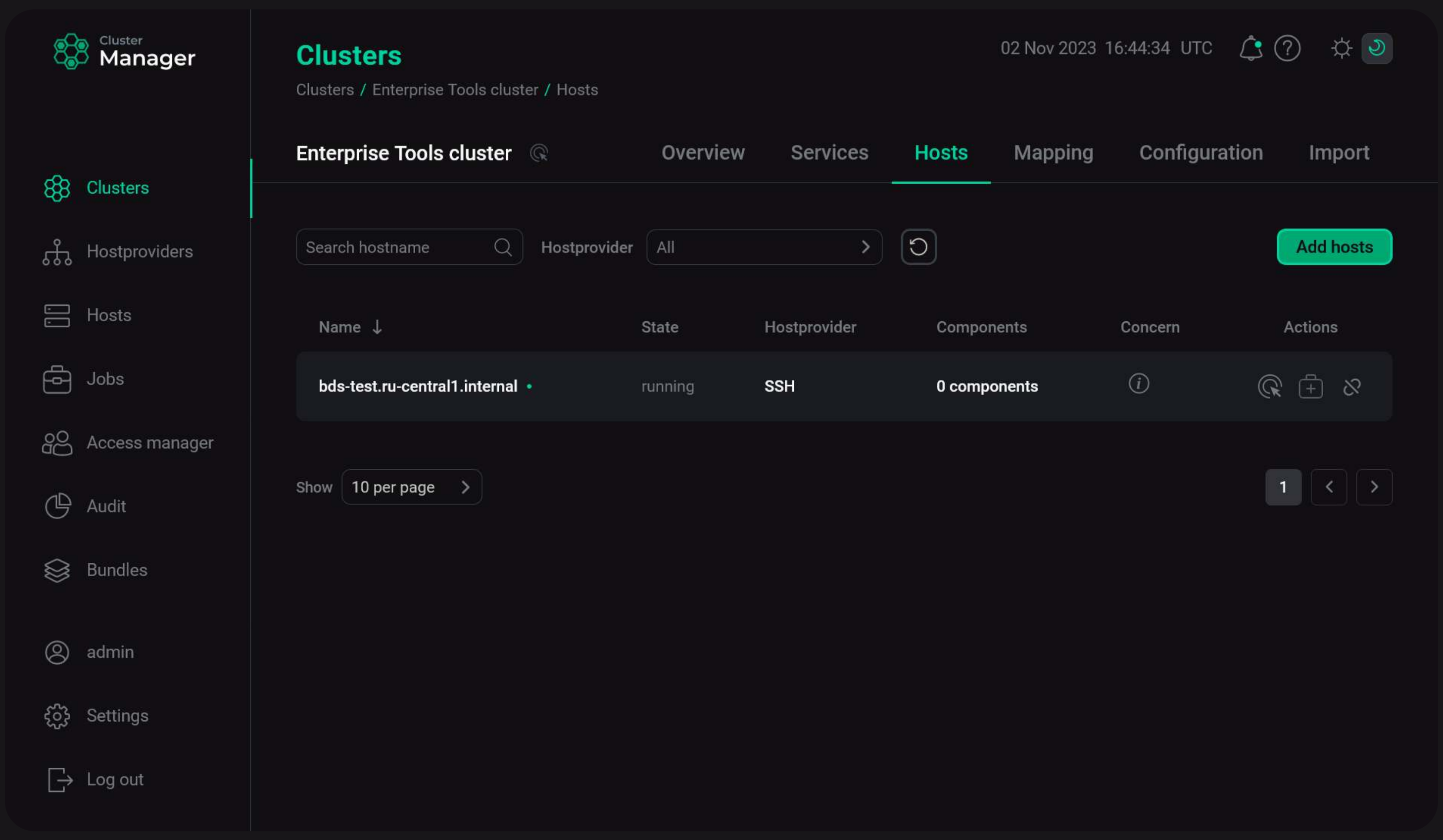
Switch to adding hosts

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

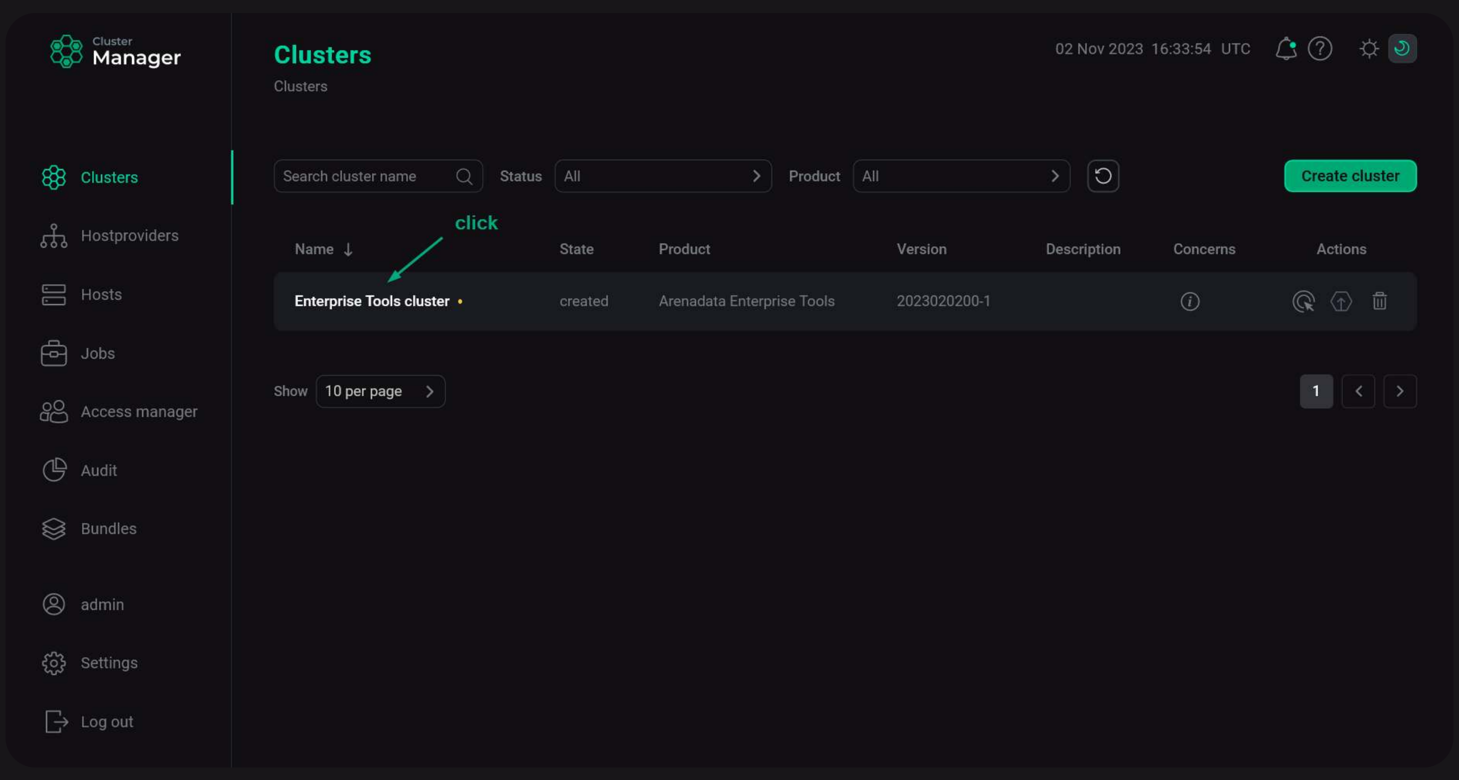
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts.

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:

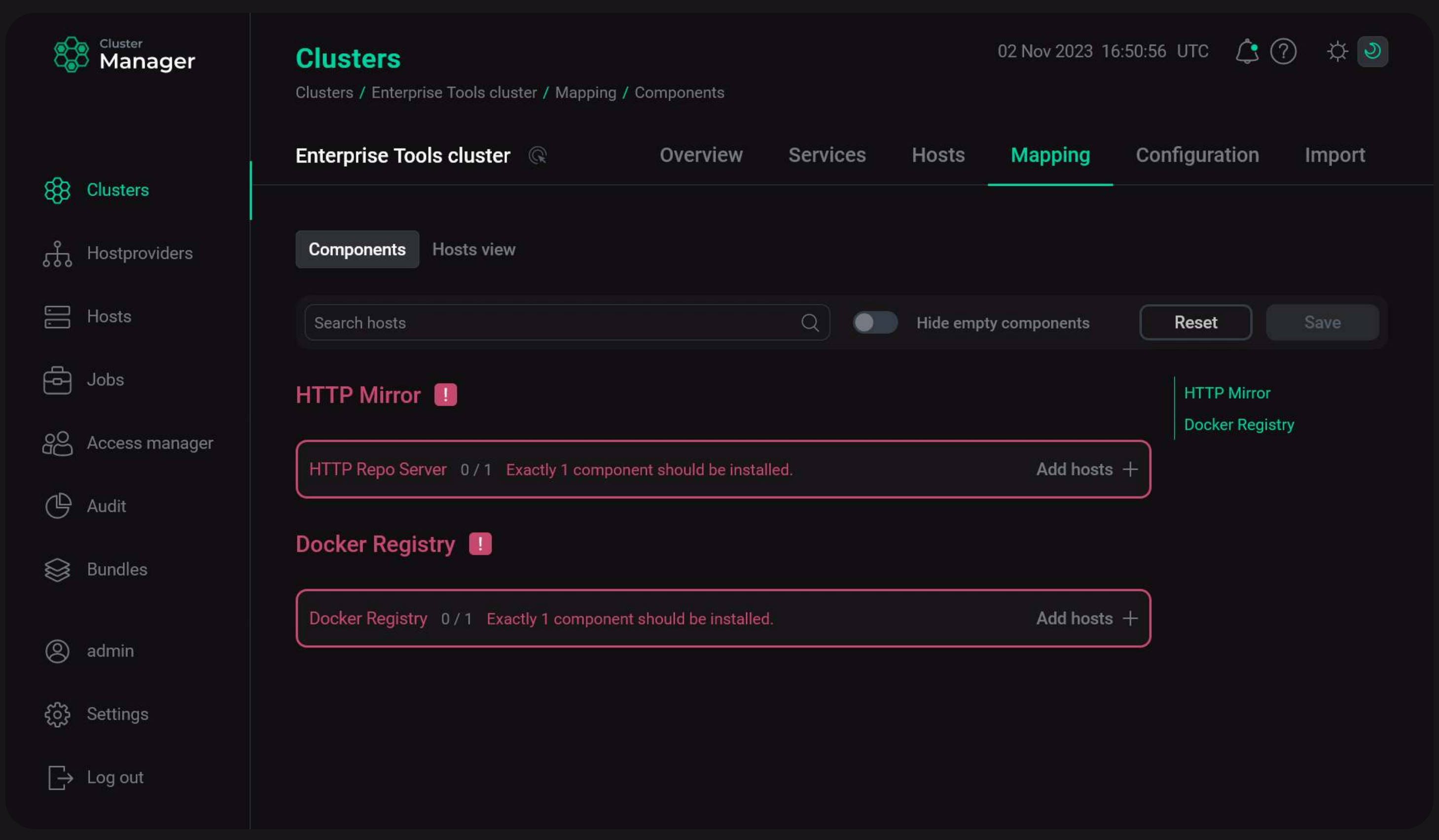
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
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 ***Mapping*** tab on the cluster page.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
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
Docker Registry
Add hosts

HTTP Mirror

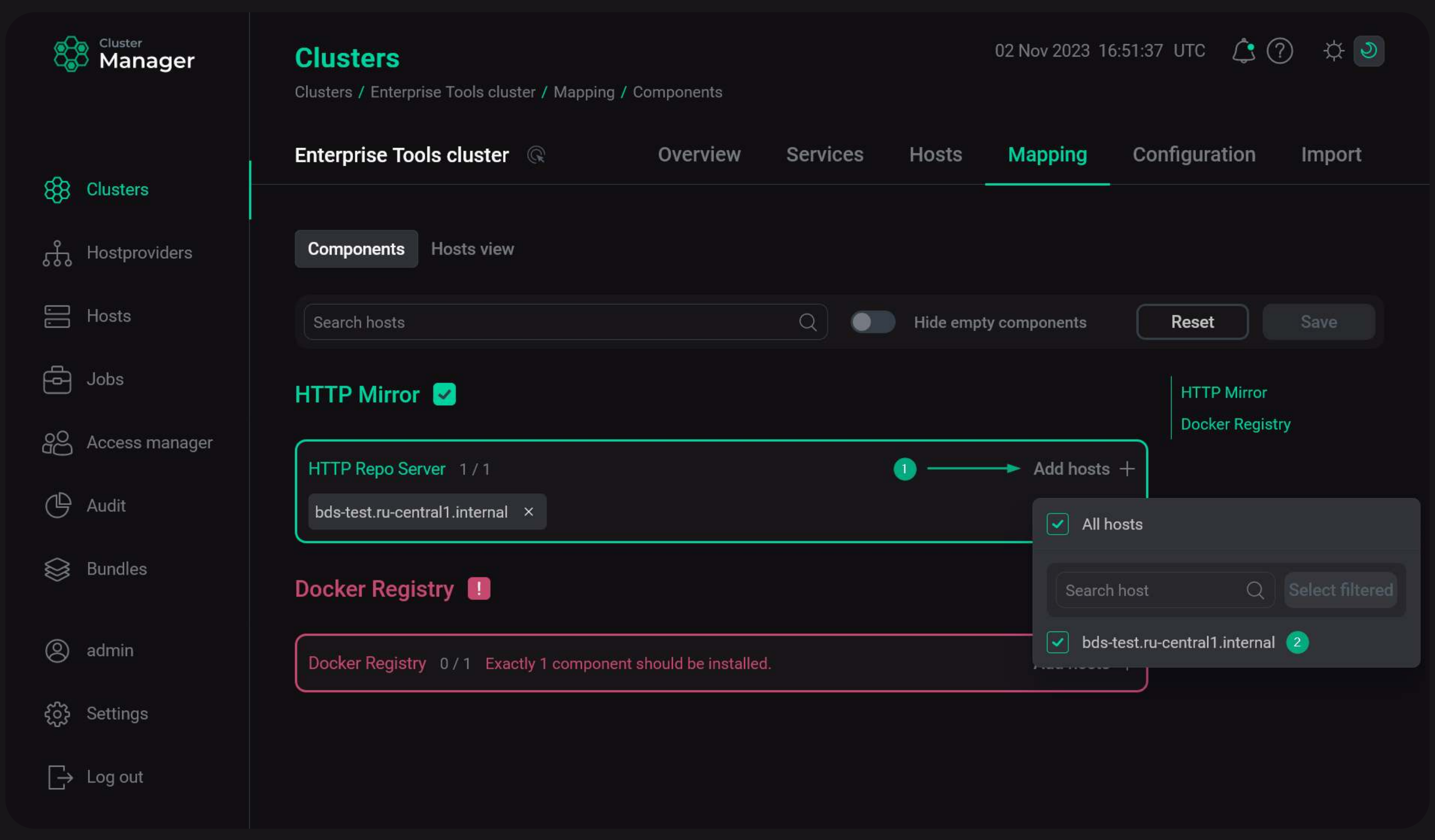
HTTP Repo Server 0 / 1 Exactly 1 component should be installed.

Docker Registry

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
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
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
Docker Registry
1
Add hosts
☑ All hosts
Search host
Select filtered
☑ bds-test.ru-central1.internal 2

HTTP Mirror ☑

HTTP Repo Server 1 / 1

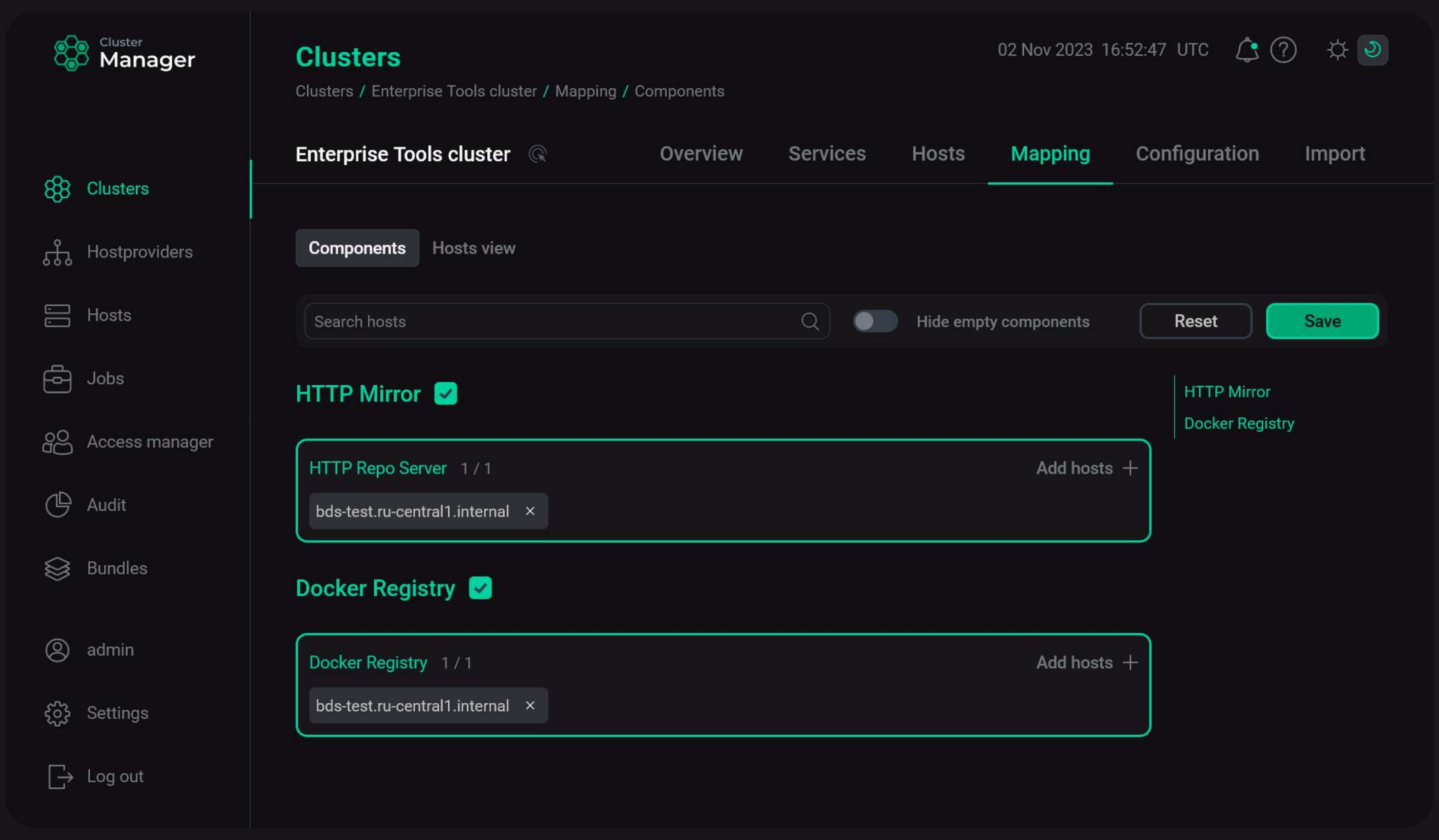
bds-test.ru-central1.internal ×

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
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
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
Docker Registry
Add hosts
Add hosts

HTTP Mirror ☑

HTTP Repo Server 1 / 1

bds-test.ru-central1.internal ×

Docker Registry ☑

Docker Registry 1 / 1

bds-test.ru-central1.internal ×

Save mapping of components

Install a cluster

Daria Barysheva

Contents

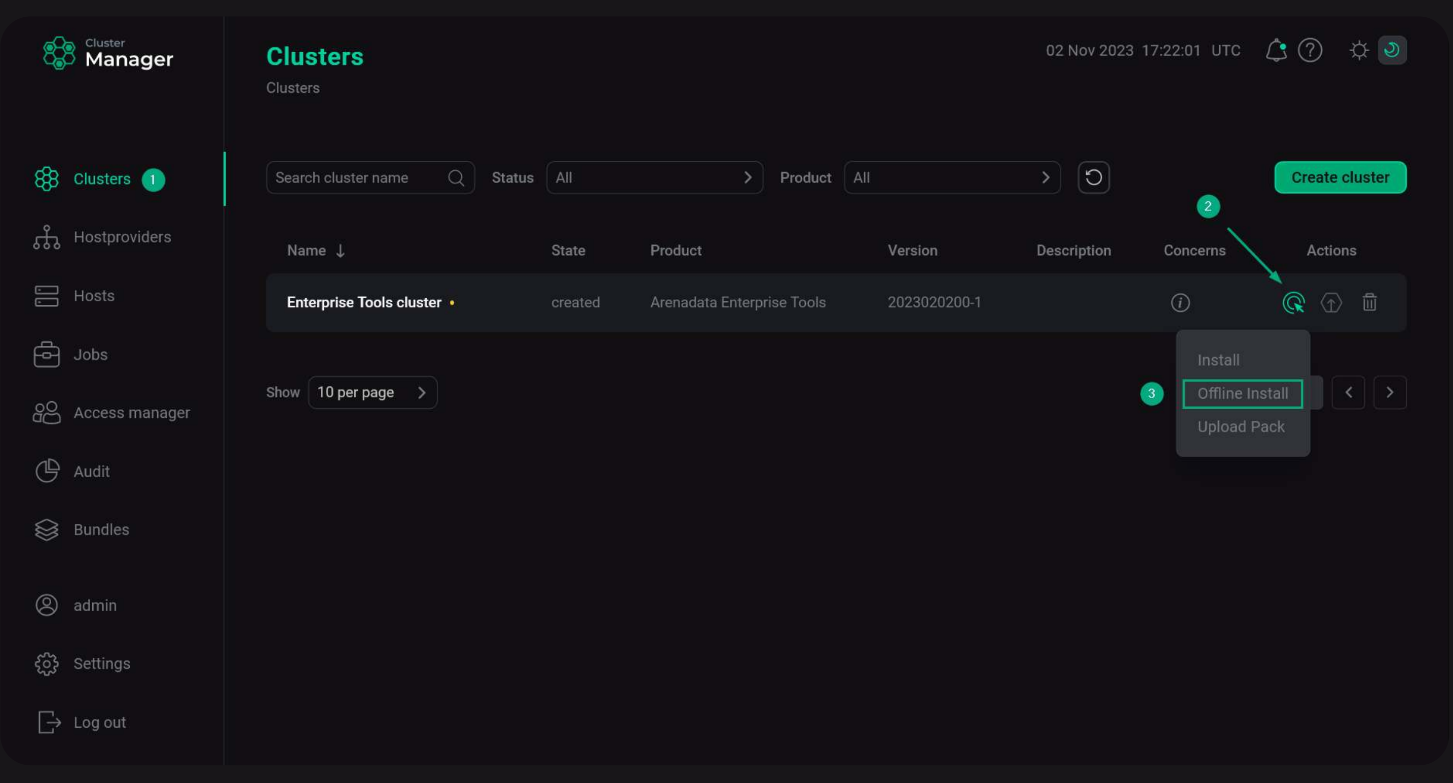
To Table of Contents

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

Step 1. Run installation

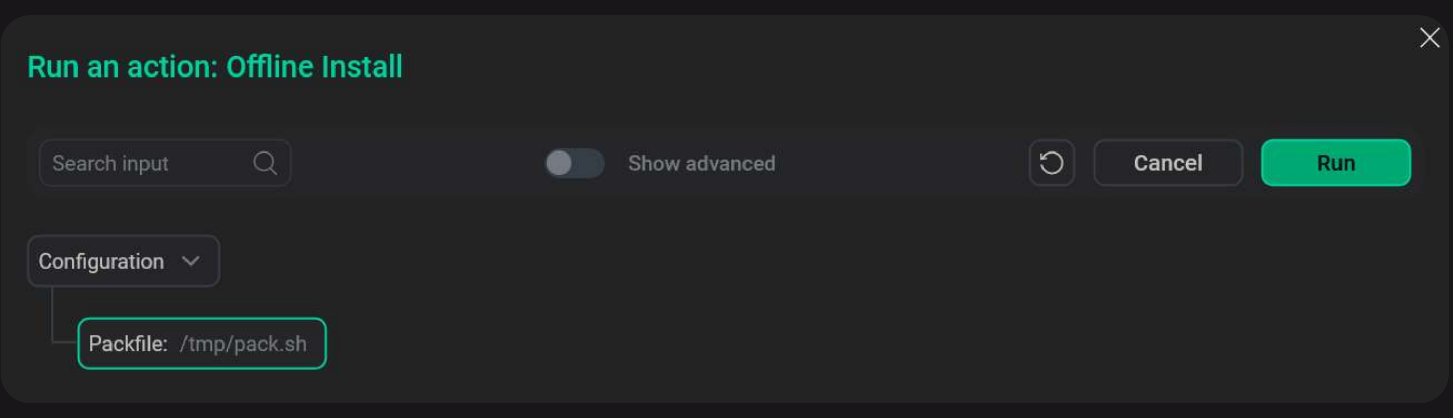
The last step of adding a new Enterprise Tools cluster is to install its services. To do this:

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
Name
State
Product
Version
Description
Concerns
Actions
Enterprise Tools cluster
created
Arenadata Enterprise Tools
2023020200-1
Install
Offline Install
Upload Pack
3
Show 10 per page

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

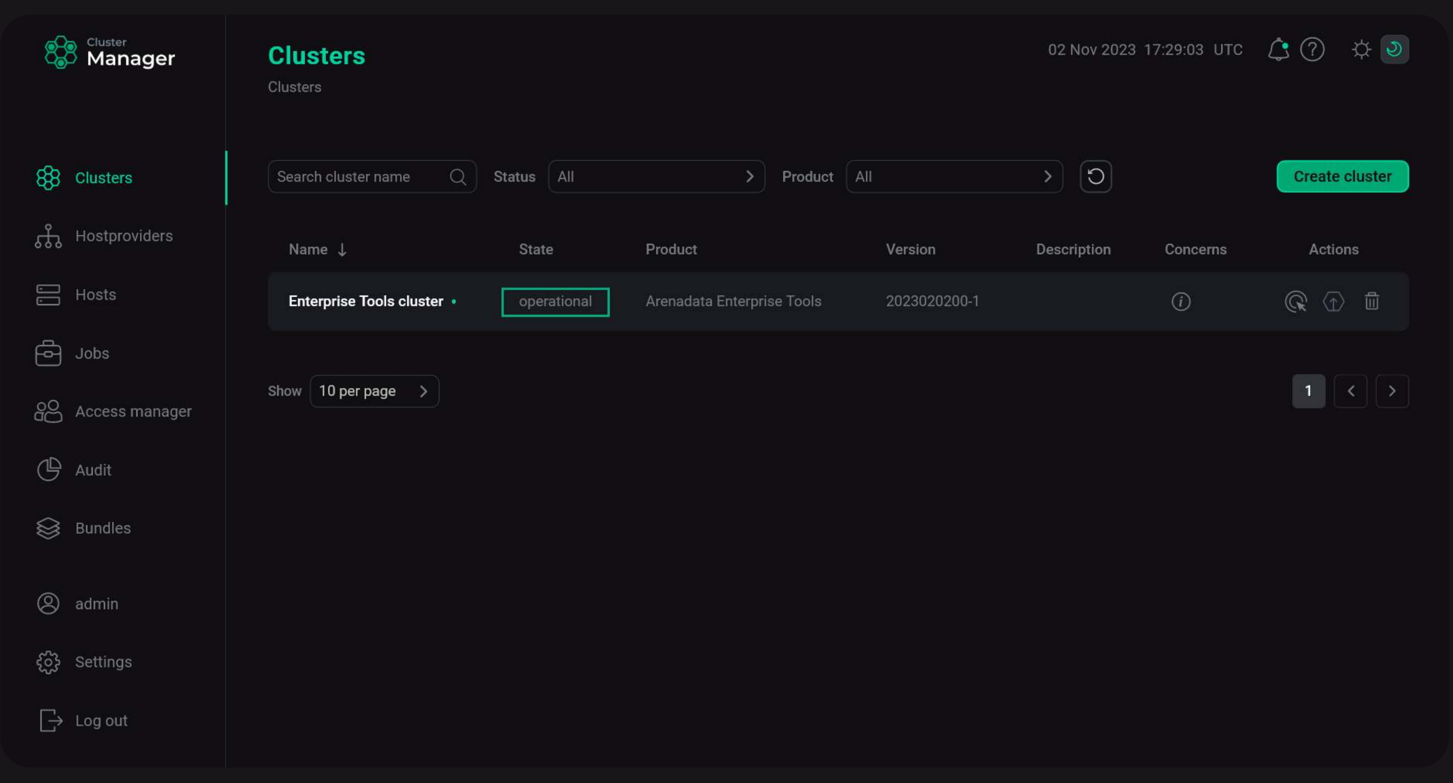
Packfile: /tmp/pack.sh

Verify installation

3. Wait until the installation process is completed.

NOTE

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

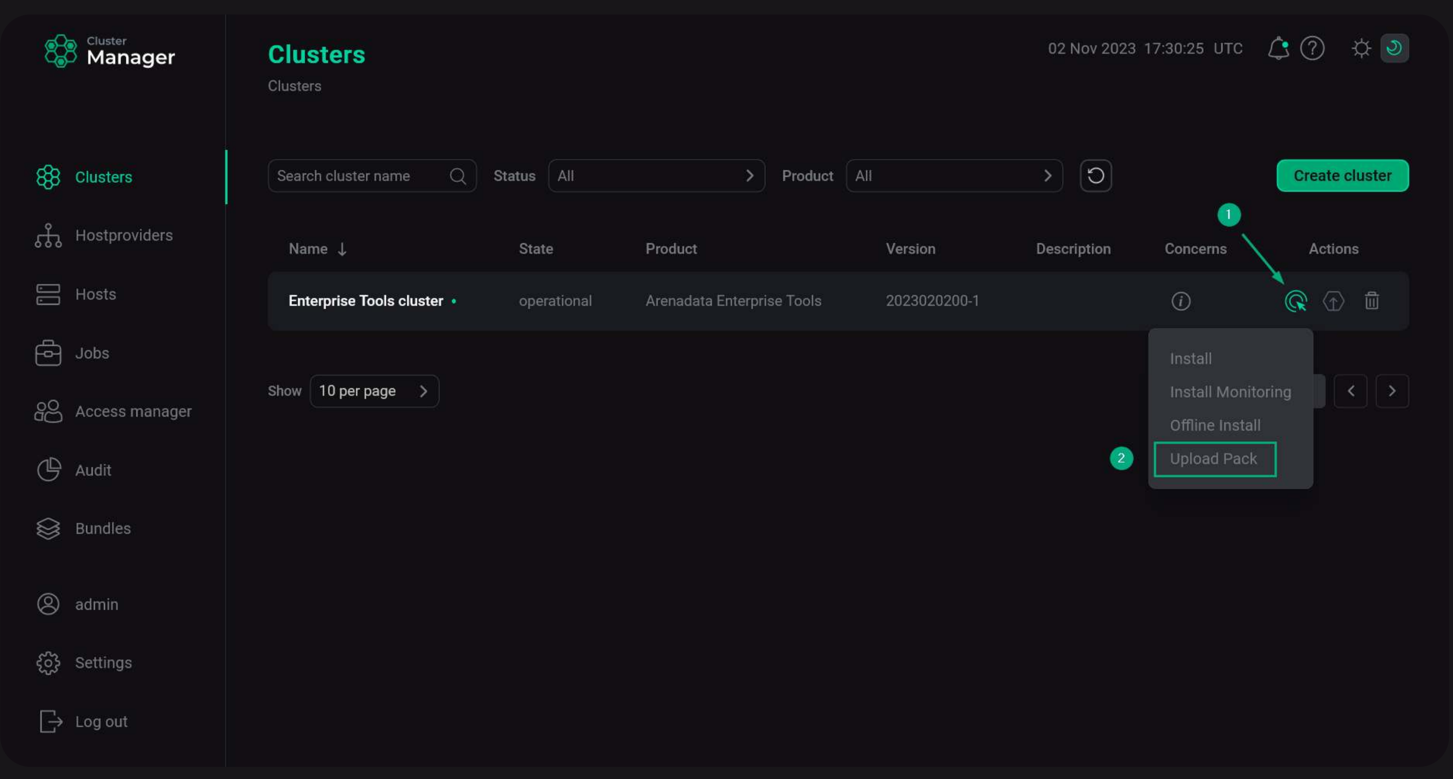

Cluster Manager
Clusters
Clusters
02 Nov 2023 17:29:03 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Name
State
Product
Version
Description
Concerns
Actions
Enterprise Tools cluster
operational
Arenadata Enterprise Tools
2023020200-1
Show 10 per page
1

Cluster installation is completed

Step 2. Upload packages for offline installation

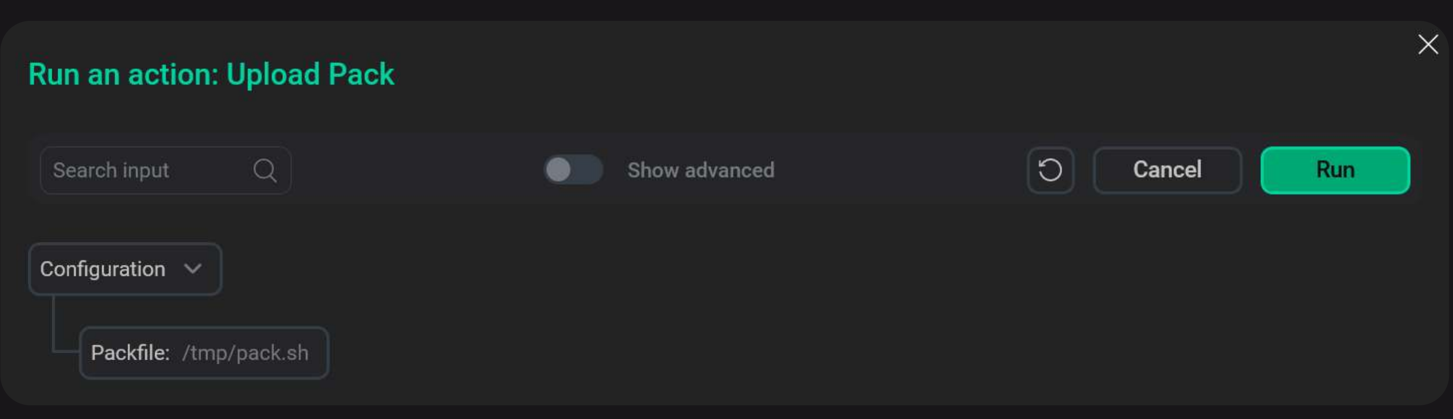
In addition to installing the cluster, you should process the shell script that is received from the Arenadata support team. It is necessary to enable offline installation of products in the future. To do this:

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
Name
State
Product
Version
Description
Concerns
Actions
Enterprise Tools cluster
operational
Arenadata Enterprise Tools
2023020200-1
Install
Install Monitoring
Offline Install
Upload Pack
2
Show 10 per page

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 ADB installation.

```
total 0
drwxr-xr-x 4 root root 28 Feb 28 17:48 .
drwxr-xr-x 3 root root 28 Feb 28 17:48 ..
drwxr-xr-x 3 root root 32 Feb 28 17:48 ADB
drwxr-xr-x 3 root root 17 Feb 28 17:48 ADM
```

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.

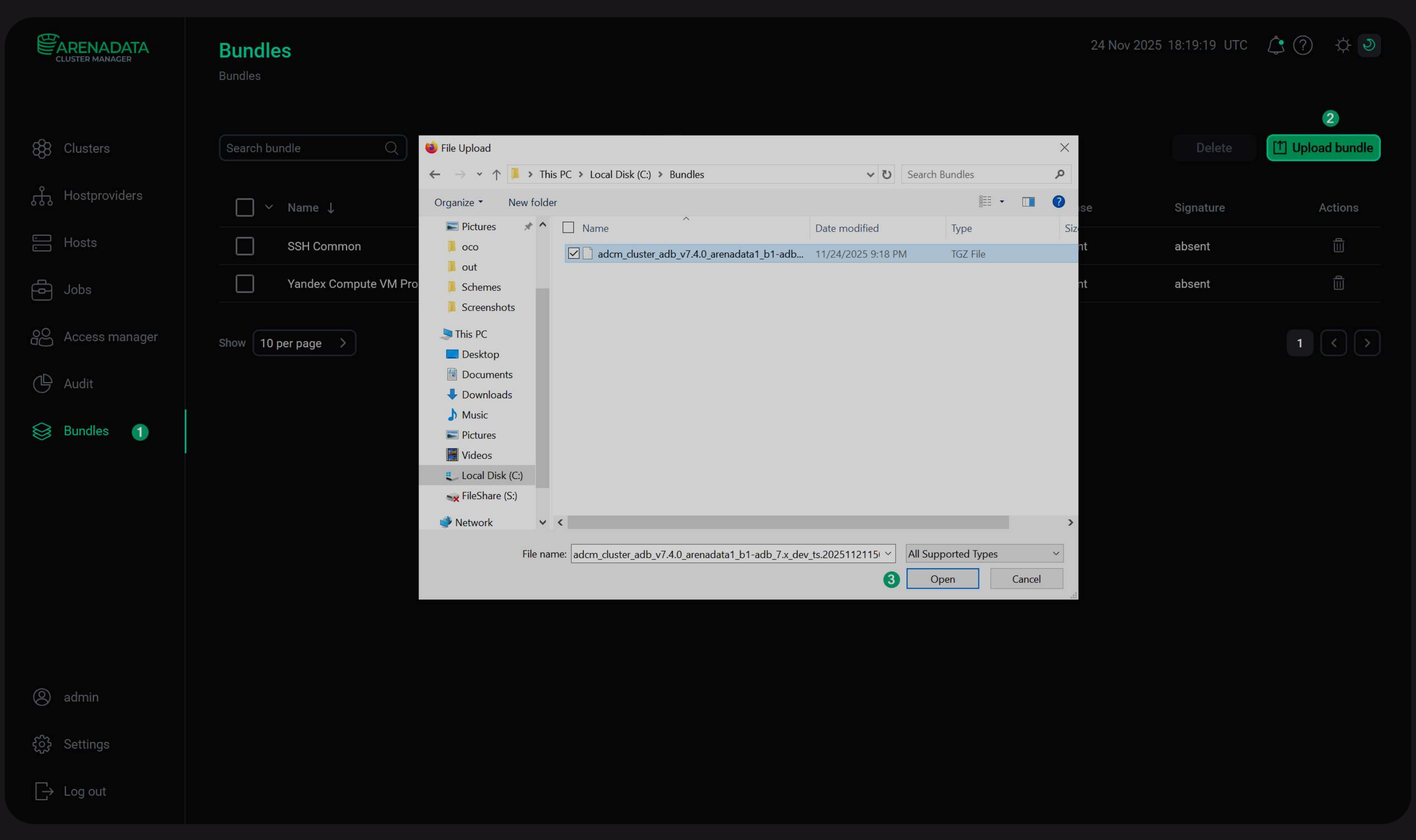
Step 1. Get the Enterprise version of a cluster bundle

Please get a cluster bundle from the Arenadata support team.

Step 2. Upload a cluster bundle to ADCM

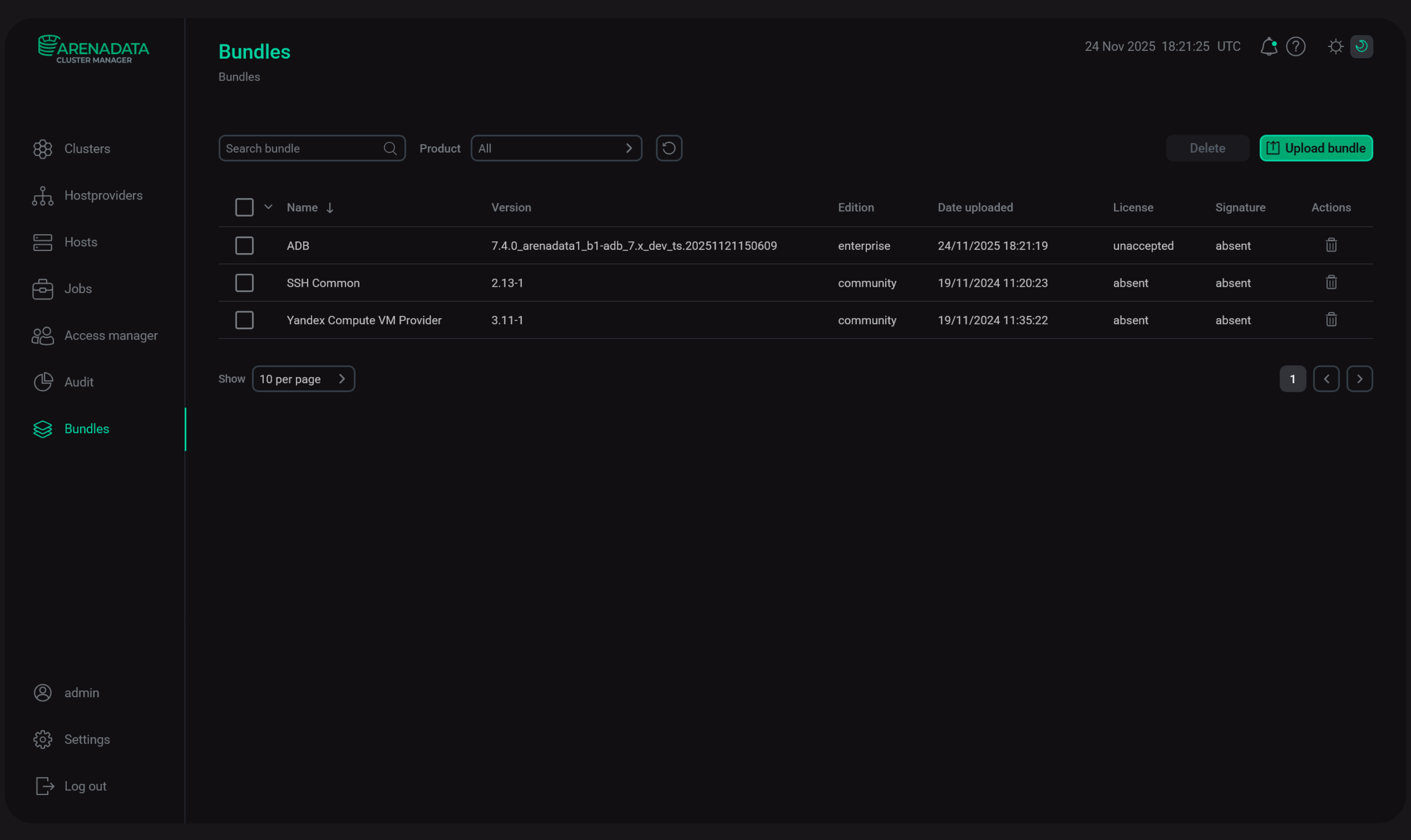
To upload a bundle to ADCM, follow the steps:

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



Upload a bundle

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



Name	Version	Edition	Date uploaded	License	Signature	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	enterprise	24/11/2025 18:21:19	unaccepted	absent	
SSH Common	2.13-1	community	19/11/2024 11:20:23	absent	absent	
Yandex Compute VM Provider	3.11-1	community	19/11/2024 11:35:22	absent	absent	

The result of successful uploading a bundle

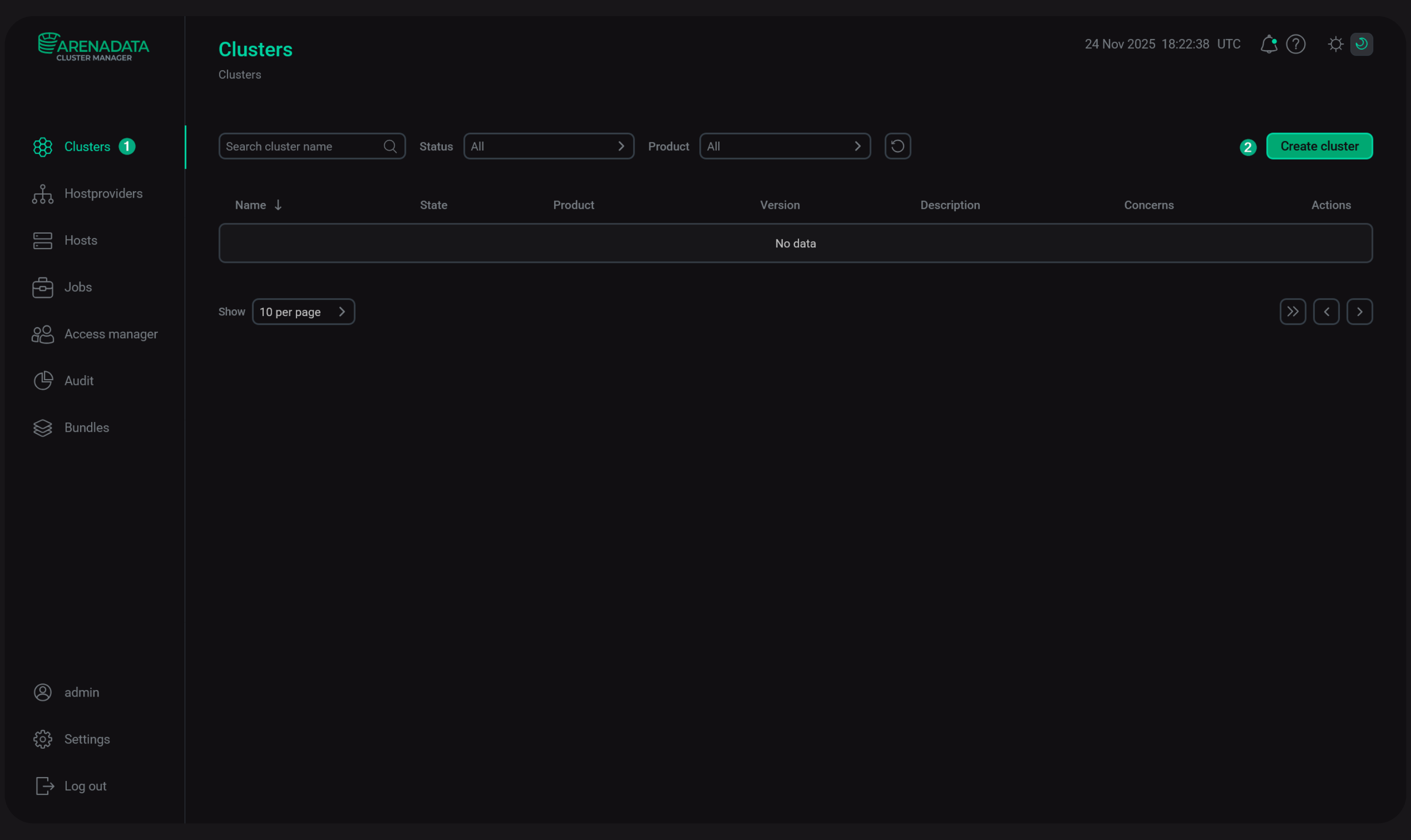
Step 3. Create a cluster based on the uploaded bundle

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.

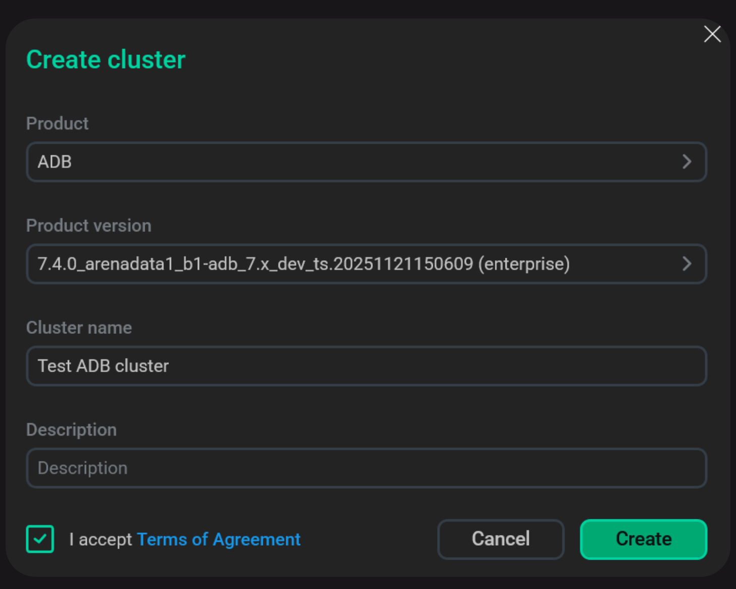
The steps for cluster creation are given below:

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



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: ADB

Product version: 7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609 (enterprise)

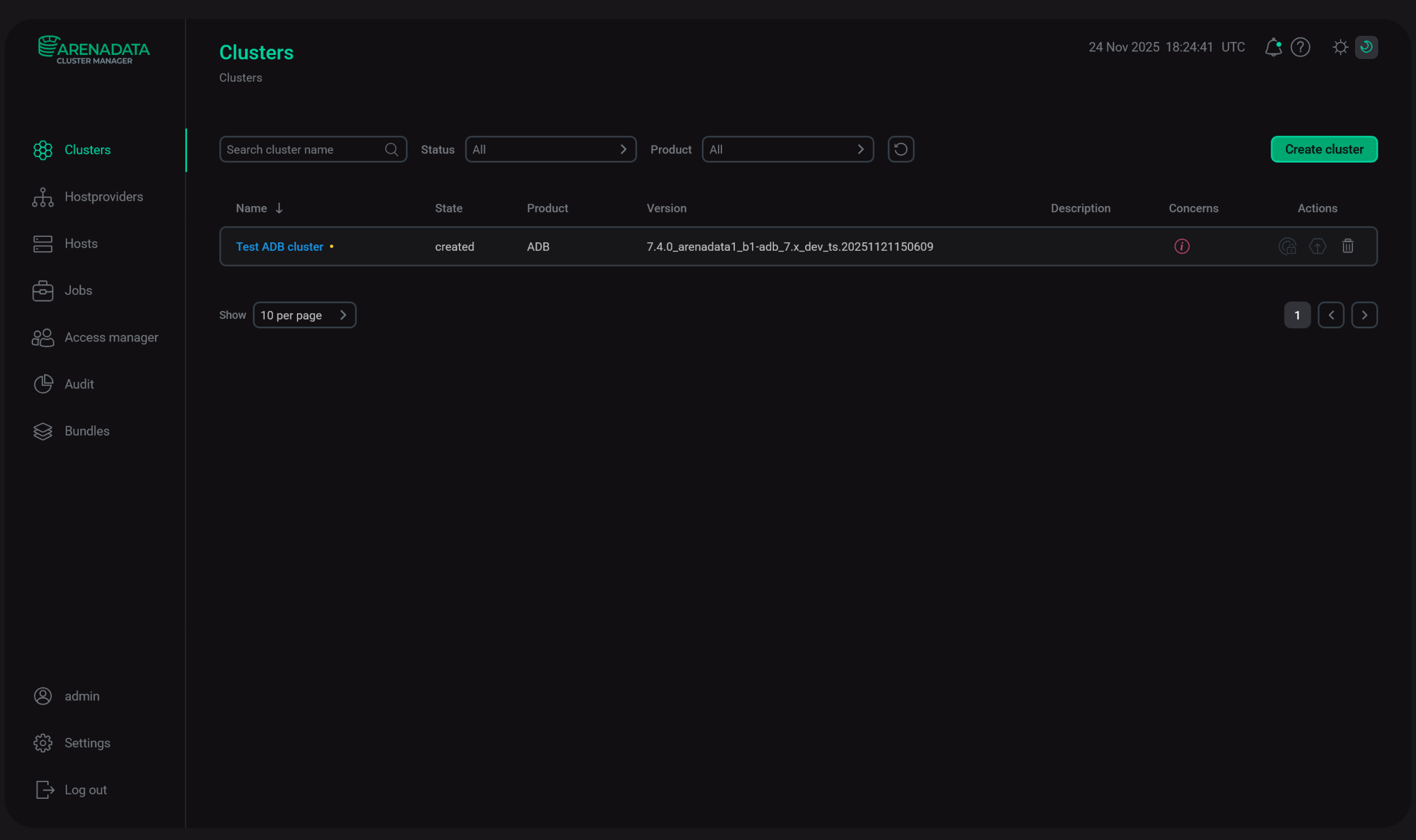
Cluster name: Test ADB cluster

Description: Description

☑ I accept Terms of Agreement

Create a cluster

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



Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

The result of successful cluster creation

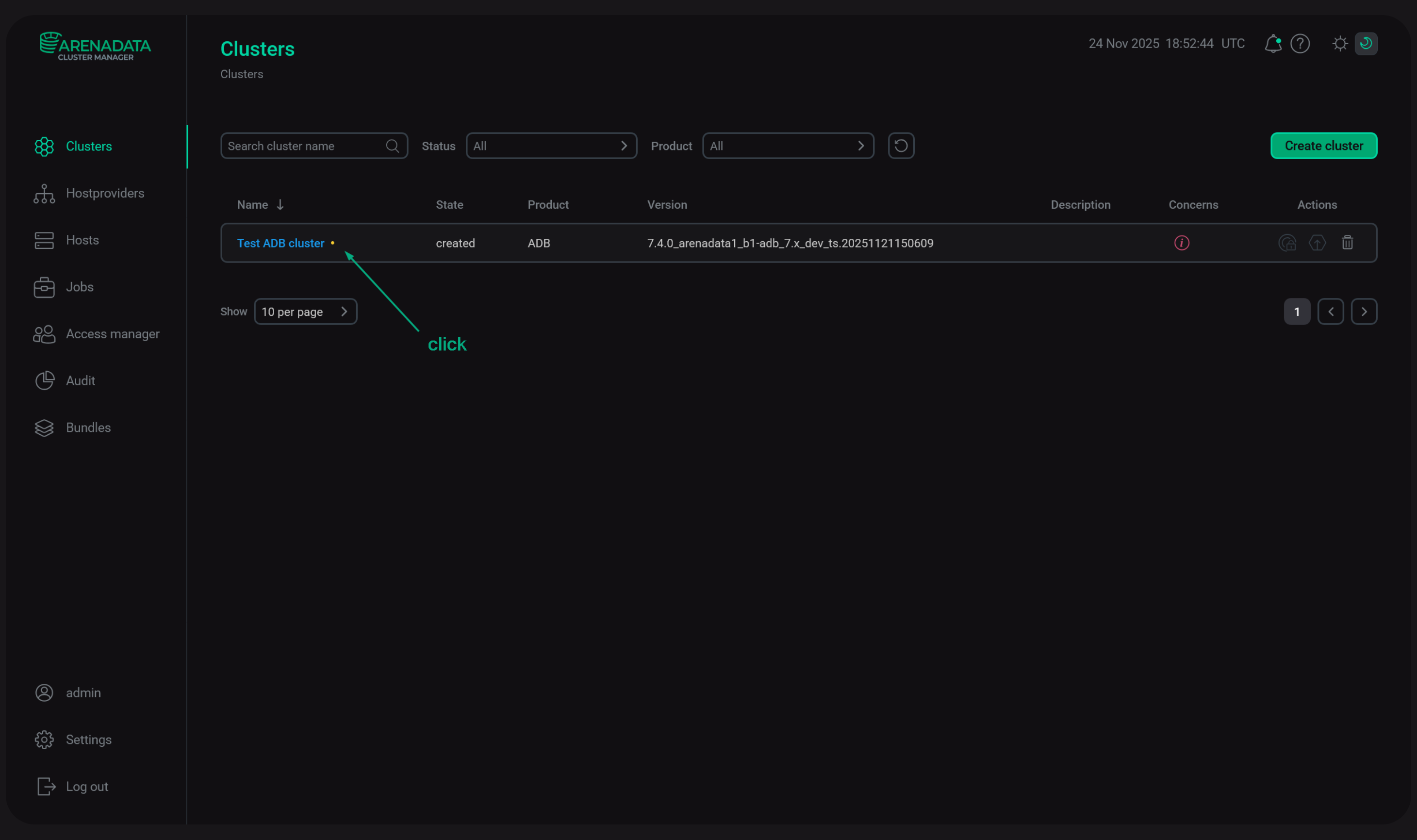
Add services

Daria Barysheva

To Table of Contents

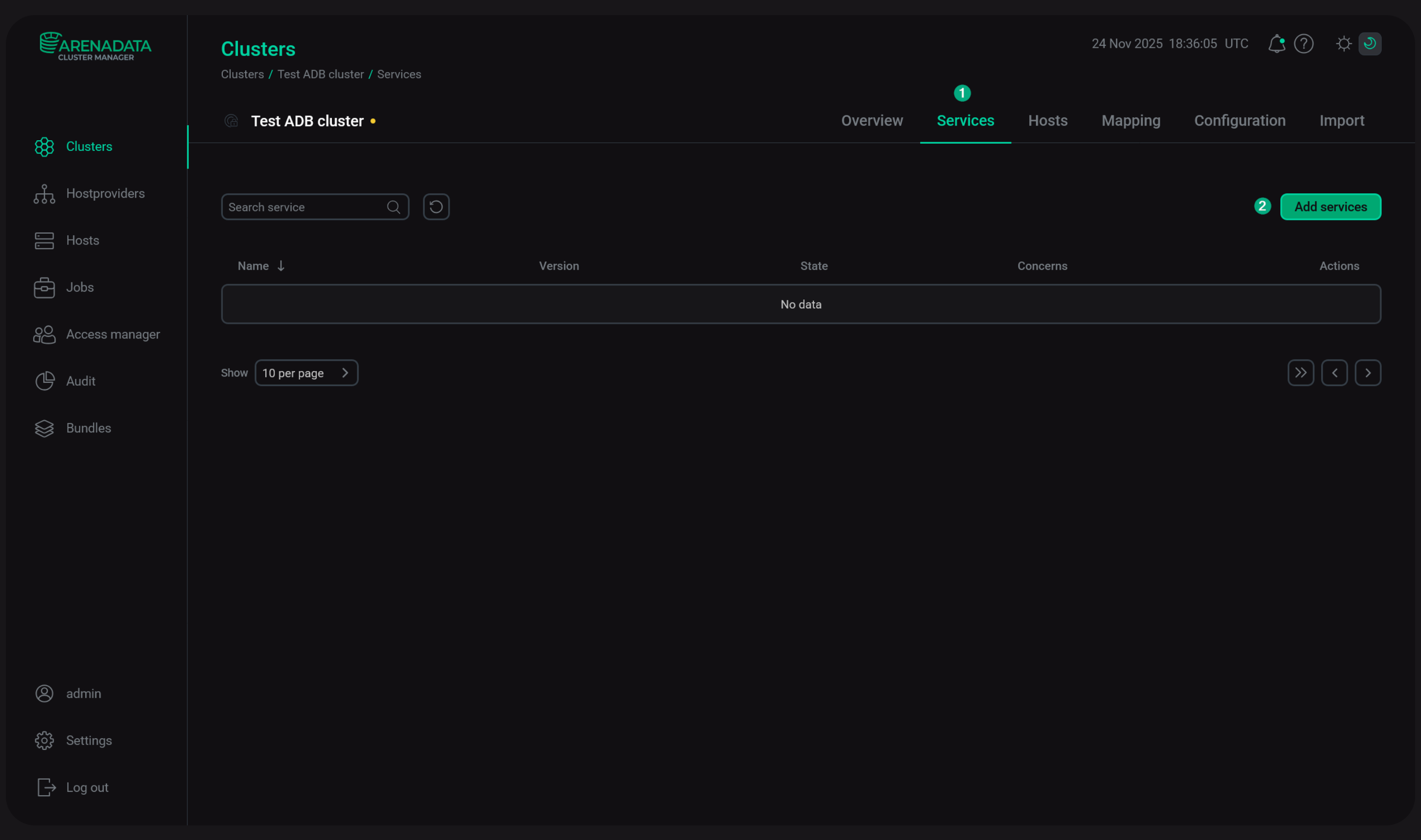
In ADCM a **service** means a software that performs some function. Examples of services for ADB clusters: ADB, ADB Control, Chrony, etc. The steps for adding services to a cluster are listed below:

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



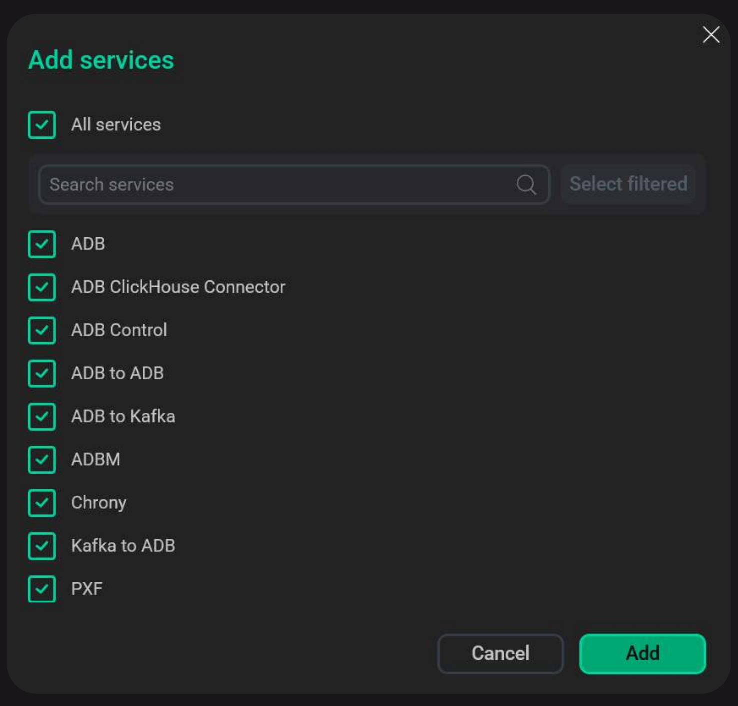
Select a cluster

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



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

☑ ADB
☑ ADB ClickHouse Connector
☑ ADB Control
☑ ADB to ADB
☑ ADB to Kafka
☑ ADBM
☑ Chrony
☑ Kafka to ADB
☑ PXF

Cancel | Add

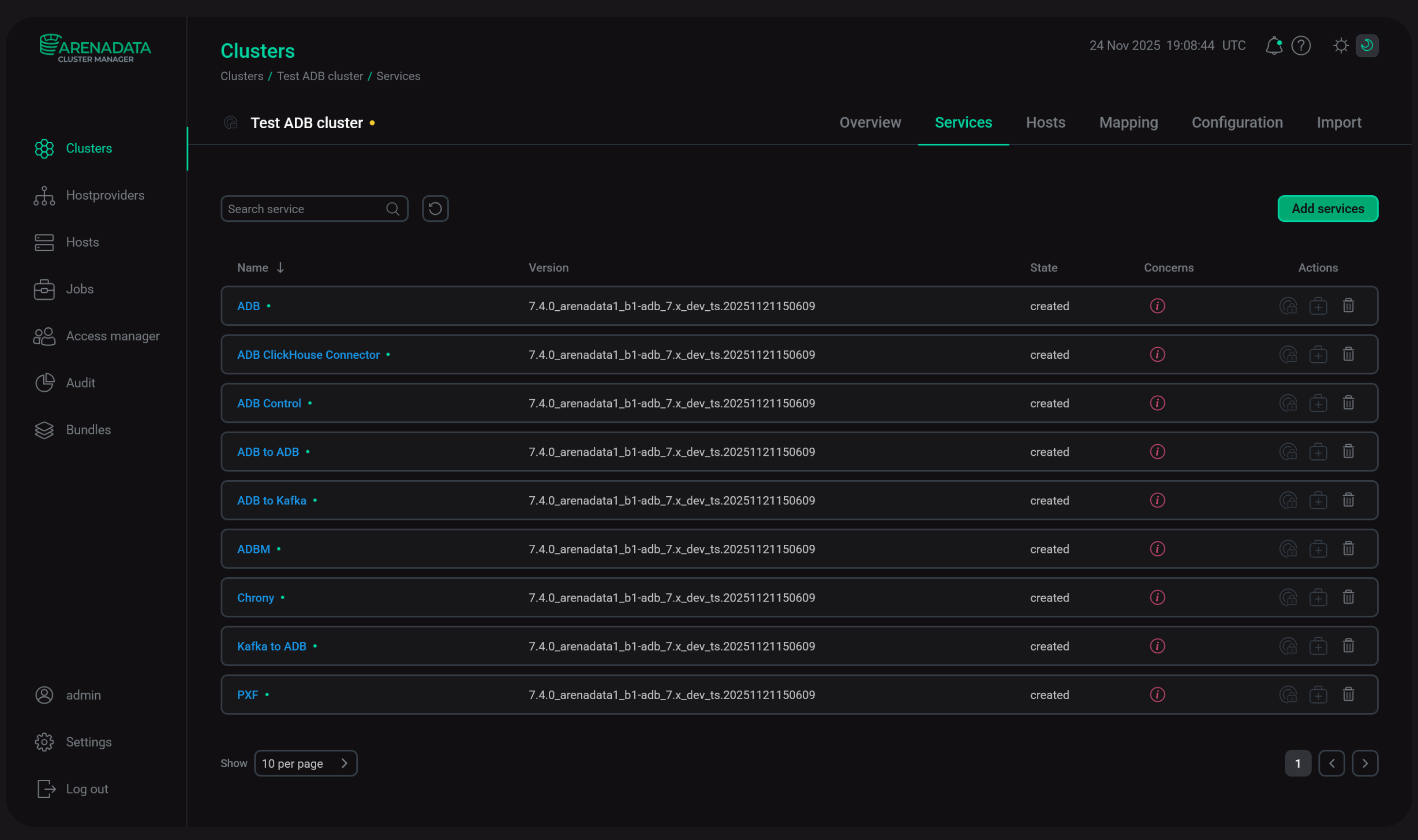
Select services

The brief description of available services is listed below. `CE` means the Common Edition of ADB, `EE` — the Enterprise Edition of ADB.

Services that can be added to the ADB cluster

Service	Mandatory	Edition	Purpose
ADB	Yes	CE/EE	A core service, corresponding to the massively parallel DBMS based on Greengage DB
Chrony	No	CE/EE	A time service. Provides time synchronization for the cluster hosts, that is necessary for the cluster to work correctly. If the time service is already configured on the hosts, there is no need to install Chrony
PXF	No	CE/EE	A service that provides ADB interactions with external data sources
ADB ClickHouse Connector	No	EE	ADB ClickHouse Connector — a service that provides two-way ADB interactions with ClickHouse. Requires PXF
ADB to Kafka	No	EE	ADB to Kafka Connector — a service that provides one-way ADB interactions with Kafka. Requires PXF
Kafka to ADB	No	EE	Kafka to ADB Connector — a service that provides one-way Kafka interactions with ADB
ADB to ADB	No	EE	ADB to ADB Connector — a service that provides two-way interactions between ADB clusters
ADBM	No	EE	Arenadata DB Backup Manager — a service for ADB binary backups management. Can be installed only offline
ADB Control	No	EE	Arenadata DB Control (ADB Control) — a service for monitoring, diagnostics, and management of queries, that run in the ADB cluster. Can be installed only offline

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



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.

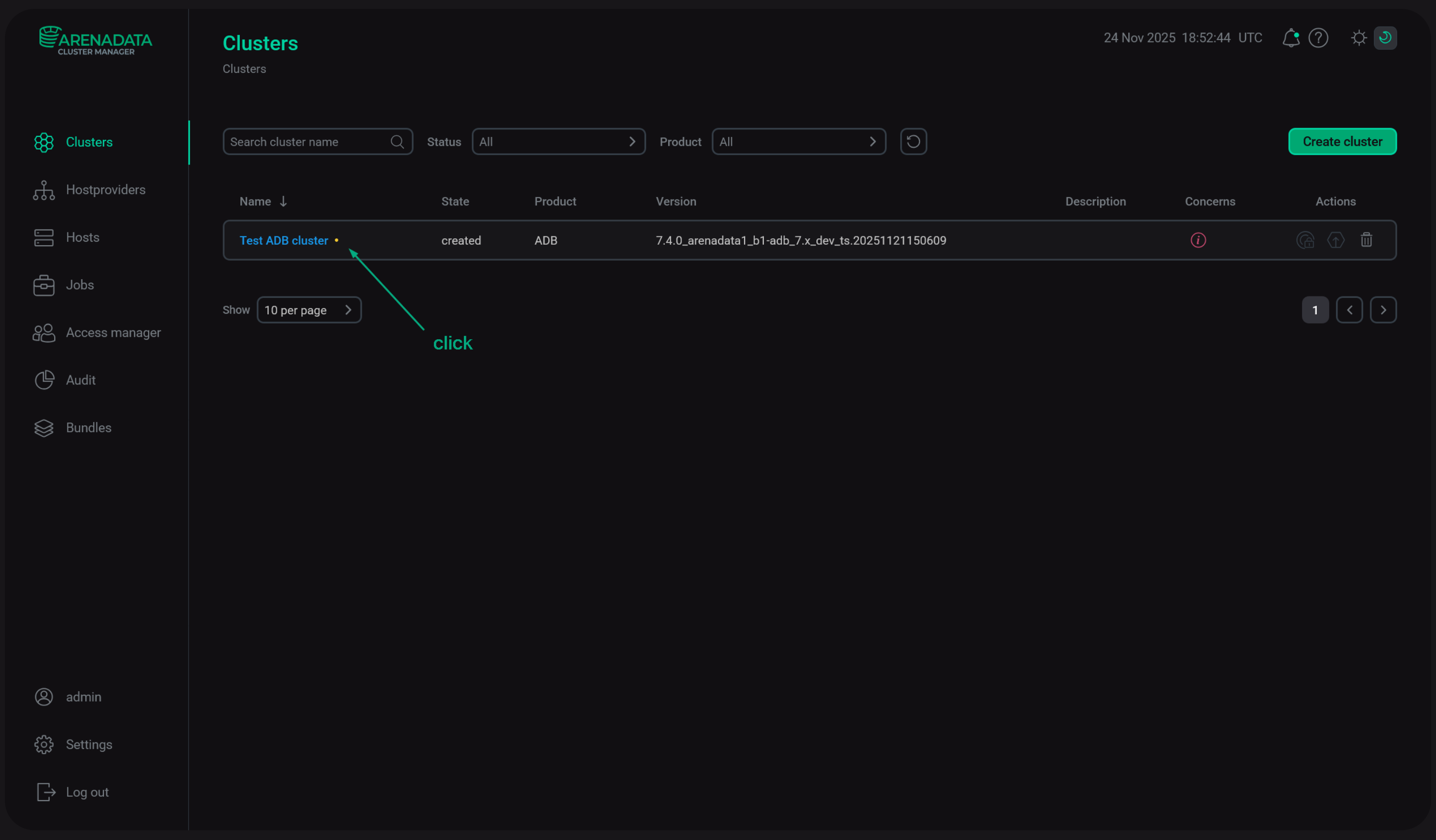
Add hosts to a cluster

Daria Barysheva

To Table of Contents

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

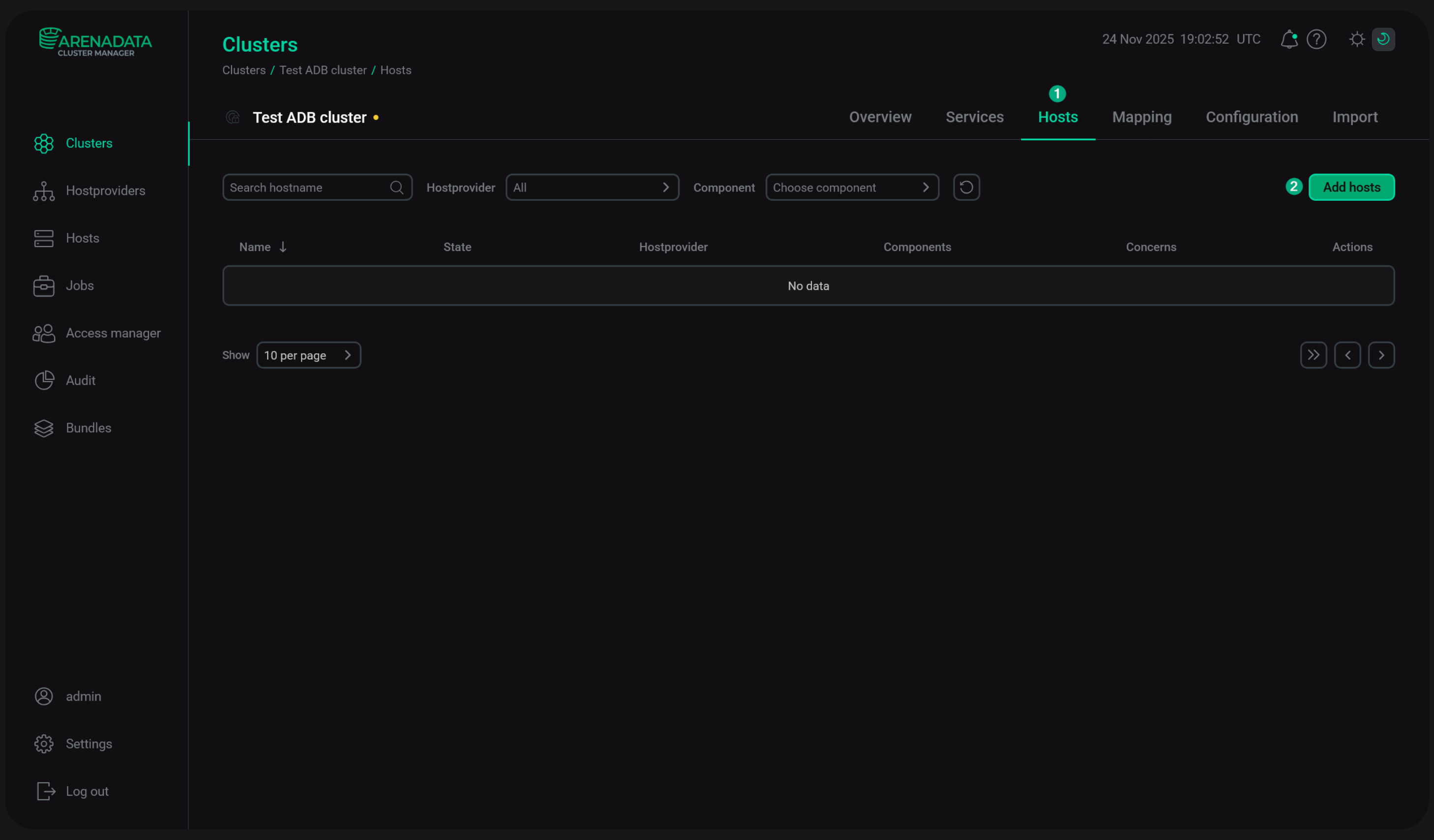
1. Select a cluster on the ***Clusters*** page. To do this, click a cluster name in the ***Name*** column.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
24 Nov 2025 18:52:44 UTC
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
1
click

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

Select a cluster

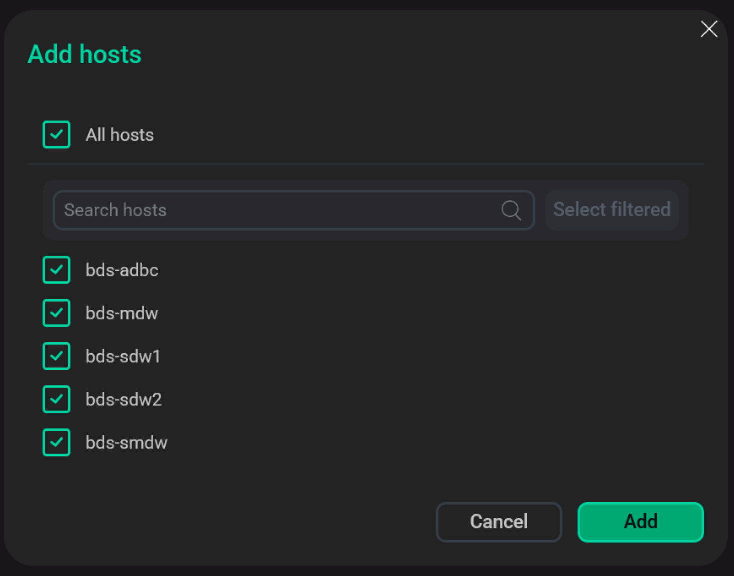
2. Open the ***Hosts*** tab on the cluster page and click ***Add hosts***.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADB cluster / Hosts
24 Nov 2025 19:02:52 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
Search hostname
Hostprovider
All
Component
Choose component
2
Add hosts
Show
10 per page

Name	State	Hostprovider	Components	Concerns	Actions
No data					

Switch to adding hosts

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

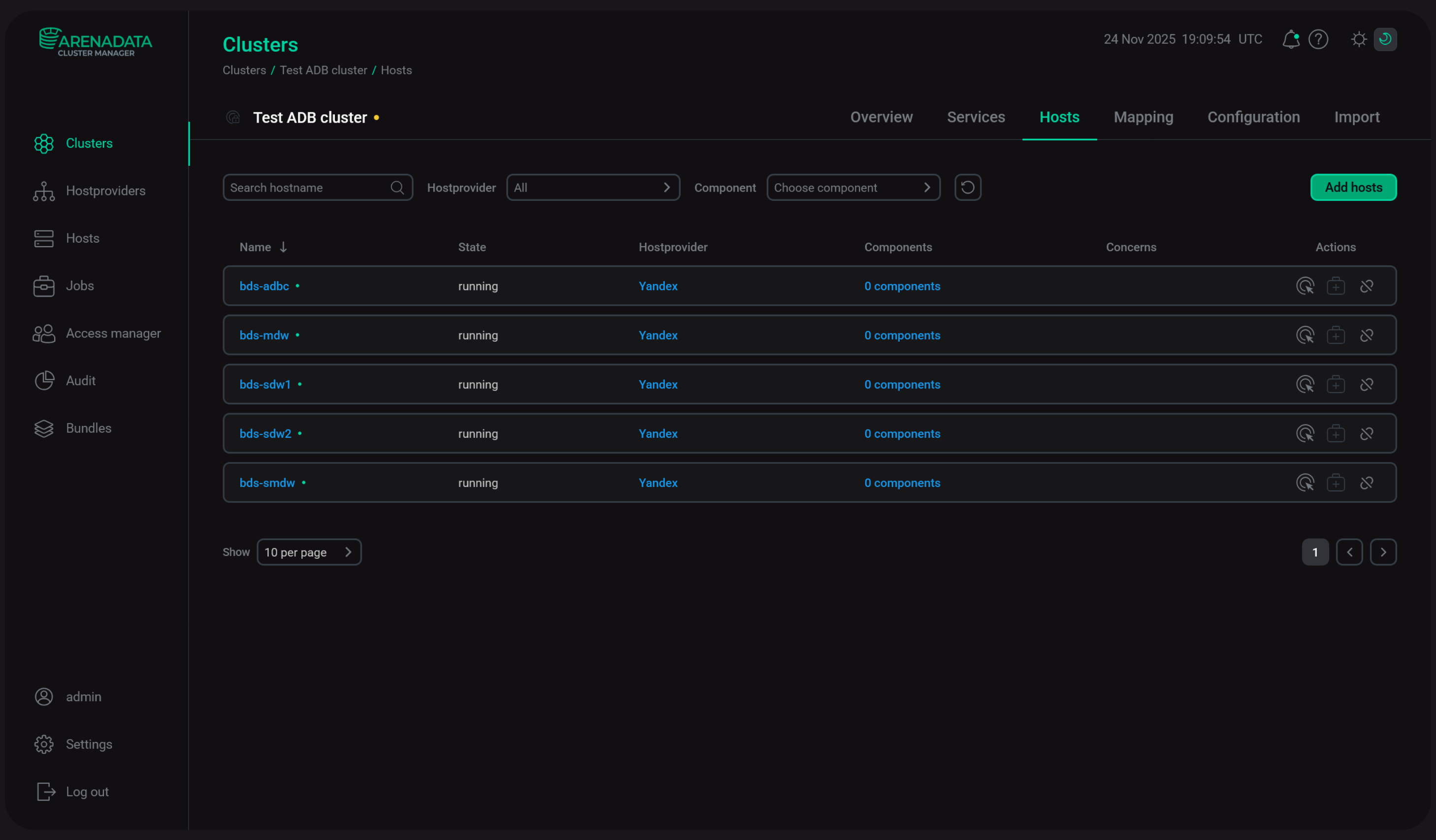

Add hosts
Search hosts
Select filtered
Cancel
Add

☑ All hosts

☑ bds-adbc
☑ bds-mdw
☑ bds-sdw1
☑ bds-sdw2
☑ bds-smdw

Select hosts

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


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADB cluster / Hosts
24 Nov 2025 19:09:54 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Component
Choose component
Add hosts
Show
10 per page
1

Name	State	Hostprovider	Components	Concerns	Actions
bds-adbc	running	Yandex	0 components		
bds-mdw	running	Yandex	0 components		
bds-sdw1	running	Yandex	0 components		
bds-sdw2	running	Yandex	0 components		
bds-smdw	running	Yandex	0 components		

The result of successful adding hosts to a cluster

Add components

Daria Barysheva

To Table of Contents

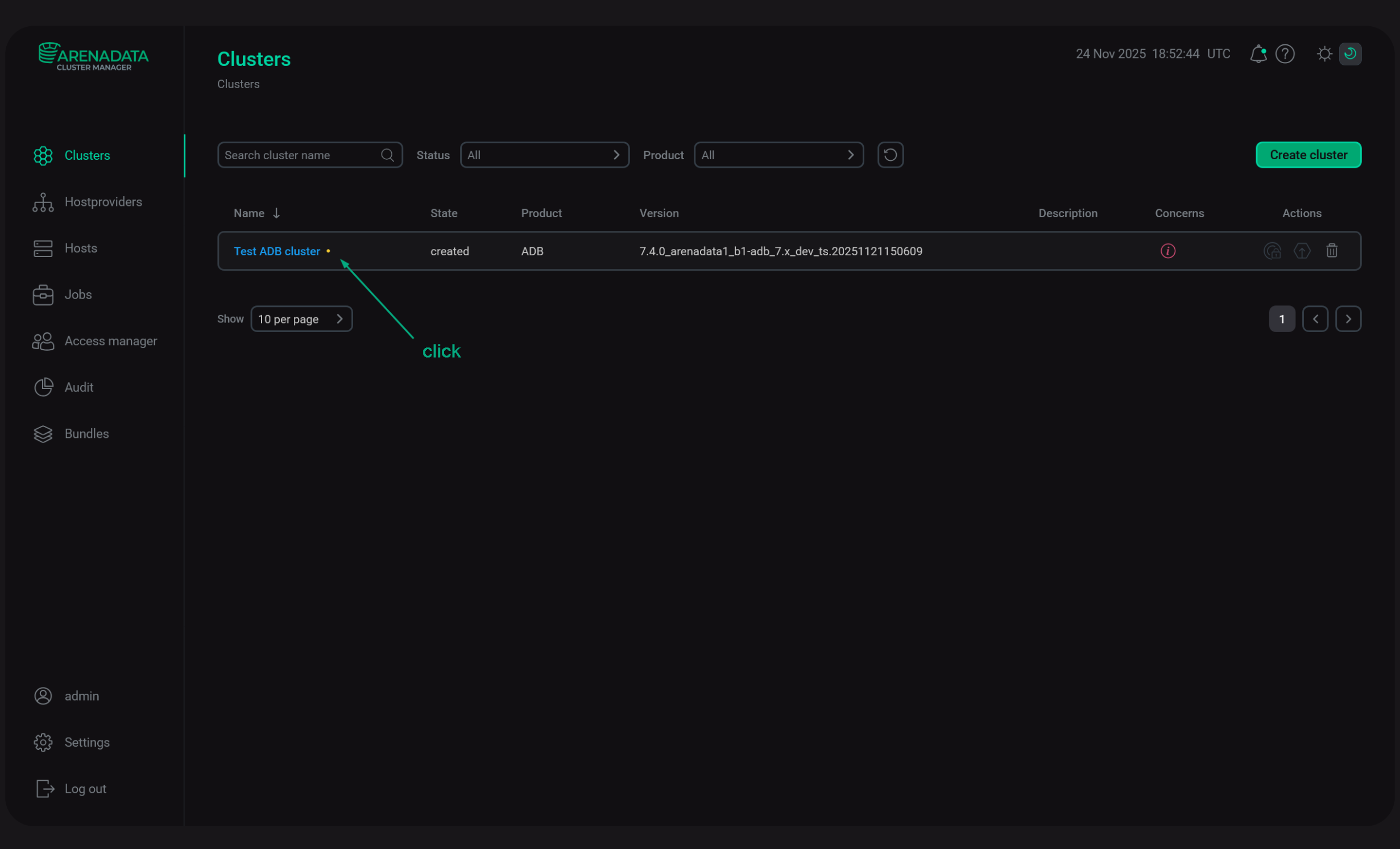
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts. For example, ADB service consists of such components as ADB Master, ADB Segment, ADB Standby, etc.

IMPORTANT

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

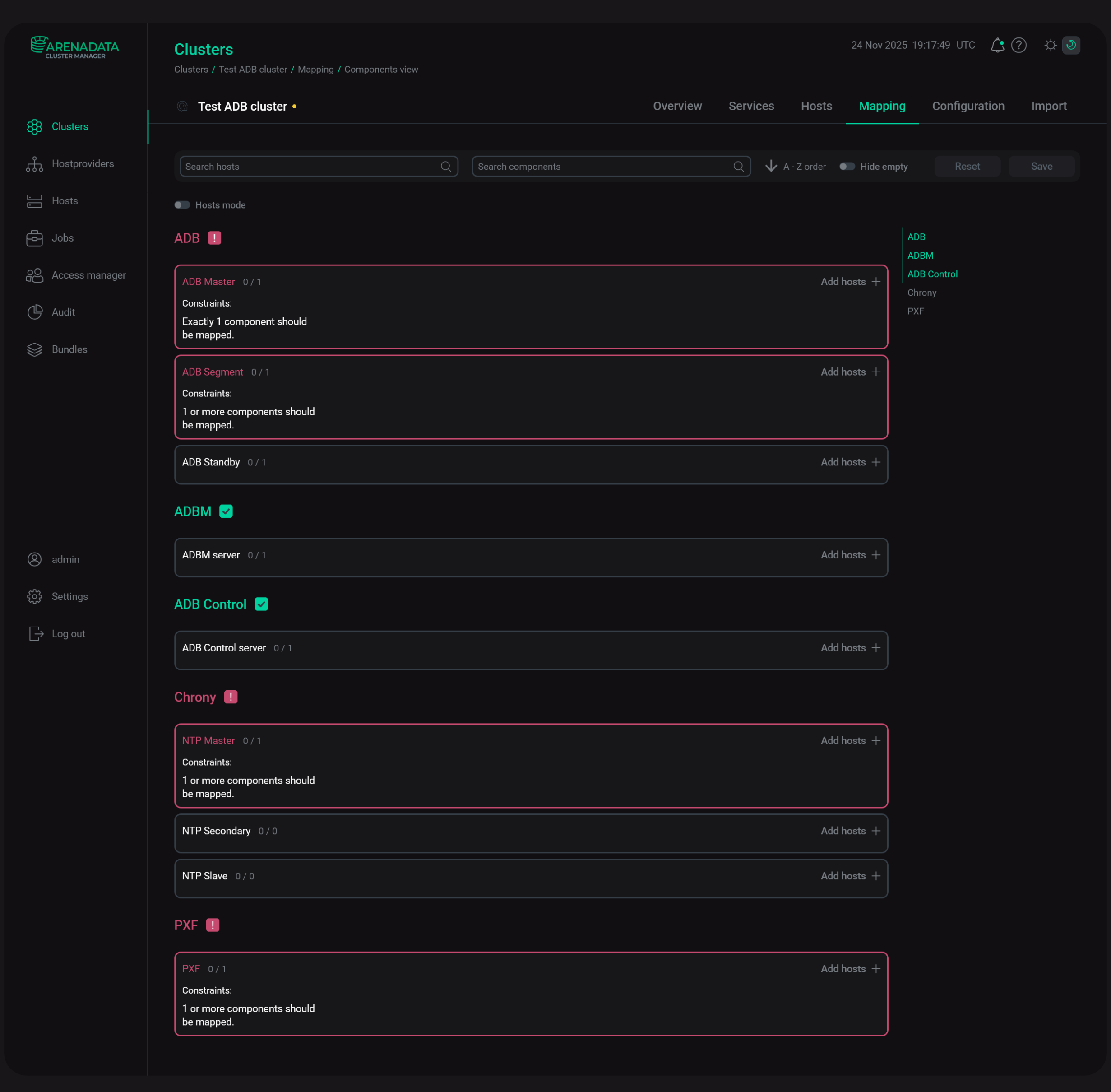
The steps for adding components to hosts are listed below:

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



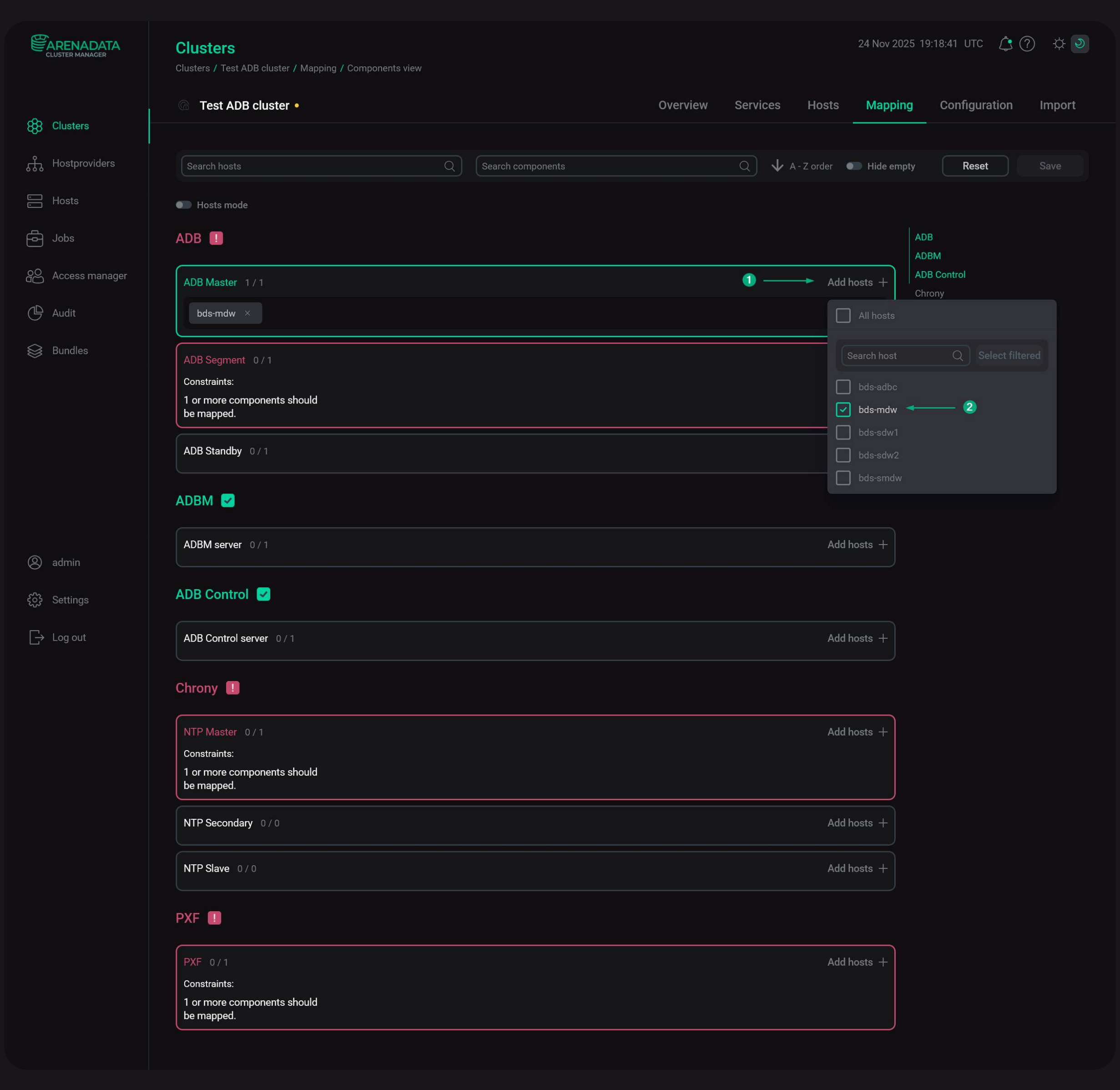
Select a cluster

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



Switch to mapping service components

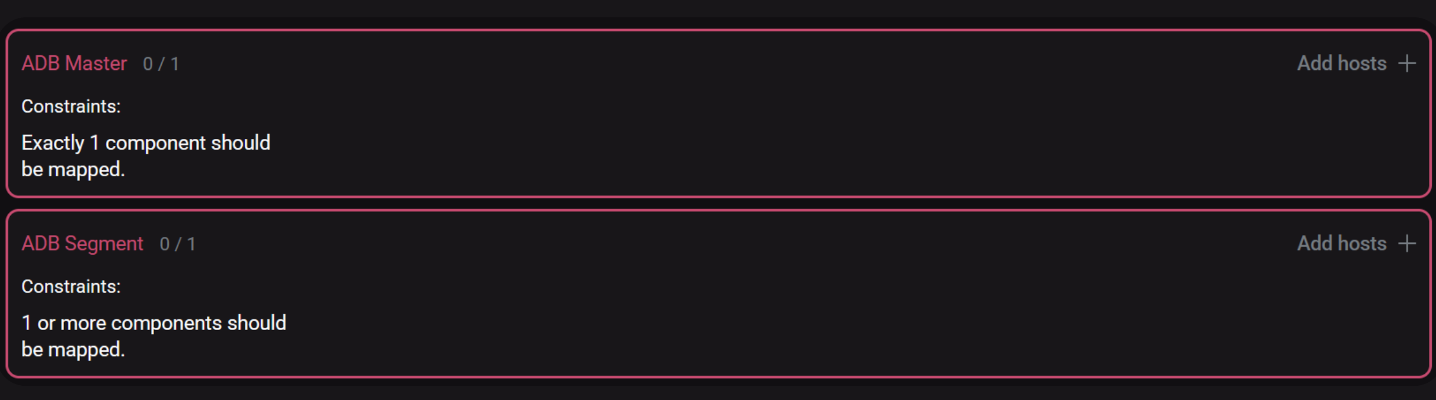
3. In the window that opens, map components to hosts. To assign hosts for each component, click ***Add hosts*** and select hosts in the pop-up window.



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.


ADB Master 0 / 1
Add hosts
Constraints:
Exactly 1 component should be mapped.
ADB Segment 0 / 1
Add hosts
Constraints:
1 or more components should be mapped.

Mandatory components

CAUTION

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

The typical scheme of mapping components in ADB clusters is described in the table below. It is based on the Enterprise Edition of ADB.

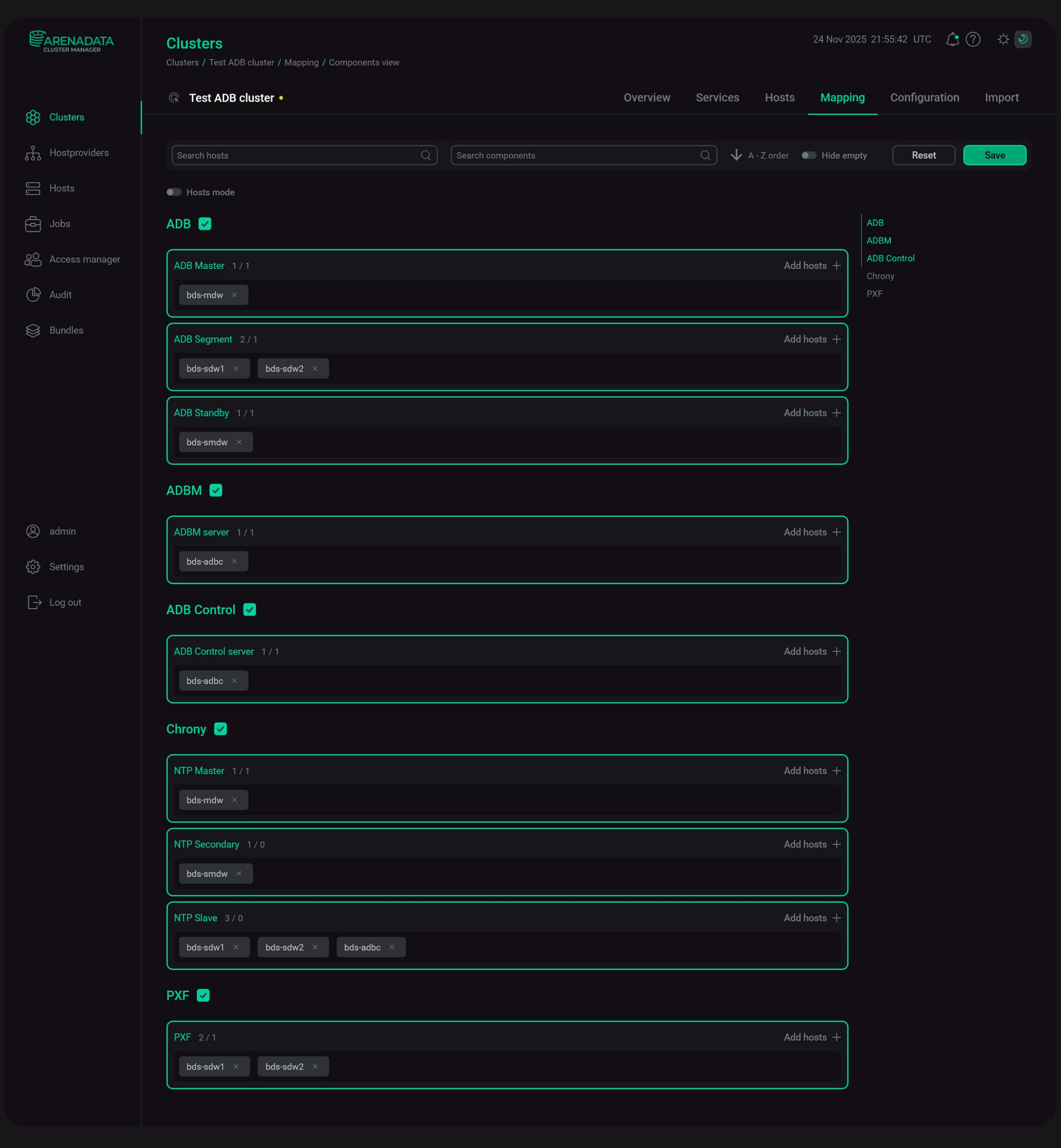
Example of mapping ADB components to hosts

Host	Service	Component	Notes
Coordinator host	ADB	ADB Master	Should be installed on the coordinator host
	Chrony	NTP Master	Should be installed on the coordinator host
Secondary Master Host (Standby)	ADB	ADB Standby	Should be installed on the standby coordinator host
	Chrony	NTP Secondary	Should be installed on the standby coordinator host
Segment host (1..N)	ADB	ADB Segment	Should be installed on each segment host
	Chrony	NTP Slave	Should be installed on each segment host
	PXF	PXF	Should be installed on one or more segment hosts and, optionally, on the coordinator host
ADB Control/ADBM host	ADB Control	ADB Control server	It is recommended to use a separate host for this component
	ADBM	ADBM server	Can be installed on the same host with ADB Control server
	Chrony	NTP Slave	Should be installed on the ADB Control/ADBM host

CAUTION

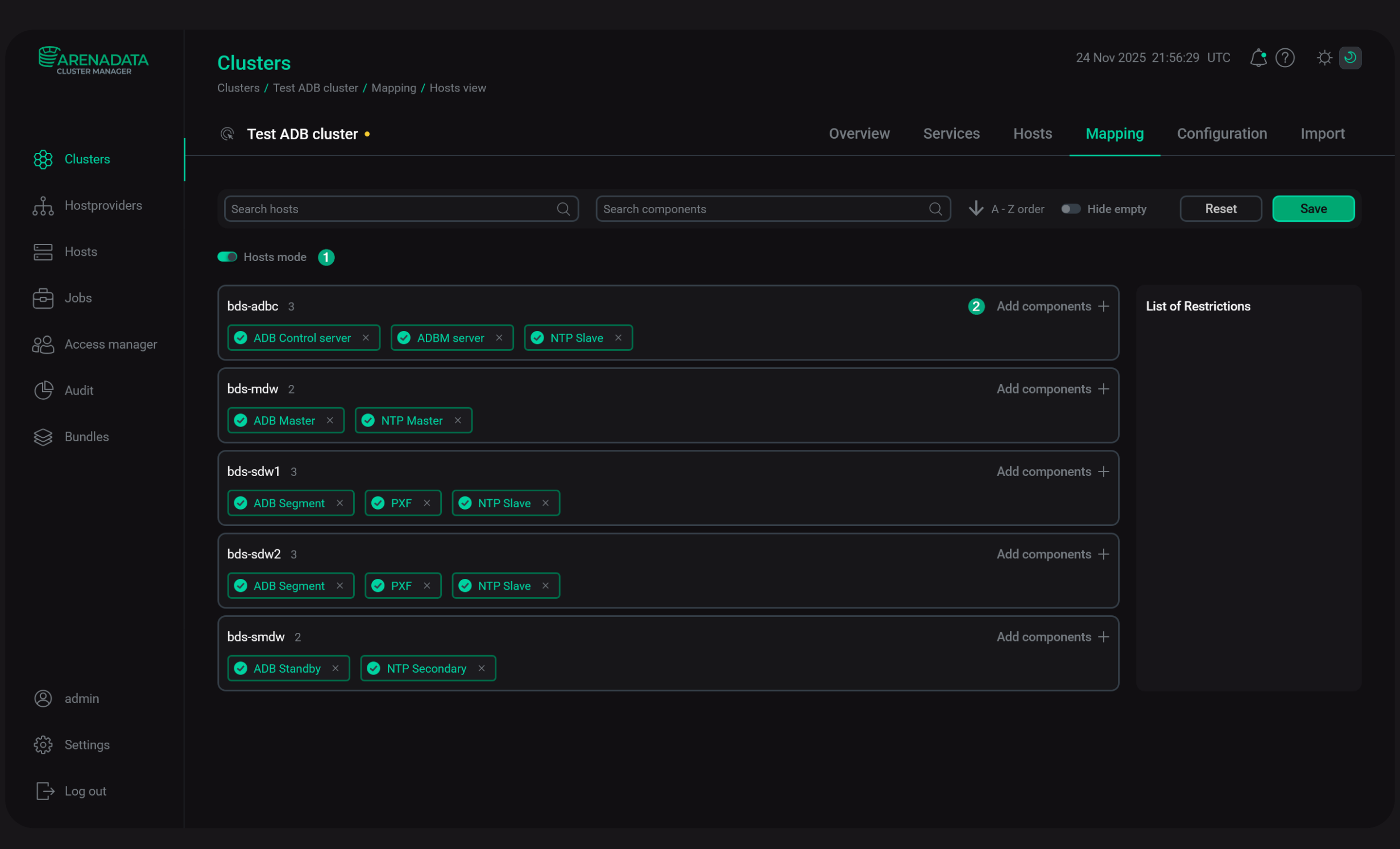
Do not install ADB Control server or ADBM server on the same host with the ADB Master or ADB Segment components. This feature is reserved for informational purposes only. Use a separate host for these components.

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.



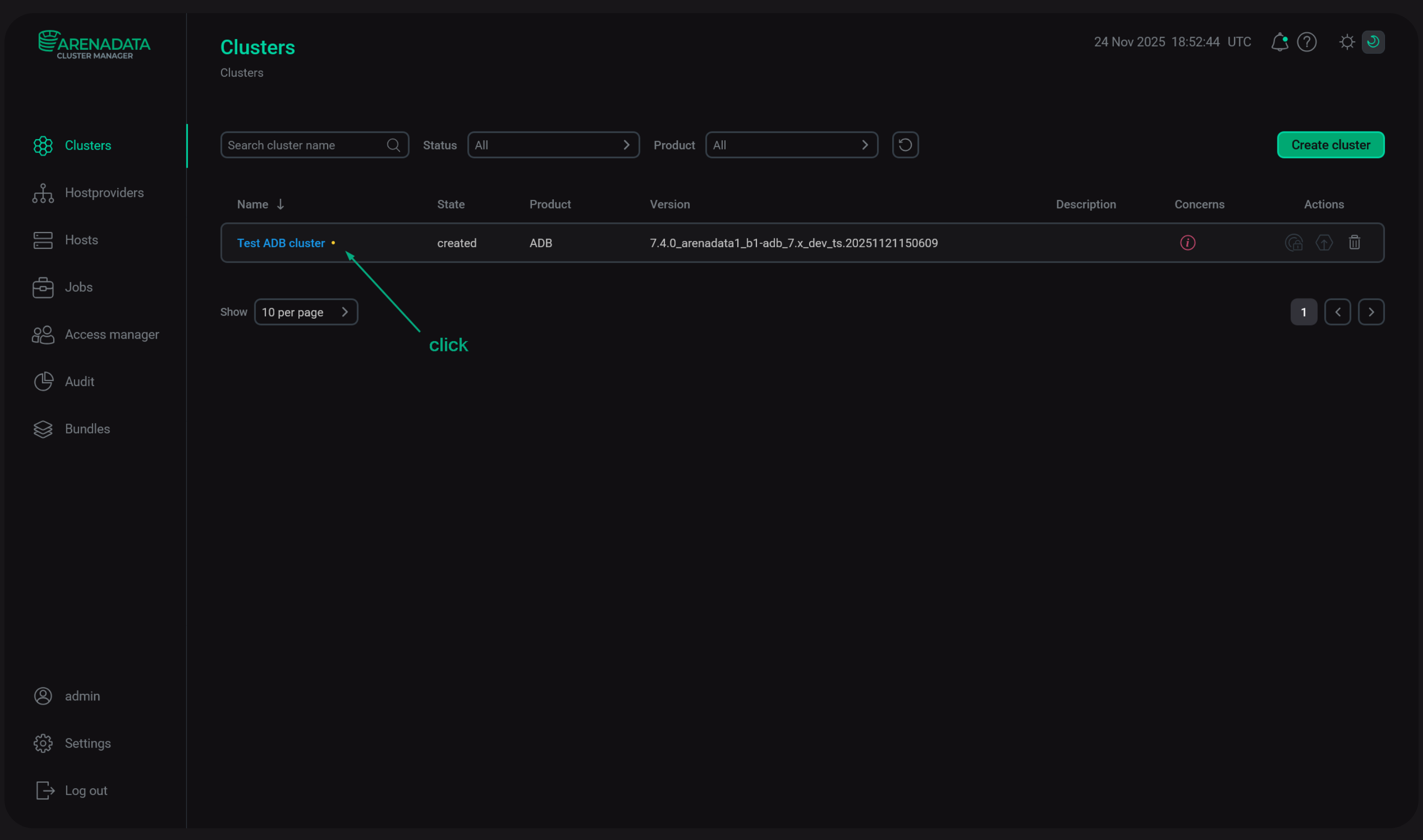
Use mapping by hosts

Configure services

Daria Barysheva

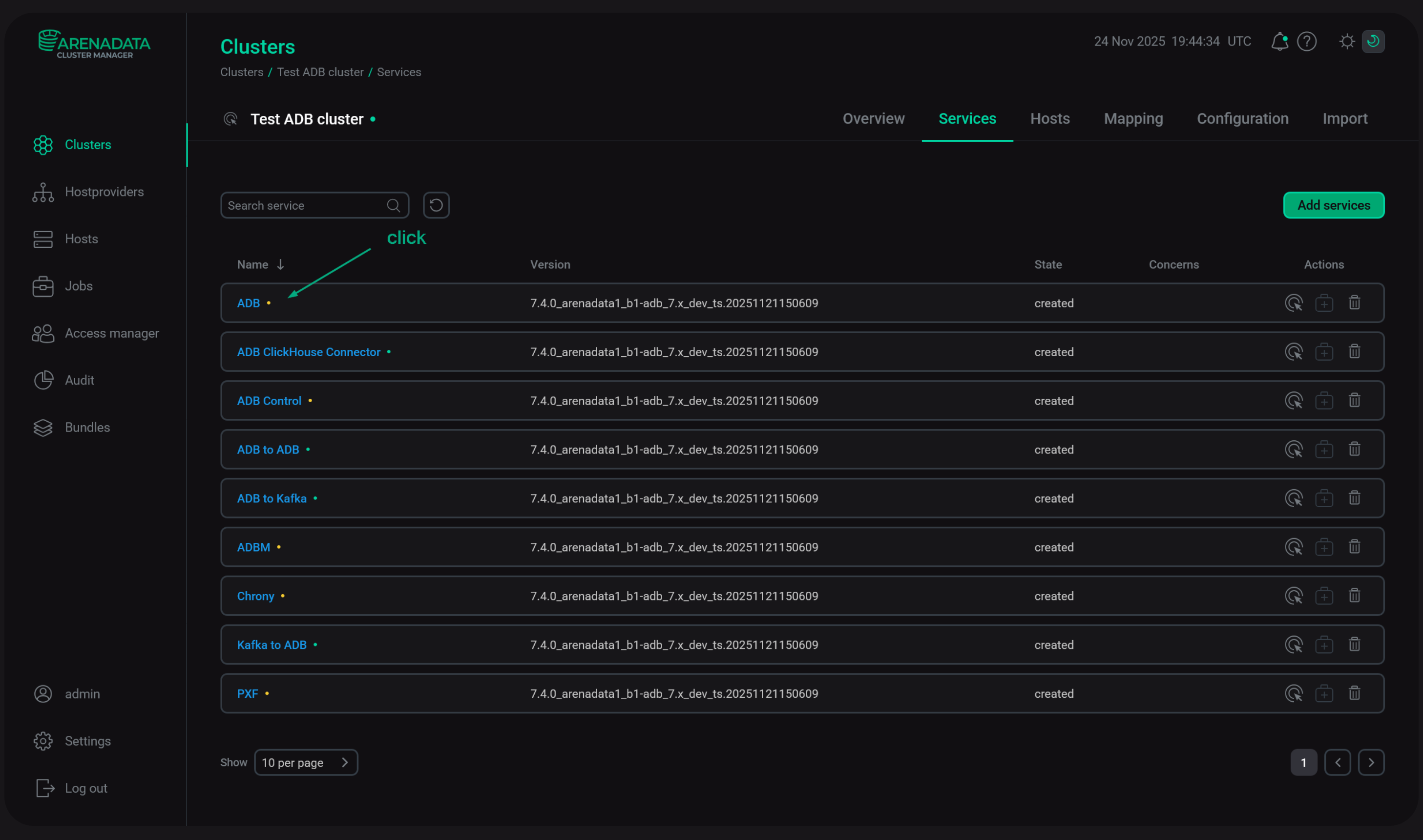
To Table of Contents

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



Select a cluster

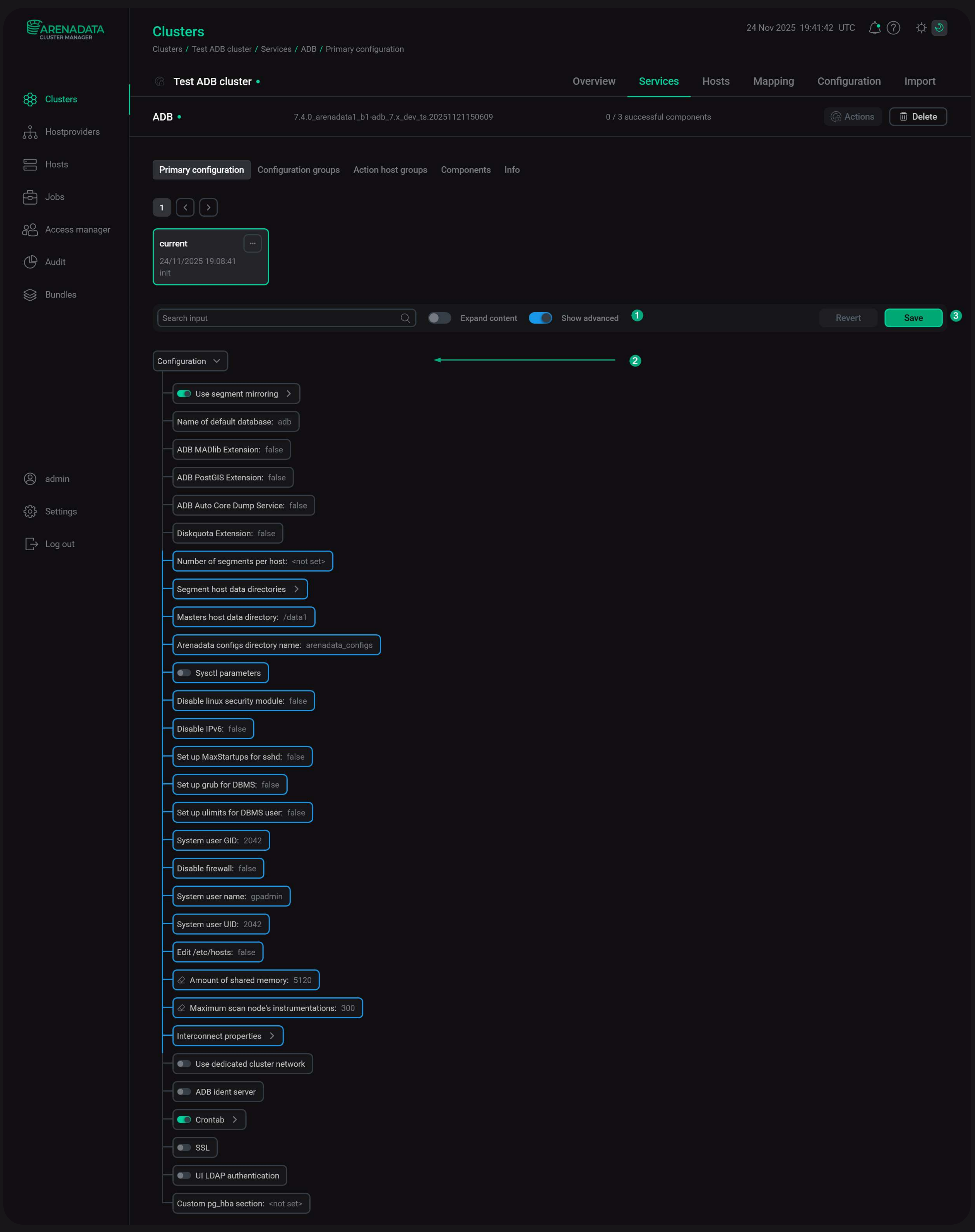
2. Open the ***Services*** tab on the cluster page, find the row that contains the desired service, and click its name in the ***Name*** column.



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***.



Configure a service

Pay attention to the following parameters:

- ***ADB → Edit /etc/hosts***. Set to `true` to let ADB modify the */etc/hosts* file to include hostnames of all cluster hosts. Otherwise, you should modify this file manually or configure a DNS server. Using the ADB service settings, you can also adjust other parameters of the cluster hosts to align them with recommended values.
- ***ADB → Disable linux security module*** and ***ADB → Set up GRUB for DBMS***. Changing these parameters requires host reboot, so it is recommended to set them to `true` before cluster installation and then set ***Reboot cluster servers after installation*** to `true` when launching the Install cluster action to reboot hosts automatically. If you change these parameters after installation, you will have to stop the cluster and reboot hosts manually.
- ***Chrony → NTP servers***. Though this parameter is optional, we recommend to set it to the nearest NTP server, e.g. the one that your company uses.

NOTE

- For more information about services configuration parameters, see Configuration parameters.
- If you plan to use external databases for ADB Control and/or ADBM, you need to perform additional configuration steps before installation of these services:
  - ADB Control
  - ADBM

Configure a cluster

Daria Barysheva

Contents

To Table of Contents

Primary configuration

Configuration groups

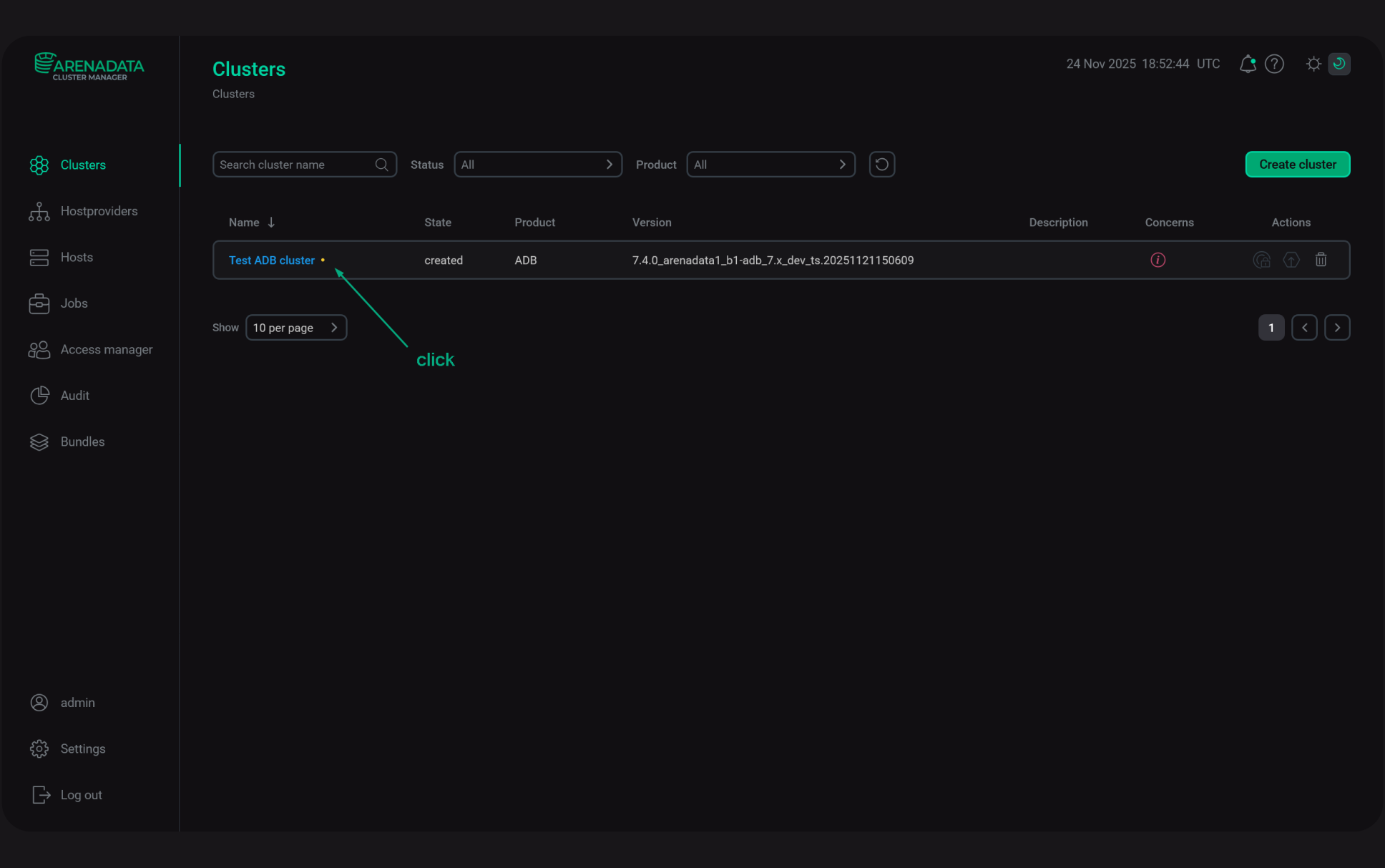
Ansible settings

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.



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.

CAUTION

Use all custom installations (***Use existing docker***, ***Use custom registry***, ***Use custom JAVA_HOME for cluster***) on your own risk.

Available configuration parameters are listed below:

- ***Cluster UUID*** — the unique cluster identifier, which is generated automatically during ADB installation.
- ***Timeout to unlock package manager*** — parameters of this section are intended to set the timeout of the DNF software package manager unlocking wait. In Ubuntu, after the host reboot, automatic update starts, which locks DNF.

Name	Description	Default
retries	The maximum number of retries	3
delay	The time delay between retries (in seconds)	10

- Paths to the repositories that are used during the installation process depending on the cluster operating system (AltLinux, Astra Linux, RED OS, and Ubuntu). Only the marked repositories are registered on the cluster hosts.

Altlinux 10 repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

Astra linux repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

redos 7 repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

Ubuntu 22.04 repositories

Parameter	Description
Set up Arenadata repo	Whether or not to use the repository that is described in the field ***Arenadata repo url***
Arenadata repo url	The repository that is required to install ADB
Set up custom Arenadata enterprise repo	Whether or not to use the repository that is described in the field ***Custom Arenadata enterprise repo***
Custom Arenadata enterprise repo	Custom Arenadata Enterprise repository. If not filled in, the default repository will be used
Set up ADB monitoring repo	Whether or not to use the repository that is described in the field ***prometheus_repo*** (for future use)
prometheus_repo	The repository that is required to install monitoring (for future use)

- ***Use existing docker*** — a flag that indicates whether to use the existing Docker during the installation of ADB services (e.g. ADB Control, ADBM, Monitoring, Enterprise Tools). If this flag is set, only containers will be deployed, while Docker and Docker Compose installation will be skipped. For more information on Docker requirements, see Software requirements.

IMPORTANT

Starting with ADB 6.27.1.56, ADB Control rewrites Docker settings in the */etc/docker/daemon.json* file for security purposes. Thus, if you need to modify this file, set the ***Use existing docker*** option when configuring ADB. This option will also allow you to use Docker in the rootless mode.

- ***Use custom registry*** — a switcher that determines whether to use a custom registry. When the switcher is activated, the following parameters are available.

Parameter	Description
registry	A custom registry URL in the following form: `<host>:<port>`
prefix	An optional subdirectory name that will be added to the custom registry URL: `<registry>/<prefix>`
Use self-signed certificate	Whether or not to use a self-signed certificate
Self-signed certificate	A self-signed certificate text

- ***Use custom JAVA_HOME for cluster*** — a switcher that indicates whether to use a custom Java path for ADB services (e.g. PXF, ADB Control, ADBM). Displayed if the ***Show advanced*** flag is set. When the switcher is activated, the ***JAVA_HOME*** field becomes available. Enter a work Java path in that field. The selected path should contain a *bin* folder with a Java binary file, for example */usr/lib/jvm/java-17-openjdk-amd64*. Installation will fail if there is no Java at the specified path. For more information on Java requirements, see Software requirements.
  If the ***JAVA_HOME*** value is empty, one of the following default paths is used (depending on the operating system).

Ubuntu

/usr/lib/jvm/java-17-openjdk-amd64

RED OS

/usr/lib/jvm/jre-17-openjdk

AltLinux

/usr/lib/jvm/jre-17

IMPORTANT

- If you plan to use custom Java (such as Axiom), Java 17 is required for installation.
- For Astra Linux, the ***JAVA_HOME*** field is mandatory starting with ADB 6.27.1.58.

- ***Parameter archiving*** — a switcher that indicates whether to export ADB settings according to the specified schedule. When the switcher is activated, the ***Parameter archiving job setting*** field becomes available. Enter a schedule for an export job in that field. The field value should be a valid `cron` expression. After you edit and save the ***Parameter archiving job setting*** value, run the cluster action Reconfigure parameter archiving.
  Each time a `cron` job is run, ADB settings are exported to the following directory:

```
/home/gpadmin/arenadata_configs/settings/<timestamp>
```

where `<timestamp>` is a timestamp of the export job launch in the `YYYY-MM-DD_HH-mm-ss` format.

A job execution log is saved to the *parameter_archiving.log* file.

Currently, the following settings are offloaded.

Service	Exported parameters	Relative path
ADB	GUC values for coordinator/segment	./<YYYY-MM-DD_HH-mm-ss>_guc_config.json
	Linux Greenplum-related parameters that are exported from */etc/sysctl.conf*	./<YYYY-MM-DD_HH-mm-ss>_os_config.json
	Resource group parameters	./<YYYY-MM-DD_HH-mm-ss>_rg_config.json
ADB Control	Configuration parameters of ADB Control components	./adcc/*
ADBM	Configuration parameters of ADBM components	./adbm/*
PXF	PXF settings that are exported from: */etc/pxf/conf/pxf-application.properties* */etc/pxf/conf/pxf-profiles.xml* *$PXF_BASE/servers/**	./pxf_config/*

- ***Vault integration*** — a switcher that indicates whether to use HashiCorp Vault to store secrets of ADB services. When the switcher is activated, the following parameters become available.

Name	Description	Default
Address of the Vault server	The Vault server address (e.g. `192.168.1.120` or `vault.domain.com`)	—
Port of the Vault server	The Vault server port. Should be specified if the service is published on a port other than HTTPS (443)	443
Internal CA certificate in PEM format (optional)	The internal CA certificate in PEM format that signed the Vault server's certificate	—
Use PEM CA for vault	Defines whether to use the CA certificate in PEM format. If you set this parameter to `true`, you also need to fill in ***Internal CA certificate in PEM format (optional)***. To hide the truststore password in Vault, set ***Use PEM CA for vault*** to `true`	false
Vault namespace (optional)	The Vault namespace	—

Writer authentication method for the Vault server	The authentication method that is used to connect to the Vault server with write access.
	Possible values:
	`token`
	`approle`
		approle

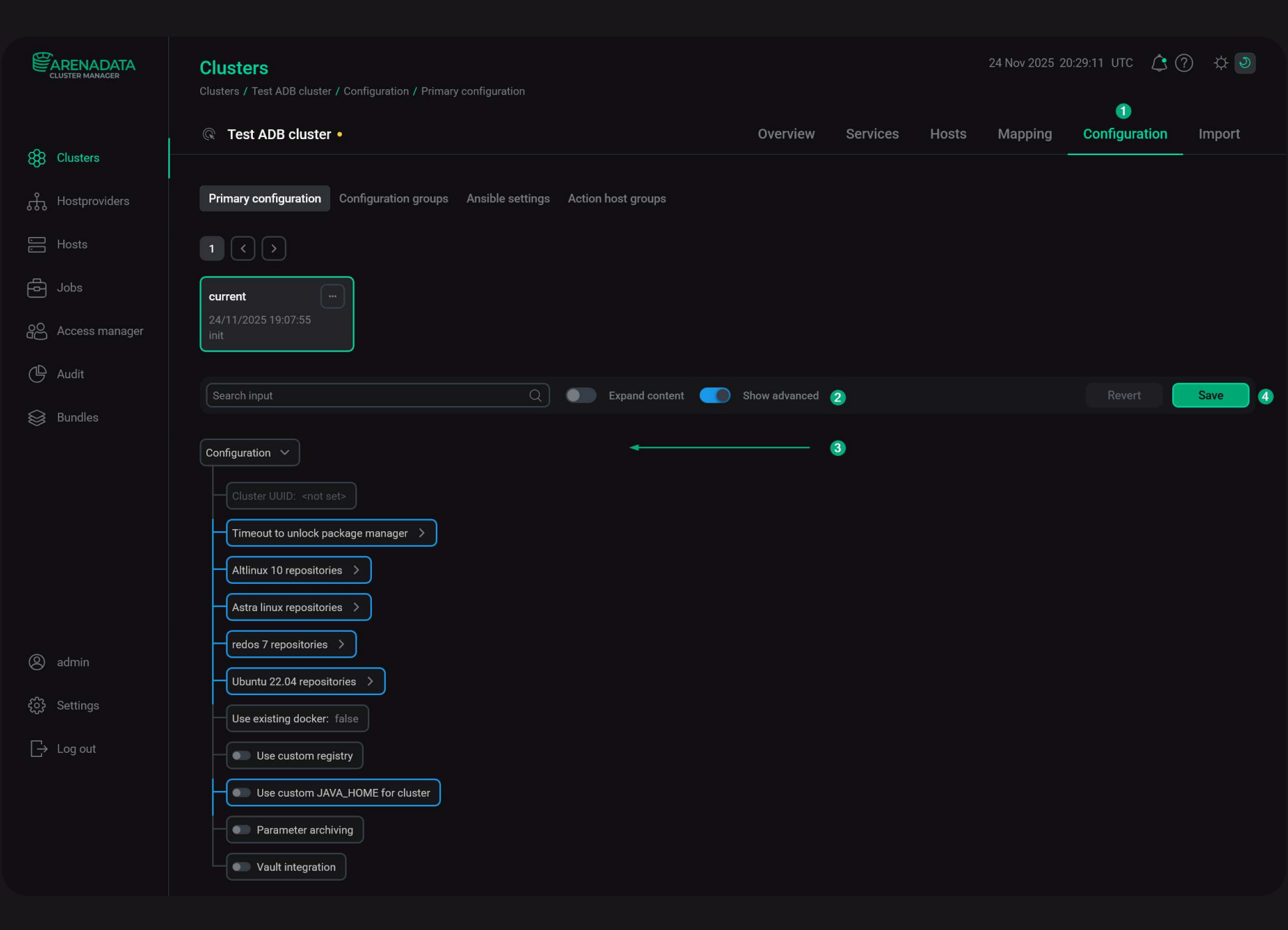
Writer Vault token	The application token that is used to connect to the Vault server with write access. Required only if the ***Writer authentication method for the Vault server*** parameter is set to `token`	—
Writer Vault role ID	The role identifier provided for applications to connect to the Vault server with write access. Required only if the ***Writer authentication method for the Vault server*** parameter is set to `approle`	—
Writer Vault secret ID	The secret identifier associated with the ***Writer Vault role ID*** role that is used by applications to connect to the Vault server with write access. Required only if the ***Writer authentication method for the Vault server*** parameter is set to `approle`	—

Reader authentication method for the Vault server	The authentication method that is used to connect to the Vault server with read access.
	Possible values:
	`token`
	`approle`
		approle

Reader Vault token	The application token that is used to connect to the Vault server with read access. Required only if the ***Reader authentication method for the Vault server*** parameter is set to `token`	—
Reader Vault role ID	The role identifier provided for applications to connect to the Vault server with read access. Required only if the ***Reader authentication method for the Vault server*** parameter is set to `approle`	—
Reader Vault secret ID	The secret identifier associated with the ***Reader Vault role ID*** role that is used by applications to connect to the Vault server with read access. Required only if the ***Reader authentication method for the Vault server*** parameter is set to `approle`	—
Mount point for ADB cluster secrets	A mount point that is accessible for read/write operations with the provided ADB cluster authentication parameters. Each service will store secrets on the Vault server at the following path: *<Mount point for ADB cluster secrets>/adb/<cluster_name>/service/<service_name>/<component>*	—
Login mount point	A custom authentication mount point. If empty, equals to the authentication method selected in the ***Writer authentication method for the Vault server*** (write) or ***Reader authentication method for the Vault server*** (read) field	—

IMPORTANT

- Before editing Vault integration parameters, ensure that Vault is installed and configured.
- Each time you edit and save parameters in the ***Vault integration*** section, run the cluster action Reconfigure Vault integration.
- Currently, Vault can be used to store authentication parameters of ADB Control and ADBM.
- If you need to store credentials for the ADB Control Query DB and Metrics DB in Vault, set up these databases as external.



Cluster configuration window

Configuration groups

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

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

Daria Barysheva

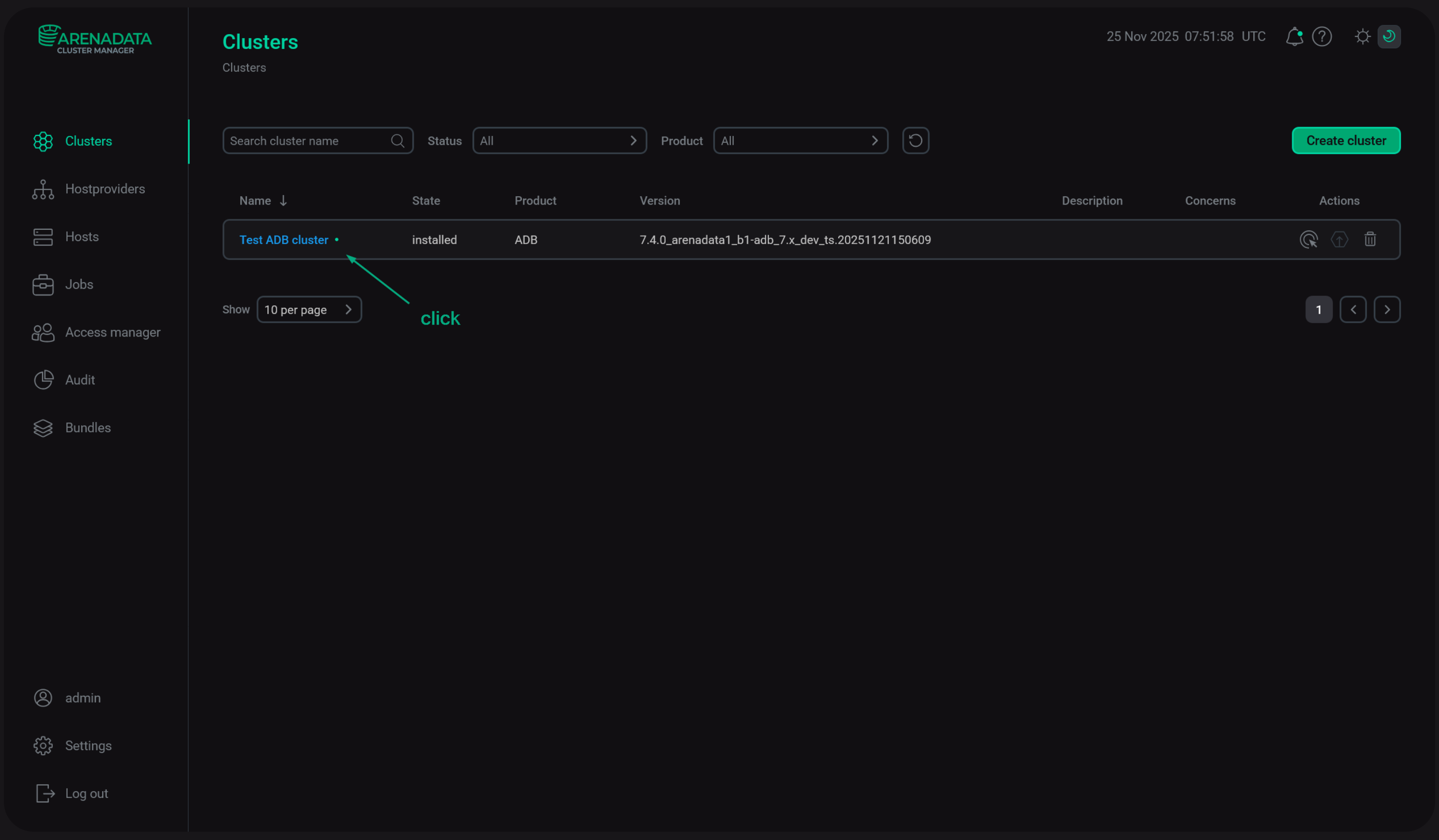
To Table of Contents

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.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters
25 Nov 2025 07:51:58 UTC
Search cluster name
Status
All
Product
All
Create cluster
Show
10 per page
click
1

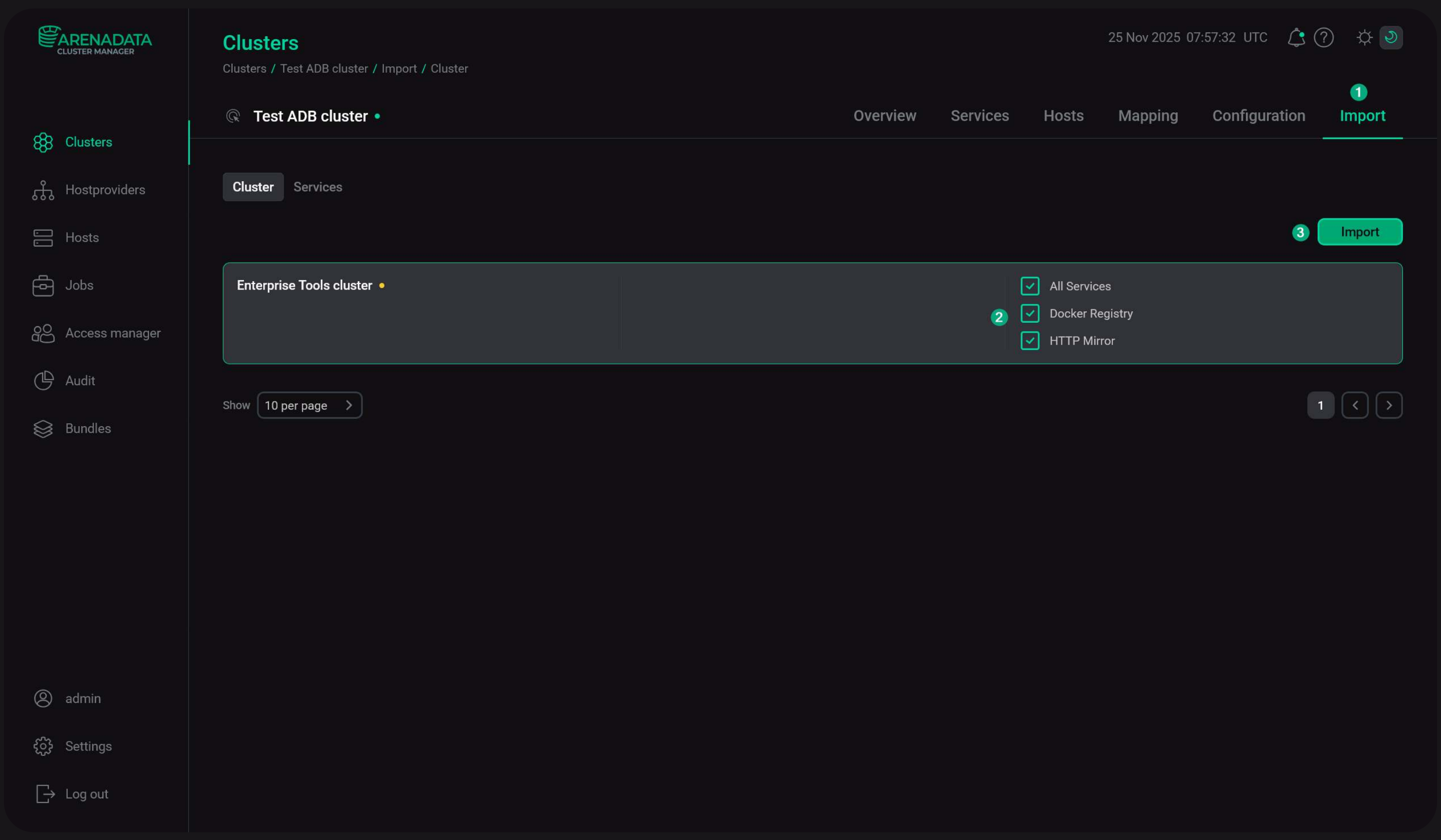
Name ↓	State	Product	Version	Description	Concerns	Actions
Test ADB cluster •	installed	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

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***.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADB cluster / Import / Cluster
25 Nov 2025 07:57:32 UTC
Test ADB cluster •
Overview
Services
Hosts
Mapping
Configuration
1
Import
Cluster
Services
3
Import
Enterprise Tools cluster •
2
☑ All Services
☑ Docker Registry
☑ HTTP Mirror
Show
10 per page
1

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

Daria Barysheva

Contents

To Table of Contents

Step 1. Precheck

Step 2. Run installation

- Whole cluster
- Single services

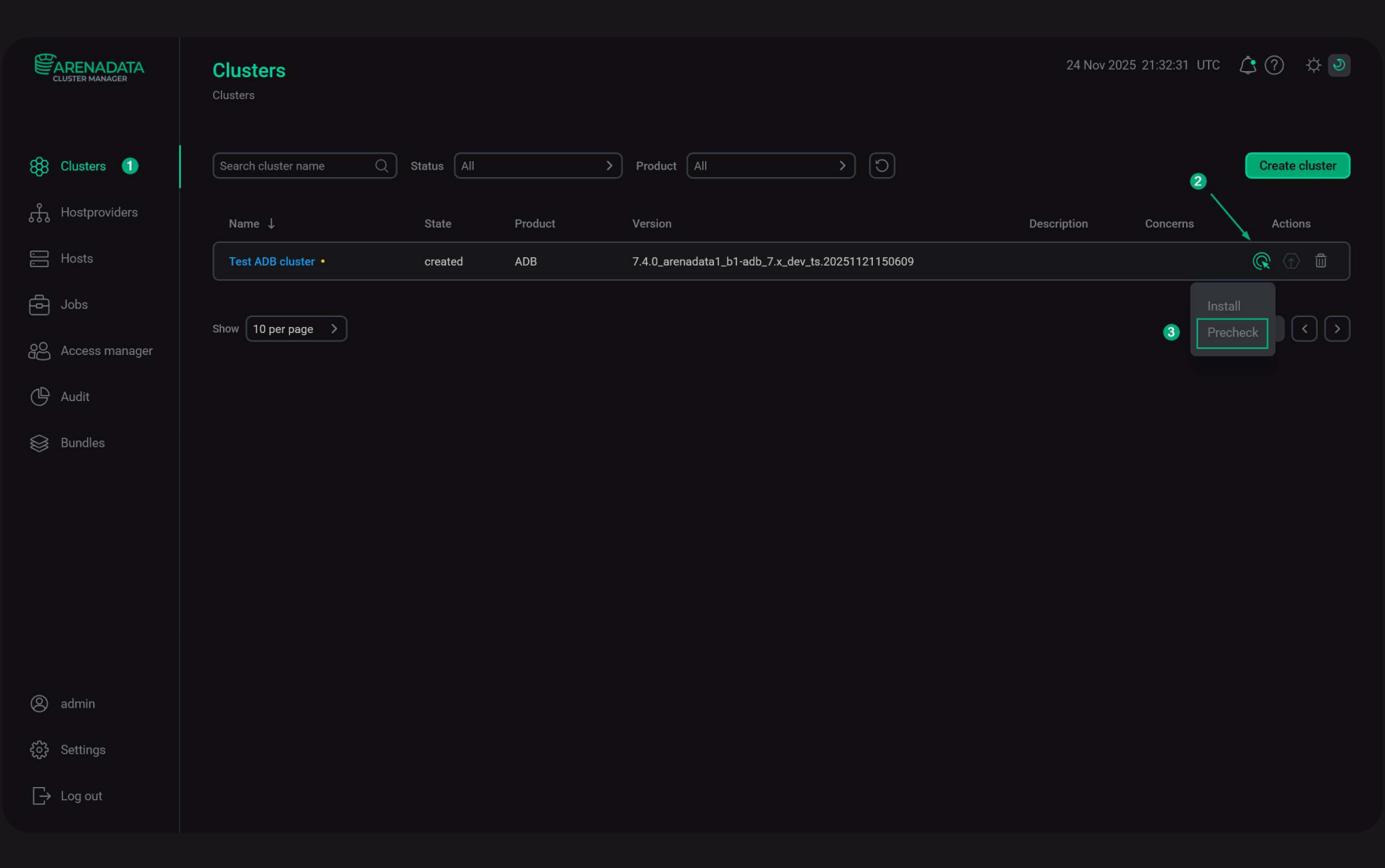
Step 3. Check the cluster state after installation

Step 4. Check the default database

Step 1. Precheck

Before installing a new cluster it is recommended to run the ***Precheck*** action. By doing this, you make sure that all necessary repositories are available, the cluster does not contain hosts with the same IP addresses, and all other prerequisites are met. Do the following:

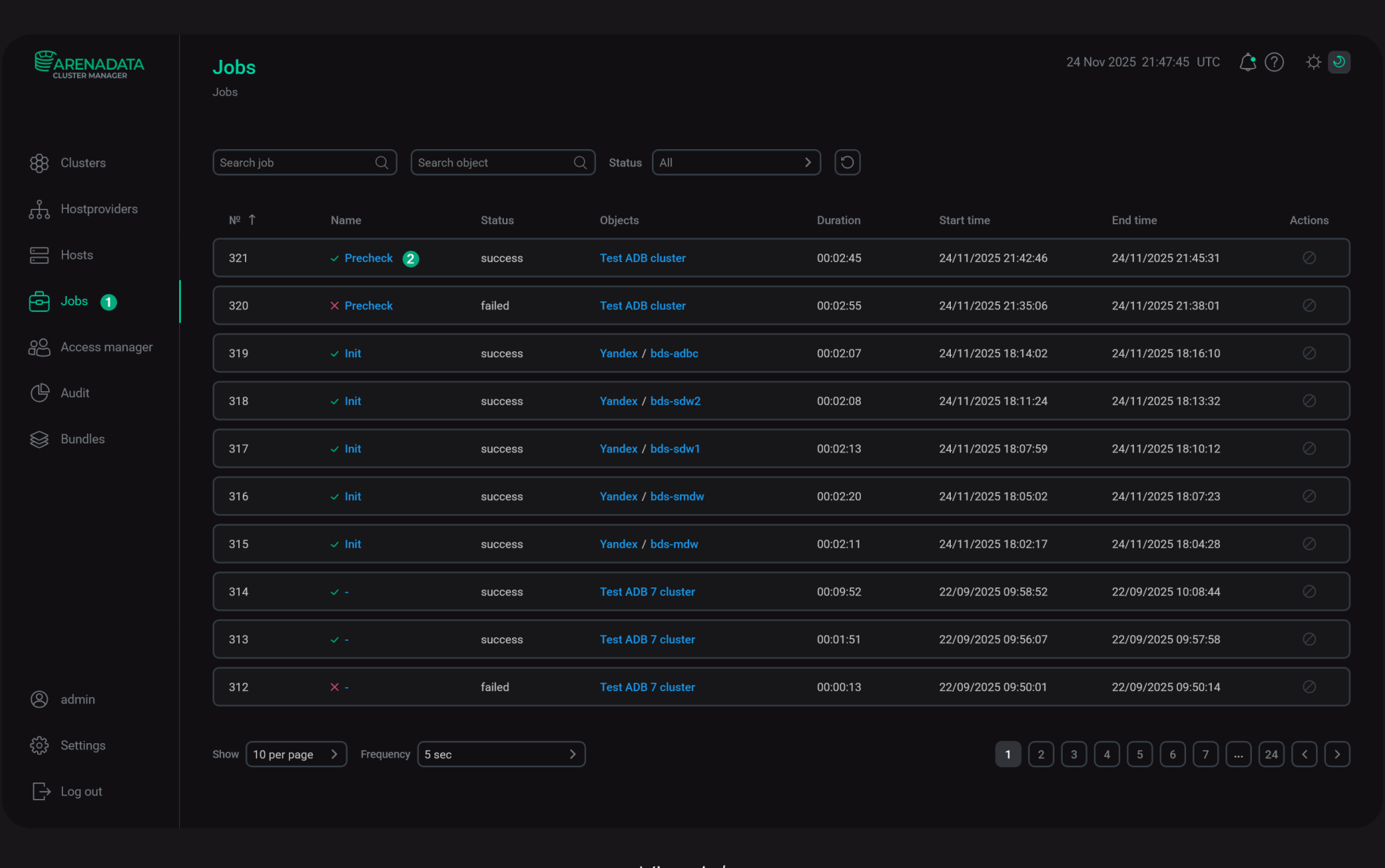
1. Apply the ***Precheck*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.


ARENADATA CLUSTER MANAGER
Clusters
Clusters
24 Nov 2025 21:32:31 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
Precheck
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

Run the cluster precheck

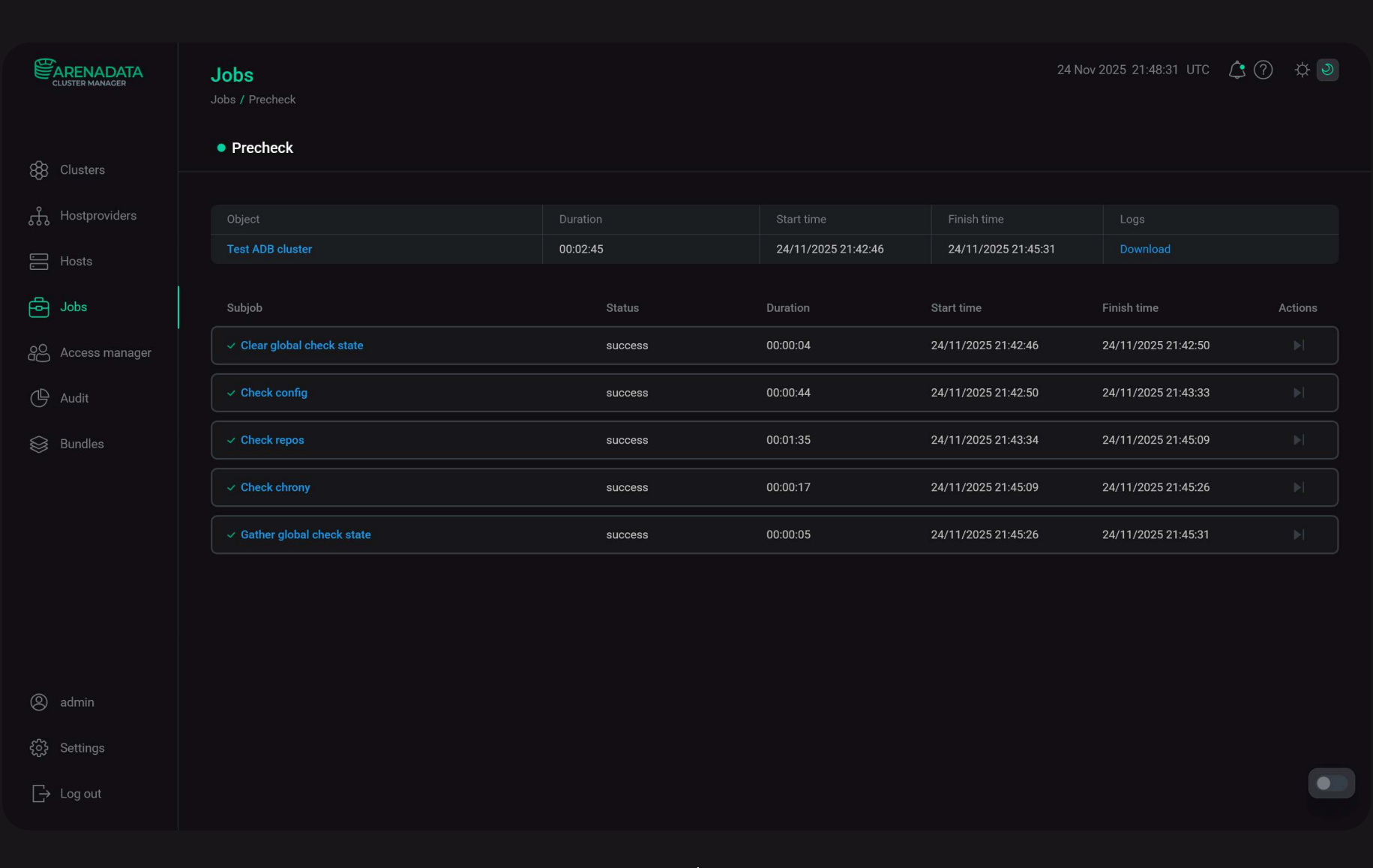
2. Confirm the requested action in the window that opens.
3. Open the ***Jobs*** page and wait until the running job is completed. After this, click the first item in the job list.


ARENADATA CLUSTER MANAGER
Jobs
Jobs
24 Nov 2025 21:47:45 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
1 2 3 4 5 6 7 ... 24

№	Name	Status	Objects	Duration	Start time	End time	Actions
321	✓ Precheck	success	Test ADB cluster	00:02:45	24/11/2025 21:42:46	24/11/2025 21:45:31	
320	✕ Precheck	failed	Test ADB cluster	00:02:55	24/11/2025 21:35:06	24/11/2025 21:38:01	
319	✓ Init	success	Yandex / bds-adbc	00:02:07	24/11/2025 18:14:02	24/11/2025 18:16:10	
318	✓ Init	success	Yandex / bds-sdw2	00:02:08	24/11/2025 18:11:24	24/11/2025 18:13:32	
317	✓ Init	success	Yandex / bds-sdw1	00:02:13	24/11/2025 18:07:59	24/11/2025 18:10:12	
316	✓ Init	success	Yandex / bds-smdw	00:02:20	24/11/2025 18:05:02	24/11/2025 18:07:23	
315	✓ Init	success	Yandex / bds-mdw	00:02:11	24/11/2025 18:02:17	24/11/2025 18:04:28	
314	✓ -	success	Test ADB 7 cluster	00:09:52	22/09/2025 09:58:52	22/09/2025 10:08:44	
313	✓ -	success	Test ADB 7 cluster	00:01:51	22/09/2025 09:56:07	22/09/2025 09:57:58	
312	✕ -	failed	Test ADB 7 cluster	00:00:13	22/09/2025 09:50:01	22/09/2025 09:50:14	

View jobs

4. View logs in the window that opens. To download logs, click ***Download***.


ARENADATA CLUSTER MANAGER
Jobs
Jobs / Precheck
24 Nov 2025 21:48:31 UTC
Precheck
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out

Object	Duration	Start time	Finish time	Logs
Test ADB cluster	00:02:45	24/11/2025 21:42:46	24/11/2025 21:45:31	Download

Subjob	Status	Duration	Start time	Finish time	Actions
✓ Clear global check state	success	00:00:04	24/11/2025 21:42:46	24/11/2025 21:42:50	
✓ Check config	success	00:00:44	24/11/2025 21:42:50	24/11/2025 21:43:33	
✓ Check repos	success	00:01:35	24/11/2025 21:43:34	24/11/2025 21:45:09	
✓ Check chrony	success	00:00:17	24/11/2025 21:45:09	24/11/2025 21:45:26	
✓ Gather global check state	success	00:00:05	24/11/2025 21:45:26	24/11/2025 21:45:31	

View logs

If the logs contain errors related to host settings (for example, regarding IPv6 or sysctl parameters), make sure the cluster hosts meet operating system prerequisites.

Step 2. Run 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.

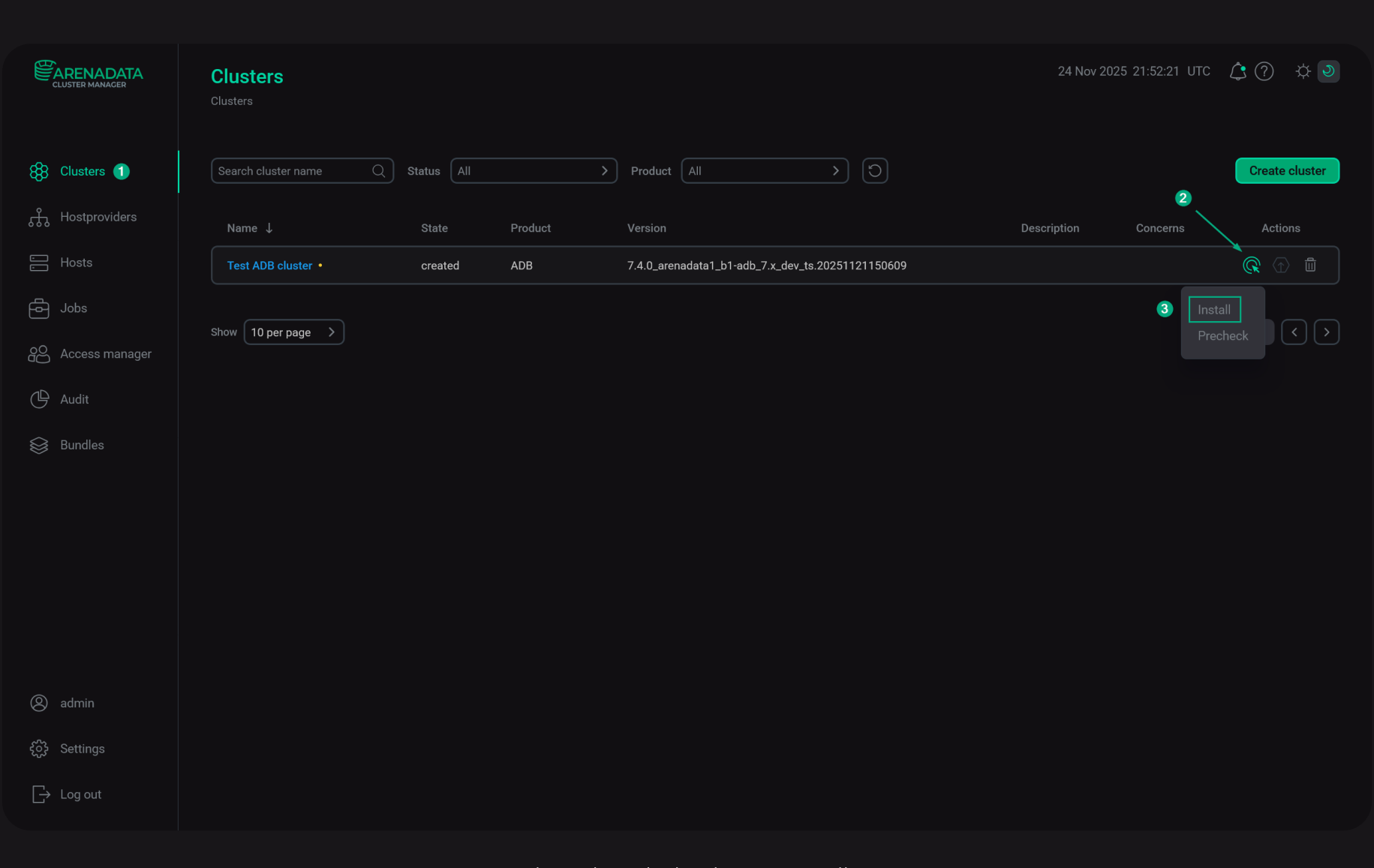
In both cases each service starts automatically after installation.

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

Whole cluster

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

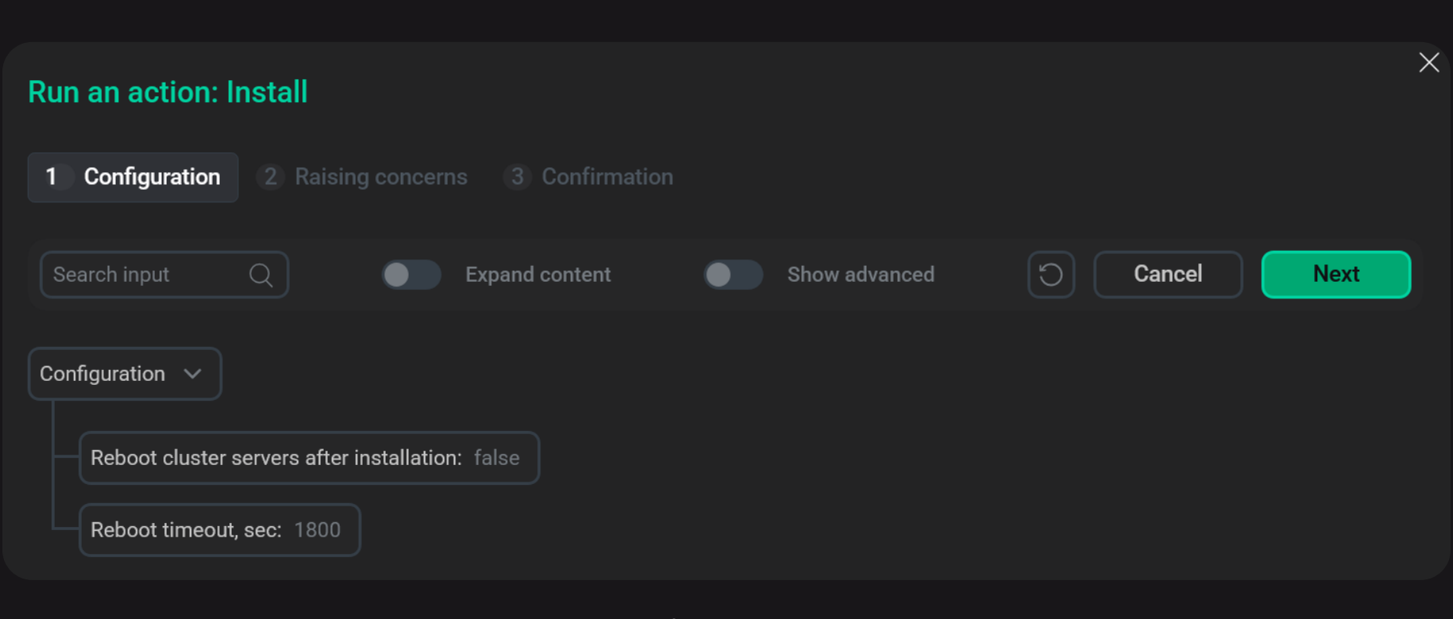
1. Apply the ***Install*** action to the cluster selected on the ***Clusters*** page by clicking the icon in the ***Actions*** column.


ARENADATA CLUSTER MANAGER
Clusters
Clusters
24 Nov 2025 21:52:21 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
Precheck
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	created	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

Switch to the whole cluster installation

2. Verify the requested action in the opened window. Notice that setting the ***Reboot cluster servers after installation*** flag leads to the hosts being rebooted after the installation is completed. Otherwise, the reboot should be done manually.

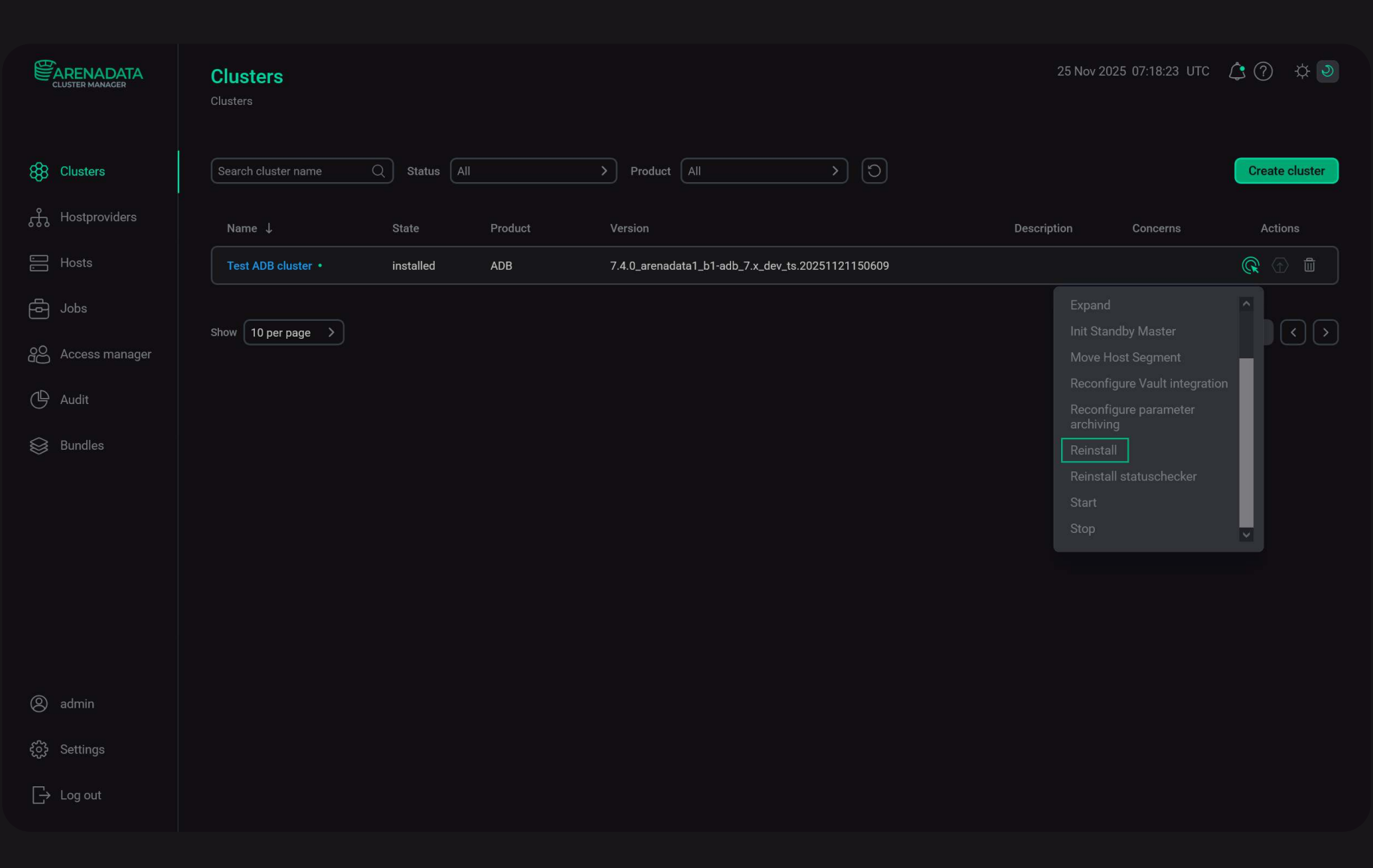

Run an action: Install
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Reboot cluster servers after installation: false

Reboot timeout, sec: 1800

Confirm the action

3. Wait until the job is completed.
4. If some service fails during installation, but the preceding services are installed successfully, you can try to fix errors and restart installation of all failed services. To do this, select the ***Reinstall*** action in the ***Actions*** column on the ***Clusters*** page.


ARENADATA CLUSTER MANAGER
Clusters
Clusters
25 Nov 2025 07:18:23 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search cluster name
Status All
Product All
Create cluster
Expand
Init Standby Master
Move Host Segment
Reconfigure Vault integration
Reconfigure parameter archiving
Reinstall
Reinstall statuschecker
Start
Stop
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	installed	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

Installation restart

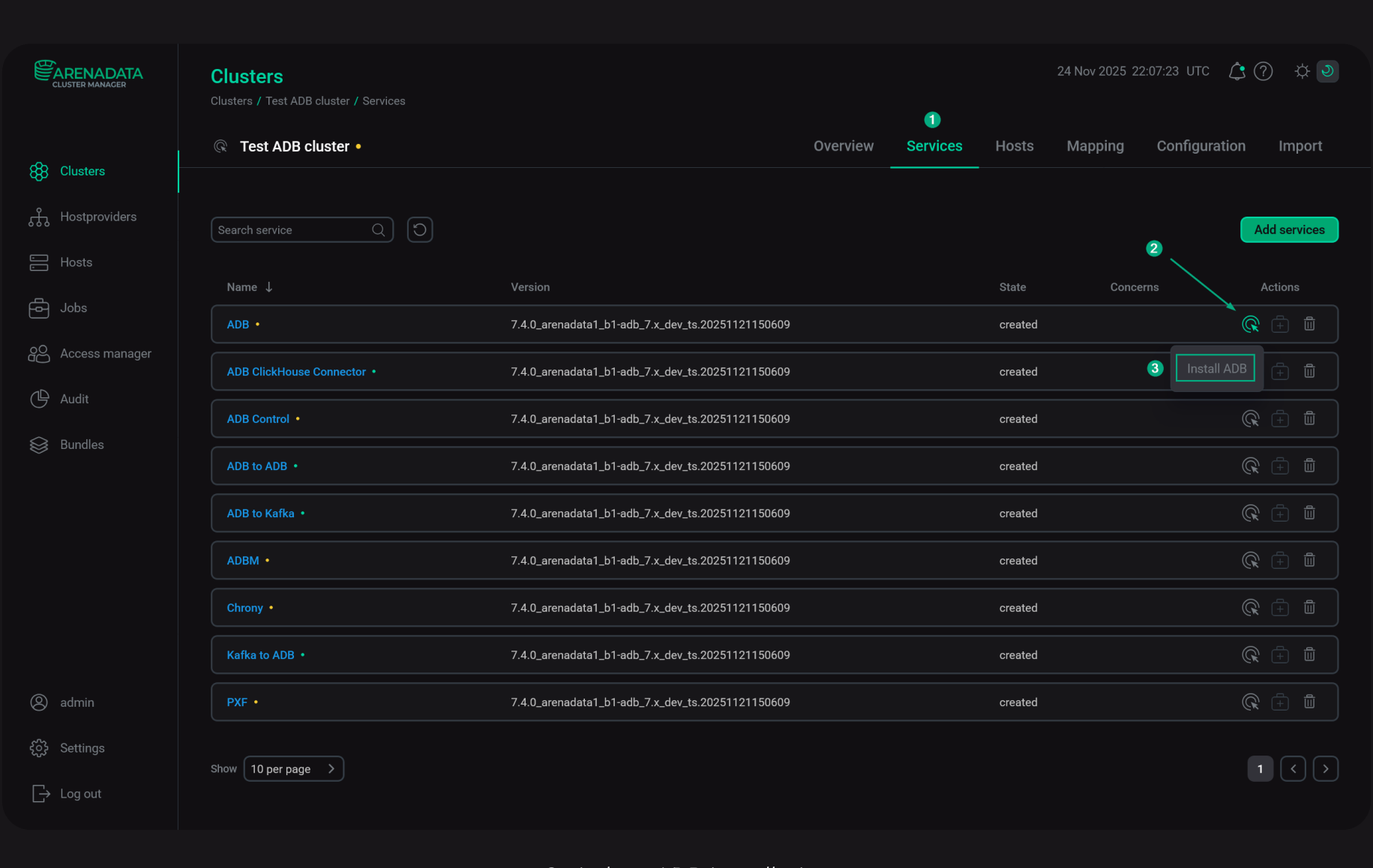
Single services

To install one or several cluster services manually, add these services to the cluster and then perform the actions described in the following subsections. The process of installing the ADB service differs from all other services.

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

ADB service

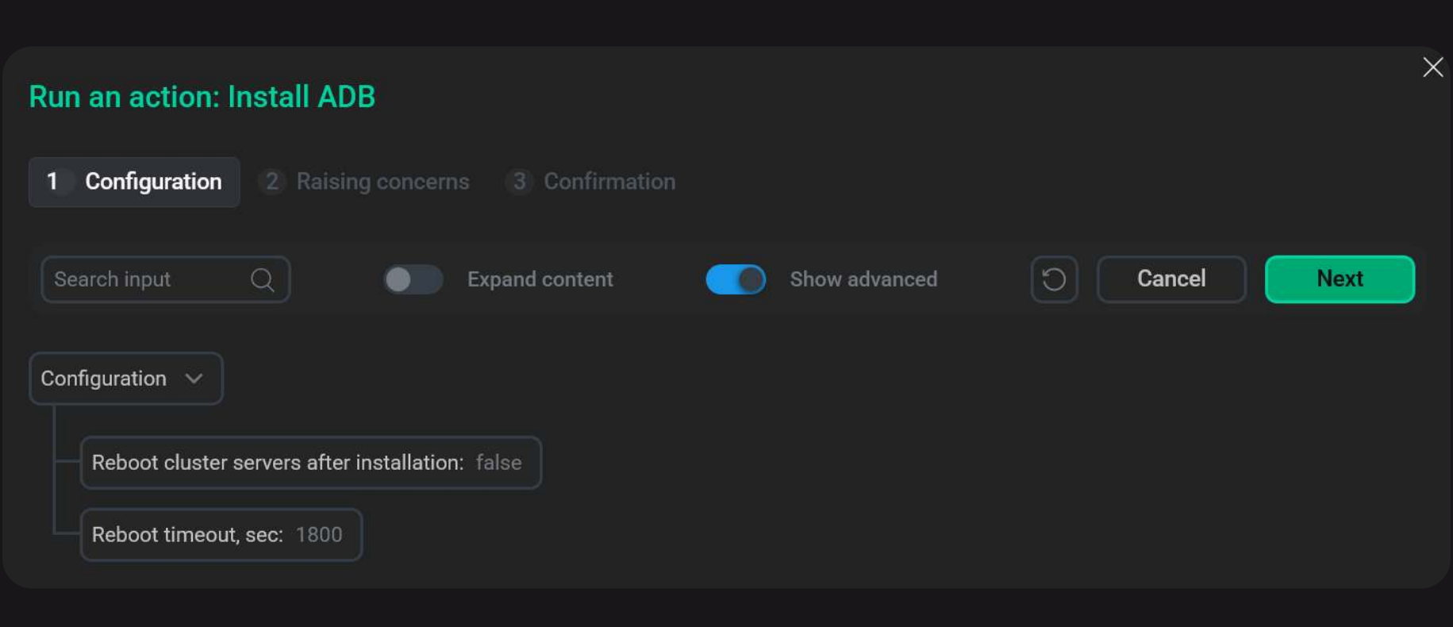
1. Open the ***Services*** tab on the cluster page. Find the row corresponding to the ADB service, click the icon in the ***Actions*** column, and select the ***Install ADB*** action.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services
24 Nov 2025 22:07:23 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add services
2
3
Install ADB
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB Control	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADBM	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		

Switch to ADB installation

2. Confirm the action in the window that opens.

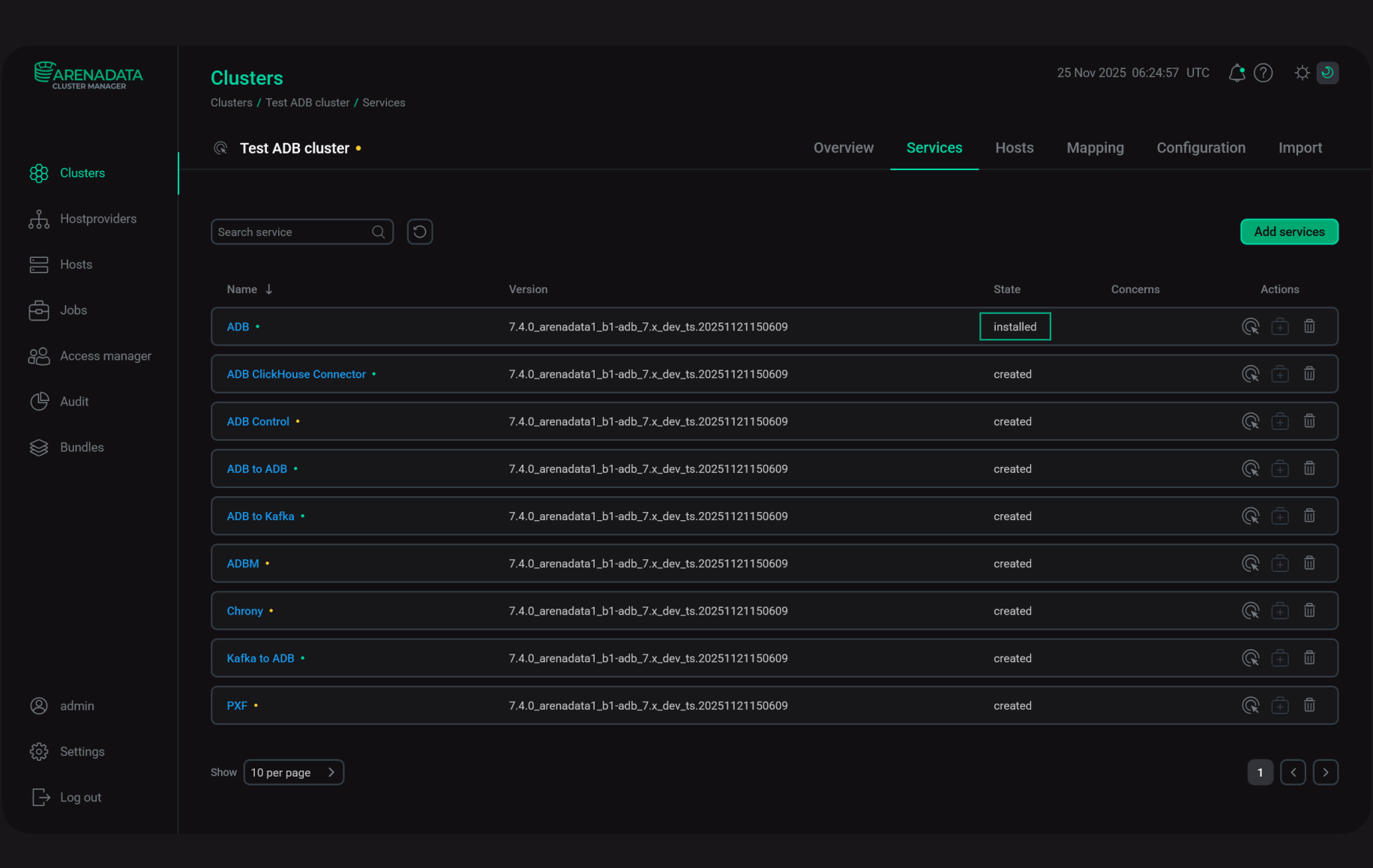

Run an action: Install ADB
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Reboot cluster servers after installation: false

Reboot timeout, sec: 1800

Confirm the action

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

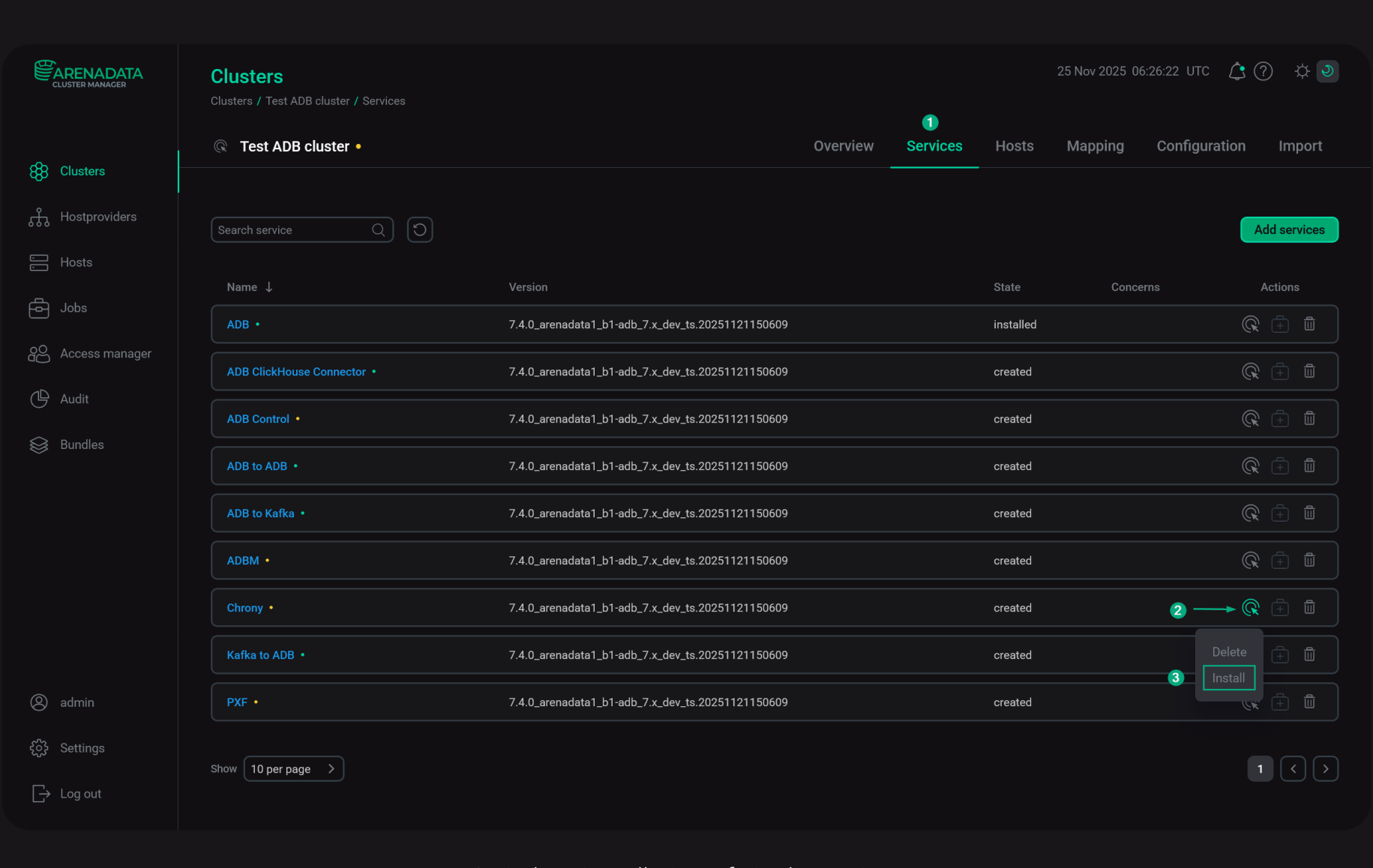

ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services
25 Nov 2025 06:24:57 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add services
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB Control	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADBM	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		

Check the ADB service state after installation

Other services

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.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services
25 Nov 2025 06:26:22 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
1
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add services
2
3
Delete
Install
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB Control	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
ADBM	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	created		

Switch to installation of single services

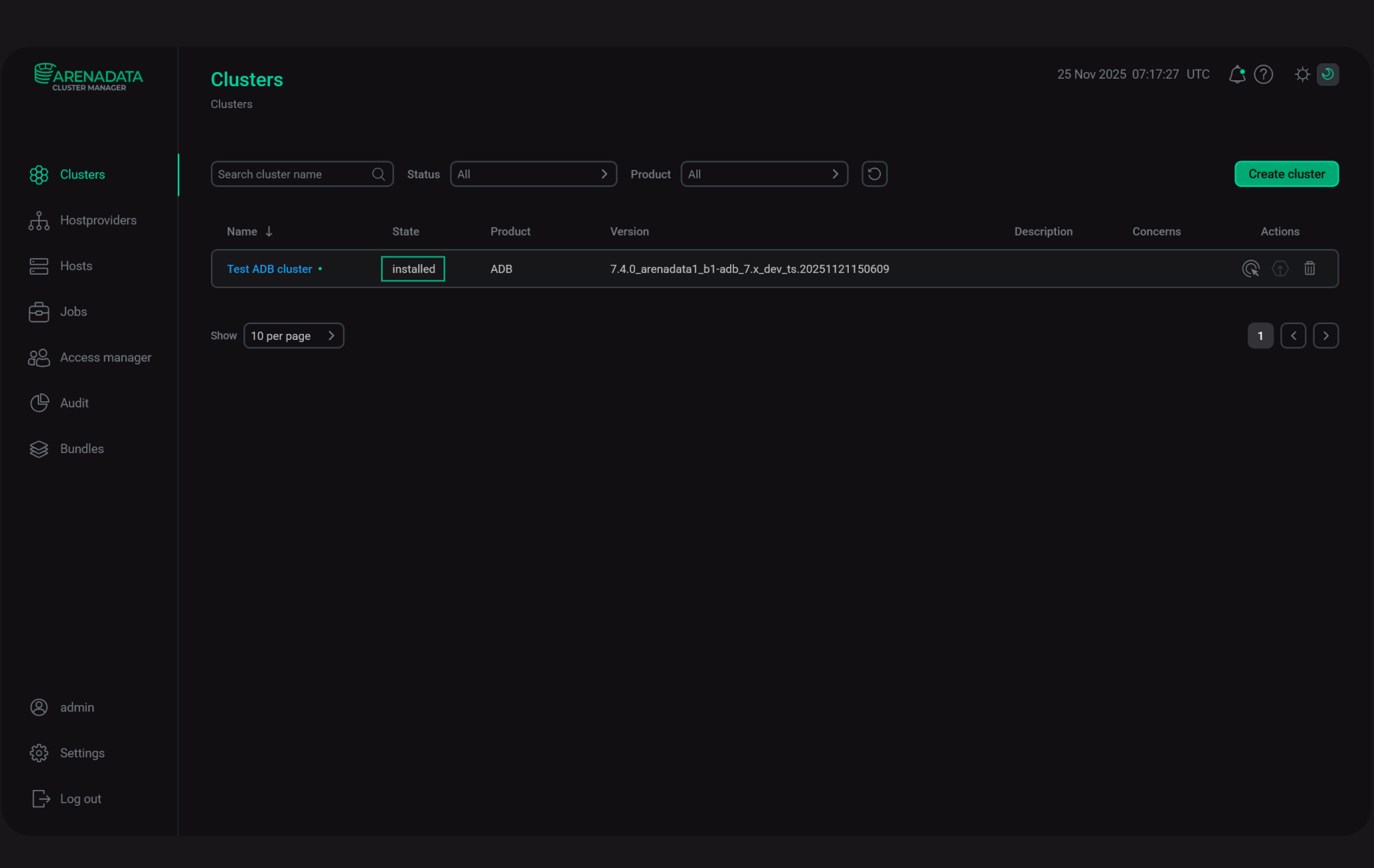
2. Confirm the action in the window that opens.
3. Wait until the job is completed. Then check that the service state has changed from `created` to a new value that is shown in the New states of ADB services after installation table.
4. Repeat the previous actions for other services that should be installed.

> IMPORTANT
>
> When you install services manually, one by one, use the following order:
>
> 1. ADB
> 2. Chrony
> 3. ADB to ADB
> 4. PXF
> 5. ADB ClickHouse Connector
> 6. ADB to Kafka
> 7. Kafka to ADB
> 8. ADB Control
> 9. ADBM

Step 3. Check the cluster state after 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`.


ARENADATA CLUSTER MANAGER
Clusters
Clusters
25 Nov 2025 07:17:27 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

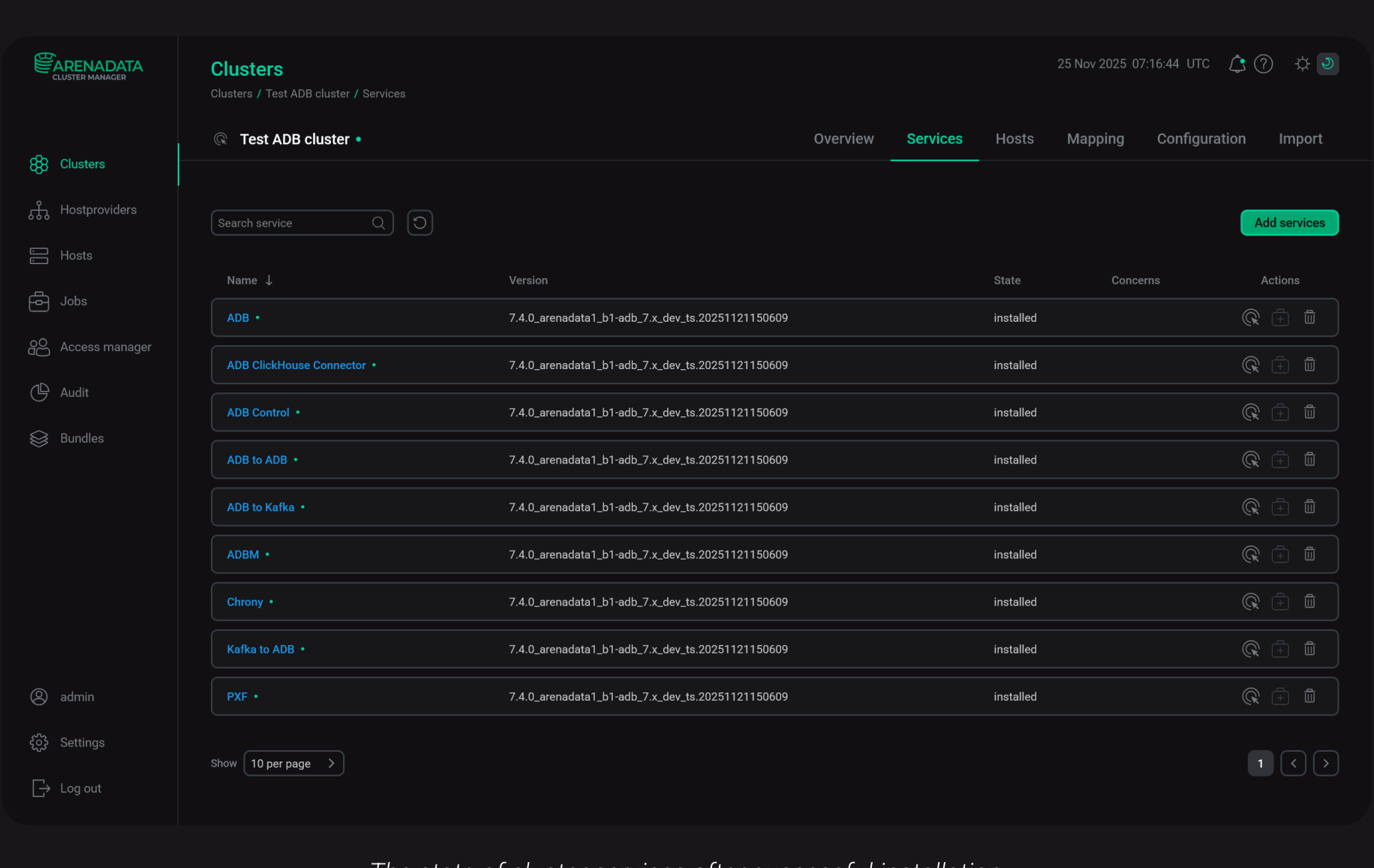
Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	installed	ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609			

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 a new value that is shown in the table below.

New states of ADB services after installation

Service	New state
ADB	installed
Chrony	installed
PXF	installed
ADB ClickHouse Connector	installed
ADB to ADB	installed
ADB to Kafka	installed
Kafka to ADB	installed
ADB Control	installed
ADBM	installed


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services
25 Nov 2025 07:16:44 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search service
Add services
Show 10 per page
1

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADB Control	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
ADBM	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251121150609	installed		

The state of cluster services after successful installation

Step 4. Check the default database

Regardless of the installation method, the default database is created during the ADB service installation. This database is used as default for client connections (for example, during the installation of extensions via ADB service actions). The database name can be specified in the ***Name of default database*** configuration parameter of the ADB service before installation. The default name is `adb`.

To ensure that the default database is successfully created, perform the following steps:

1. Connect to the ADB coordinator host (via SSH, etc.) and log in under the default user `gpadmin`.

```
$ sudo su - gpadmin
```

2. List all ADB databases.

```
psql -l
```

The command output contains the `adb` database:

```
                             List of databases
   Name    |  Owner  | Encoding |   Collate   |    Ctype    |  Access privileges
-----------+---------+----------+-------------+-------------+---------------------
 adb       | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/gpadmin        +
           |         |          |             |             | gpadmin=CTc/gpadmin
 postgres  | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0 | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/gpadmin         +
           |         |          |             |             | gpadmin=CTc/gpadmin
 template1 | gpadmin | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/gpadmin         +
           |         |          |             |             | gpadmin=CTc/gpadmin
(4 rows)
```

LDAP authentication in ADB

Daria Barysheva

Contents

To Table of Contents

Configure LDAP authentication

Synchronize ADB and LDAP

Check LDAP authentication

ADB (as well as Greenplum) supports Lightweight Directory Access Protocol (LDAP) user authentication. When LDAP authentication is used, user accounts and passwords are checked on the selected LDAP server. The steps for configuring LDAP authentication in ADB via ADCM are described below.

Configure LDAP authentication

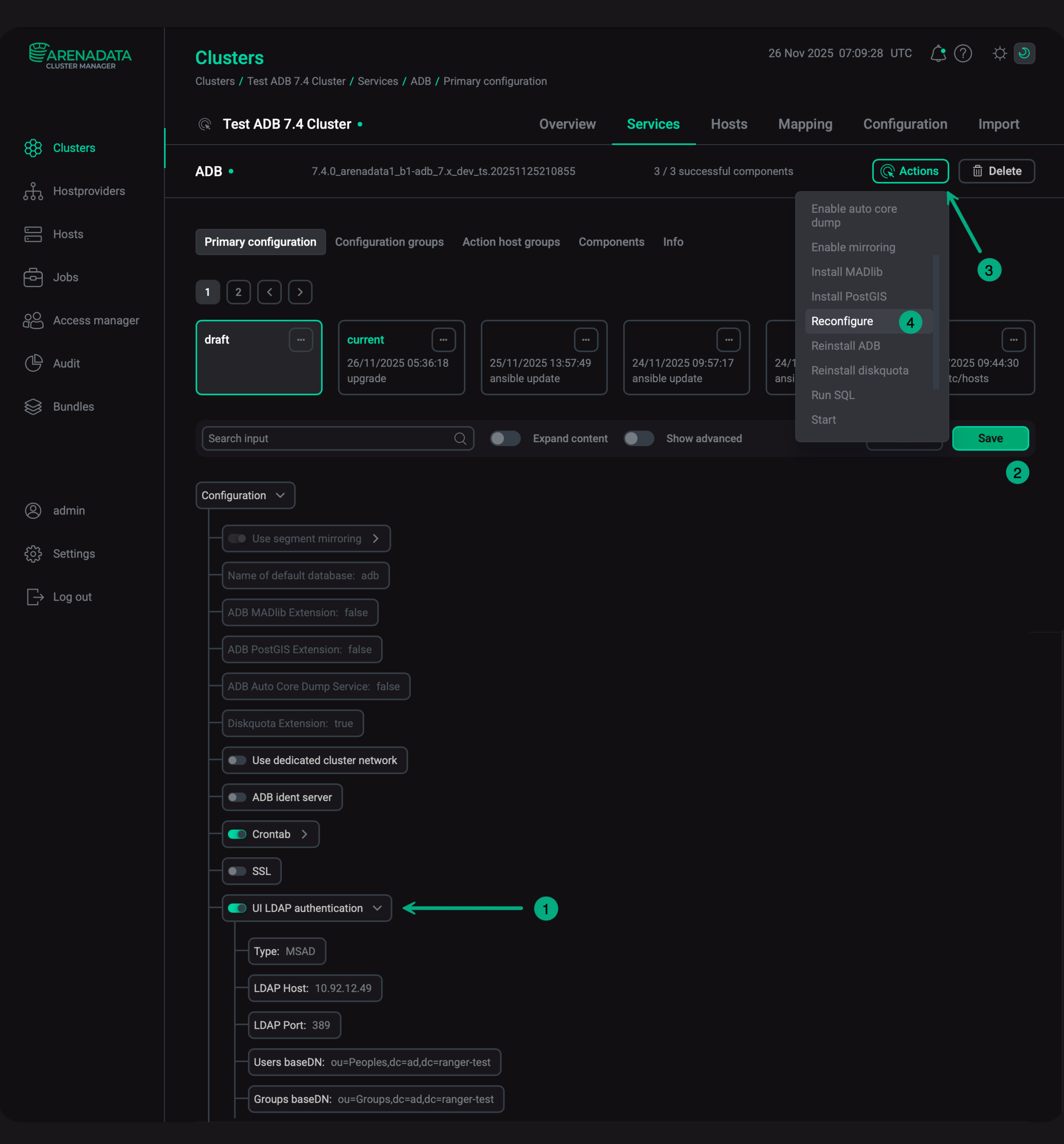
To configure LDAP authentication in ADB, follow the steps:

1. Ensure the target LDAP server is configured and available. ADB supports two LDAP implementations: Microsoft Active Directory (MS AD) and 389 Directory server as a part of FreeIPA.
2. Open the ADB service configuration page in ADCM.
3. Switch on the ***UI LDAP authentication*** toggle.
4. Fill in the following fields.

Field	Description	Default value	Example in MS AD
Type	A type of the LDAP server. Possible values: ▪ `MSAD` — Microsoft Active Directory; ▪ `FreeIPA` — 389 Directory server as a part of FreeIPA.	MSAD	MSAD
LDAP Host	A domain name or an IPv4 address of the LDAP server	—	10.92.12.49
LDAP Port	A port number on the LDAP server to connect to	—	389
Users baseDN	A limitation of the search scope of objects in the LDAP directory that is used in user search requests	—	ou=Peoples,dc=ad,dc=ranger-test
Groups baseDN	A limitation of the search scope of objects in the LDAP directory that is used in group search requests	—	ou=Groups,dc=ad,dc=ranger-test
Users filter	An LDAP filter that is used to search users. Should be specified according to the RFC 2254	(&(|(objectClass=person)(objectClass=inetOrgPerson))(cn=*))	(&(objectClass=person)(objectClass=organizationalPerson)(givenName=*)(sn=*)(sAMAccountName=*))
Groups filter	An LDAP filter that is used to search groups. Should be specified according to the RFC 2254	(&(|(objectClass=group)(objectClass=groupofnames)(objectClass=groupOfUniqueNames))(cn=*))	(cn=*)
SA Login	A user name that is used for service requests to the LDAP server	—	cn=admin,dc=ad,dc=ranger-test
Password	A user password that is used for service requests to the LDAP server	—	A password of the `admin` user
Lowercase login	Defines whether to convert user names to lowercase	false	false
Lowercase group	Defines whether to convert group names to lowercase	false	false
Sync interval	A `cron` expression that defines how often synchronization of ADB users and groups with an LDAP server should be performed. The pg-ldap-sync utility, which comes with the ADB bundle, is used for synchronization	0 * * * *	0 * * * *

> **NOTE**
>
> Various LDAP implementations use different names for object types and identifiers. It is recommended to refine the settings format for a specific LDAP implementation from the LDAP server administrator.

5. Click ***Save***. Then apply the ***Reconfigure*** action to the ADB service.



ARENADATA CLUSTER MANAGER
Clusters
26 Nov 2025 07:09:28 UTC
Clusters / Test ADB 7.4 Cluster / Services / ADB / Primary configuration
Test ADB 7.4 Cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
ADB
7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855
3 / 3 successful components
Actions
Delete
Enable auto core dump
Enable mirroring
Install MADlib
Install PostGIS
Reconfigure
Reinstall ADB
Reinstall diskquota
Run SQL
Start
Primary configuration
Configuration groups
Action host groups
Components
Info
1
2
draft
current
26/11/2025 05:36:18
upgrade
25/11/2025 13:57:49
ansible update
24/11/2025 09:57:17
ansible update
2025 09:44:30
tc/hosts
Search input
Expand content
Show advanced
Save
Configuration
Use segment mirroring
Use dedicated cluster network
ADB ident server
Crontab
SSL
UI LDAP authentication
1
2
3
4

Name of default database: adb
ADB MADlib Extension: false
ADB PostGIS Extension: false
ADB Auto Core Dump Service: false
Diskquota Extension: true
Type: MSAD
LDAP Host: 10.92.12.49
LDAP Port: 389
Users baseDN: ou=Peoples,dc=ad,dc=ranger-test
Groups baseDN: ou=Groups,dc=ad,dc=ranger-test

Configure LDAP authentication for ADB via ADCM

Synchronize ADB and LDAP

If LDAP authentication is configured successfully, the following operations are performed in ADB automatically:

- **Update of the pg_hba.conf file**. The following entry about LDAP authentication is added to the *pg_hba.conf* file, which is located in the data directory of the coordinator host (*/data1/master/gpseg-1/* in the following example):

```
# BEGIN LDAP
host all +ldap_users 0.0.0.0/0 ldap ldapserver=<LDAP Host> ldapport=<LDAP Port> ldapbasedn="<Users baseDN>" ldapbinddn="<SA
Login>" ldapbindpasswd="<Password>" ldapsearchattribute="<attribute>"
# END LDAP
```

  where:

  - `ldap_users` — a name of the ADB role, to which all LDAP users will be added.
  - `<LDAP Host>`, `<LDAP Port>`, `<Users baseDN>`, `<SA Login>`, `<Password>` — values of the eponymous parameters of the ADB service.
  - `<attribute>` — the LDAP attribute whose value will be used as a name of each role in ADB when synchronizing users. Possible values:
    - `sAMAccountName` — for MS AD.
    - `uid` — for FreeIPA.

- **Synchronization of ADB users and groups with LDAP**. Information about LDAP users and groups, which meet the selection criteria specified in the configuration step, is added to the ADB database. Synchronization is performed via the pg-ldap-sync utility. To ensure successful synchronization, you can run the following query against the `pg_roles` table:

```
SELECT rolname FROM pg_roles;
```

  As the SQL command output shows, the following roles are automatically created in ADB after successful LDAP synchronization:

  - `ldap_users` — a role into which all LDAP users are added (by running the `CREATE ROLE … LOGIN IN ROLE ldap_users …` command automatically).
  - `ldap_groups` — a role into which all LDAP groups are added (by running the `CREATE ROLE … NOLOGIN IN ROLE ldap_groups …` command automatically).
  - `ffedorov`, `ppetrov`, `ssemenov`, `zeptest` — all users received from the LDAP server and added to the `ldap_users` role.
  - `ADCM`, `ADH`, `Adccadmins`, `Analysts`, `Managers` — all groups received from the LDAP server and added to the `ldap_groups` role. The membership of users in these groups is set automatically by running `GRANT`.

```
   rolname
-------------
 gpadmin
 gpmon
 ldap_users
 ldap_groups
 ADCM
 ADH
 Adccadmins
 Analysts
 Managers
 ffedorov
 ppetrov
 ssemenov
 zeptest
(13 rows)
```

Synchronization details

- After initial configuration, ADB users are automatically synchronized with an LDAP server according to the schedule specified in the Sync interval parameter. There is no need to run the `CREATE ROLE` and `GRANT` commands manually.
- To synchronize ADB users with an LDAP server on demand (without waiting for the next auto launch), apply the ***Reconfigure*** action to the ADB service.
- After LDAP authentication is configured, only users registered on the LDAP server can connect to the ADB database. If you delete users on the LDAP side, they cannot further log in to ADB.
- If a user exists in ADB but not on the LDAP server side, synchronization does not affect this user. However, if a user with the same name exists both in ADB and on the LDAP server — the next time synchronization runs, the user data in ADB will be overwritten with the information received from LDAP.
- The `pg-ldap-sync` utility automates only the following DCL commands: `CREATE ROLE <ROLE>`, `DROP ROLE <ROLE>`, `GRANT <ROLE> TO <ROLE>`, and `REVOKE <ROLE> FROM <ROLE>`. The commands that define access privileges (e.g. `GRANT SELECT`, `GRANT EXECUTE`, `GRANT CONNECT`, `GRANT PRIVILEGES ON <OBJECT> TO <ROLE>`) should be run manually in ADB if necessary.

Check LDAP authentication

To check that LDAP authentication is properly configured in ADB, you can run `psql` to connect to ADB using the name of any LDAP user as the `-U` argument (`ffedorov` in the following example).

```
$ psql adb -h bds-mdw -U "ffedorov"
```

If LDAP authentication is configured successfully, you will be prompted to enter the user password. After entering the correct password (assigned on the LDAP server side), the user can access the database.

```
Password for user ffedorov:
psql (12.22)
Type "help" for help.

adb=>
```

LDAP authentication errors

- If a user with the specified name cannot be found among members of the `ldap_users` role in ADB, the following error occurs when you try to log in to ADB under that user:

```
psql: FATAL:  no pg_hba.conf entry for host "10.92.42.236", user "bds", database "adb", SSL off
```

- If a user with the specified name is a member of the `ldap_users` role in ADB, but the user name or password cannot be found in LDAP, the following error occurs when you try to log in to ADB under that user:

```
psql: FATAL:  LDAP authentication failed for user "bds"
```

In both cases, you should wait for automatic synchronization with an LDAP server or run it manually using the ***Reconfigure*** action of the ADB service.

Use PAM for LDAP authentication in ADB

Daria Barysheva

Contents

To Table of Contents

- Install and configure OpenLDAP server
  - Set up FQDN
  - Install OpenLDAP packages
  - Configure OpenLDAP server
  - Create base groups
  - Create a user
  - Disable anonymous binding
- Configure ADB coordinator
- Check authentication

Pluggable authentication modules (PAM) is a mechanism that provides the ability to use multiple low-level authentication types in applications by means of the single high-level API. User authentication is requested from the PAM system (by calling its API) rather than directly from security systems, which provides flexibility and allows you to use different authentication methods without making changes to the code of the applications themselves.

The article describes how to use PAM for configuring LDAP authentication in ADB.

Install and configure OpenLDAP server

The example below demonstrates how to install and configure an OpenLDAP server on the `bds-pam.ru-central1.internal` host with IP address `10.92.38.114` and operating system Ubuntu 22.04. All actions are performed with `root` privileges. In your environment, configuration steps can be different.

If you already have the LDAP server configured, go to the Configure ADB coordinator step.

Set up FQDN

1. Run the following command to set up the fully qualified domain name (FQDN) of your future LDAP server:

```
$ sudo hostnamectl set-hostname bds-pam.ru-central1.internal
```

2. Open the */etc/hosts* configuration file for editing:

```
$ sudo vi /etc/hosts
```

Add one line to the file in the `<IP> <FQDN> <hostname>` format:

```
10.92.38.114 bds-pam.ru-central1.internal bds-pam
```

Save and close the file.

3. Run the command below to check the FQDN:

```
$ sudo hostname -f
```

Result:

```
bds-pam.ru-central1.internal
```

4. Ping the hostname and ensure that you get response from the server IP address instead of `localhost`:

```
$ ping bds-pam
```

Result:

```
PING bds-pam.ru-central1.internal (10.92.38.114) 56(84) bytes of data.
64 bytes from bds-pam.ru-central1.internal (10.92.38.114): icmp_seq=1 ttl=64 time=0.015 ms
64 bytes from bds-pam.ru-central1.internal (10.92.38.114): icmp_seq=2 ttl=64 time=0.038 ms
64 bytes from bds-pam.ru-central1.internal (10.92.38.114): icmp_seq=3 ttl=64 time=0.039 ms
^C
--- bds-pam.ru-central1.internal ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2051ms
rtt min/avg/max/mdev = 0.015/0.030/0.039/0.011 ms
```

Install OpenLDAP packages

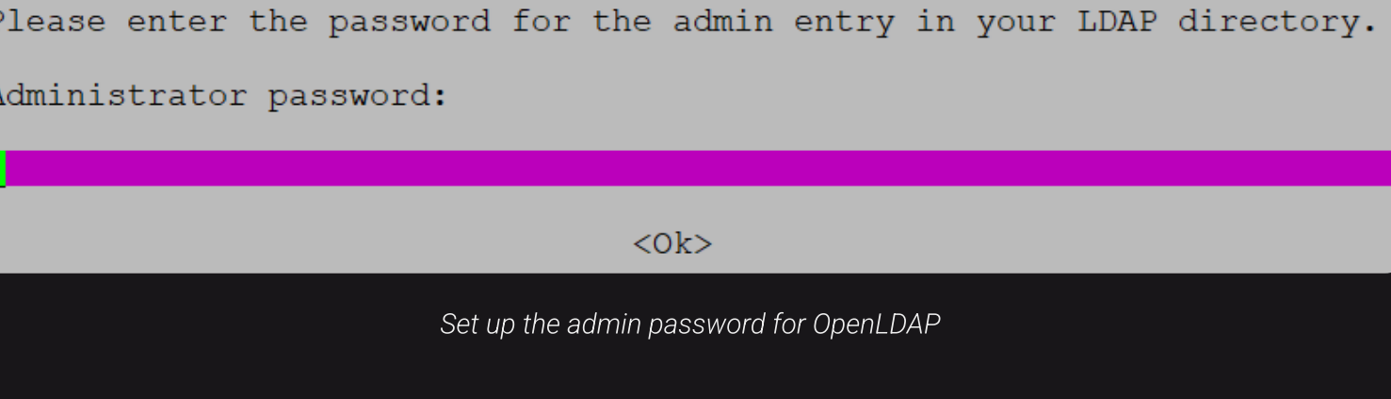
1. Update all packages that are currently installed in your system:

```
$ sudo apt update
```

2. Install the OpenLDAP packages using the following command:

```
$ sudo apt install slapd ldap-utils
```

During the OpenLDAP packages installation, you will be asked to set up the `admin` password for OpenLDAP. Enter a password, then repeat your password in the next window.



```
Please enter the password for the admin entry in your LDAP directory.

Administrator password:

                              <Ok>
```

Set up the admin password for OpenLDAP

In the last dialog box (with selection of services for restart), you can confirm the default settings.

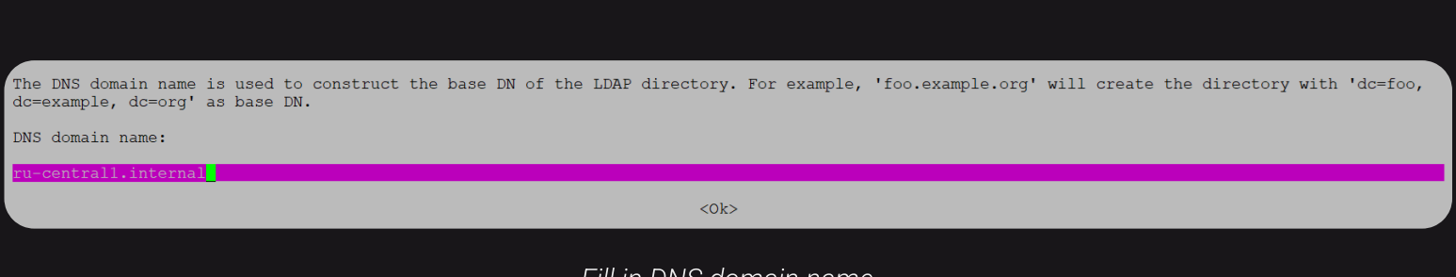
Configure OpenLDAP server

1. To start configuring OpenLDAP, run the following command:

```
$ sudo dpkg-reconfigure slapd
```

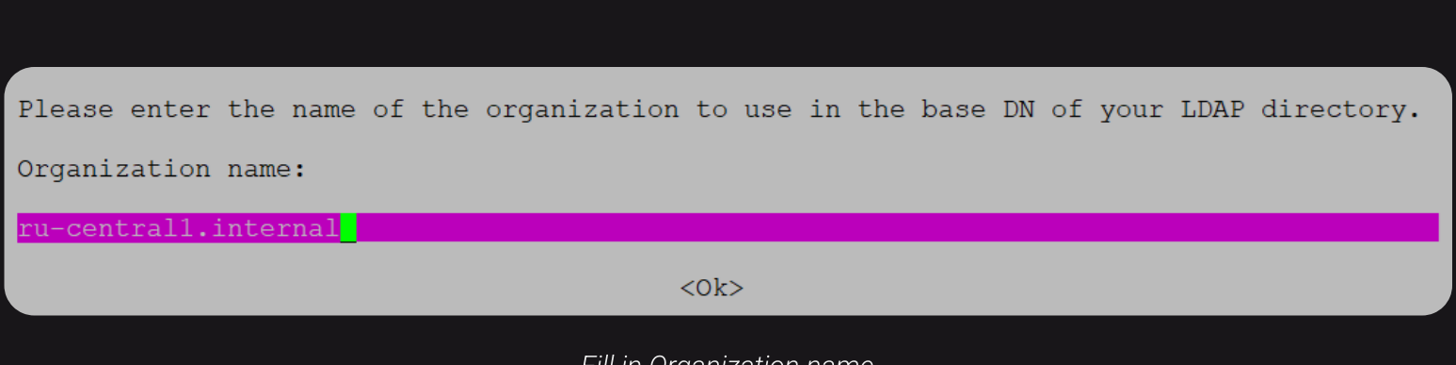
During the configuration process, you will see several dialog boxes with parameters. The following parameters are required:

- ***DNS domain name*** — domain name that will be used to construct the base distinguished name (DN) of your LDAP directory.
- ***Organization name*** — organization name that will be used in the base DN of your LDAP directory. By default, it is the same as the domain name. You can use another name if necessary.
- ***Administrator password*** — password of the `admin` entry in the LDAP directory. Enter the password selected in step 2 of the Install OpenLDAP packages section. Then, confirm it in the next window.



```
The DNS domain name is used to construct the base DN of the LDAP directory. For example, 'foo.example.org' will create the directory with 'dc=foo,
dc=example, dc=org' as base DN.

DNS domain name:

ru-central1.internal

                              <Ok>
```

Fill in DNS domain name



```
Please enter the name of the organization to use in the base DN of your LDAP directory.

Organization name:

ru-central1.internal

                              <Ok>
```

Fill in Organization name

In other dialog boxes, you can confirm the default settings.

The successful command result is shown below:

```
Backing up /etc/ldap/slapd.d in /var/backups/slapd-2.5.19+dfsg-0ubuntu0.22.04.1... done.
  Moving old database directory to /var/backups:
  - directory unknown... done.
  Creating initial configuration... done.
  Creating LDAP directory... done.
```

2. Open the */etc/ldap/ldap.conf* file for editing:

```
$ sudo vi /etc/ldap/ldap.conf
```

In the file, uncomment the `BASE` and `URI` fields. As their values, enter your base DN and LDAP server URI accordingly:

```
BASE    dc=ru-central1,dc=internal
URI     ldap://bds-pam.ru-central1.internal
```

Save and close the file.

3. Restart the `slapd` service to apply changes:

```
$ sudo systemctl restart slapd
```

4. Check the `slapd` service status:

```
$ sudo systemctl status slapd
```

The command output should be as follows:

```
slapd.service - LSB: OpenLDAP standalone server (Lightweight Directory Access Protocol)
     Loaded: loaded (/etc/init.d/slapd; generated)
    Drop-In: /usr/lib/systemd/system/slapd.service.d
             └─slapd-remain-after-exit.conf
     Active: active (running) since Mon 2025-07-28 15:28:15 UTC; 7s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 5958 ExecStart=/etc/init.d/slapd start (code=exited, status=0/SUCCESS)
      Tasks: 3 (limit: 19050)
     Memory: 3.3M
        CPU: 23ms
     CGroup: /system.slice/slapd.service
             └─5965 /usr/sbin/slapd -h "ldap:/// ldapi:///" -g openldap -u openldap -F /etc/ldap/slapd.d
```

5. Check the OpenLDAP server configuration:

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:///
```

The command output should contain the correct base DN:

```
dn: dc=ru-central1,dc=internal
objectClass: top
objectClass: dcObject
objectClass: organization
o: ru-central1.internal
dc: ru-central1
```

Create base groups

1. Create a new LDIF file using the command below:

```
$ sudo vi base-groups.ldif
```

In the file, describe configuration of two base groups: `People` (for storing users) and `Groups` (for storing groups):

```
dn: ou=People,dc=ru-central1,dc=internal
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=ru-central1,dc=internal
objectClass: organizationalUnit
ou: Groups
```

Save and close the file.

2. Run the `ldapadd` command to add groups from the previously created file to the LDAP directory:

```
$ sudo ldapadd -x -D cn=admin,dc=ru-central1,dc=internal -W -f base-groups.ldif
```

Enter the `admin` password to confirm the operation.

The successful result:

```
adding new entry "ou=People,dc=ru-central1,dc=internal"
adding new entry "ou=Groups,dc=ru-central1,dc=internal"
```

3. Run the `ldapsearch` command to check that new groups have been created:

```
$ sudo ldapsearch -Q -LLL -Y EXTERNAL -H ldapi:///
```

The command returns new groups:

```
dn: dc=ru-central1,dc=internal
objectClass: top
objectClass: dcObject
objectClass: organization
o: ru-central1.internal
dc: ru-central1

dn: ou=People,dc=ru-central1,dc=internal
objectClass: organizationalUnit
ou: People

dn: ou=Groups,dc=ru-central1,dc=internal
objectClass: organizationalUnit
ou: Groups
```

Create a user

1. Run the `slappasswd` command to generate an encrypted password for the new LDAP user:

```
$ sudo slappasswd
```

Enter the new user password and confirm it. Remember the password as it will be required during authentication check (see step 2 of the Check authentication section).

The encrypted password is displayed:

```
{SSHA}XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

2. Create a new LDIF file using the command below:

```
$ sudo vi user.ldif
```

In the file, describe the `alice` user configuration as follows. Note the following:

- In the `userPassword` field, you should enter the encrypted password generated in the previous step.
- The user is added to the `People` base group. If necessary, you can use another group.

```
dn: uid=alice,ou=People,dc=ru-central1,dc=internal
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alice
sn: Test
givenName: Alice
cn: Alice Test
displayName: Alice Test
uidNumber: 10000
gidNumber: 5000
userPassword: {SSHA}XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
gecos: Alice Test
loginShell: /bin/bash
homeDirectory: /home/alice
```

Save and close the file.

3. Run the `ldapadd` command to add a new user to the LDAP directory:

```
$ sudo ldapadd -x -D cn=admin,dc=ru-central1,dc=internal -W -f user.ldif
```

Enter the `admin` password to confirm the operation.

The successful command result:

```
adding new entry "uid=alice,ou=People,dc=ru-central1,dc=internal"
```

4. Check that the new user has been created:

```
$ sudo ldapsearch -x -LLL -b dc=ru-central1,dc=internal '(uid=alice)' cn uidNumber gidNumber
```

Result:

```
dn: uid=alice,ou=People,dc=ru-central1,dc=internal
cn: Alice Test
uidNumber: 10000
gidNumber: 5000
```

Disable anonymous binding

IMPORTANT

If you disable anonymous binding to LDAP as follows, it will be necessary to fill in the `binddn` and `bindpw` options in the PAM configuration file */etc/nslcd.conf* (see step 3 in the Configure ADB coordinator section).

If you need to disable anonymous binding to an LDAP server, perform the following:

1. Create a new LDIF file:

```
$ sudo vi ldap_disable_bind_anon.ldif
```

Add the following lines to the file:

```
dn: cn=config
changetype: modify
add: olcDisallows
olcDisallows: bind_anon

dn: cn=config
changetype: modify
add: olcRequires
olcRequires: authc

dn: olcDatabase={-1}frontend,cn=config
changetype: modify
add: olcRequires
olcRequires: authc
```

Save and close the file.

2. Run the `ldapadd` command to apply changes:

```
$ sudo ldapadd -Y EXTERNAL -H ldapi:/// -f ldap_disable_bind_anon.ldif
```

Result:

```
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
modifying entry "cn=config"
modifying entry "cn=config"
modifying entry "olcDatabase={-1}frontend,cn=config"
```

Configure ADB coordinator

Run the following commands on the ADB coordinator host:

1. Update all packages that are currently installed in your system:

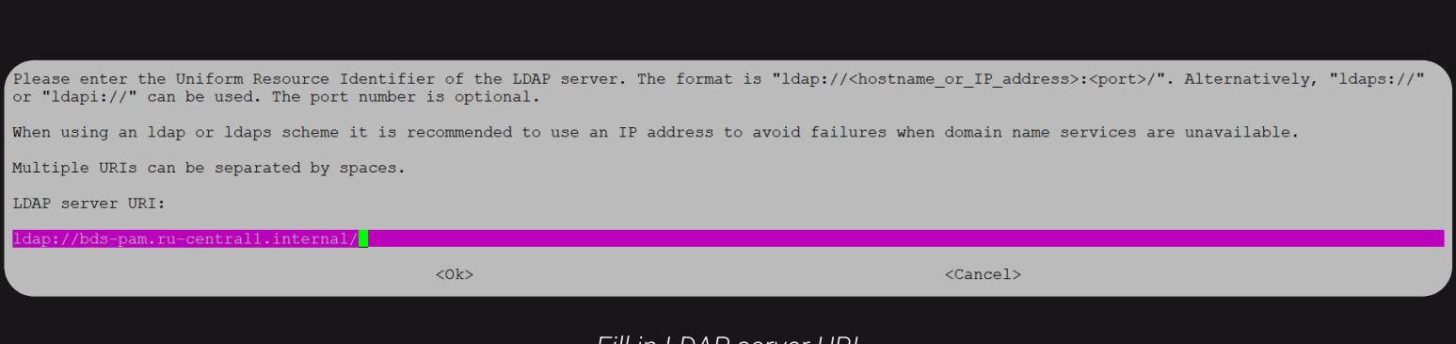
```
$ sudo apt update
```

2. Install the libpam-ldapd package:

```
$ sudo apt install libpam-ldapd
```

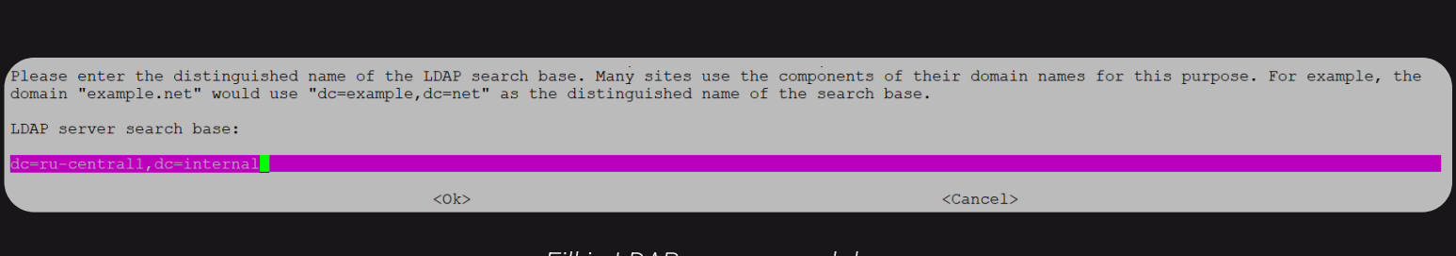
During the configuration process, you will see several dialog boxes with parameters. The following parameters are required:

- ***LDAP server URI*** — URI of the LDAP server, for example: `ldap://bds-pam.ru-central1.internal/`.
- ***LDAP server search base*** — base DN for LDAP search, for example: `dc=ru-central1,dc=internal`.



```
Please enter the Uniform Resource Identifier of the LDAP server. The format is "ldap://<hostname_or_IP_address>:<port>/". Alternatively, "ldaps://"
or "ldapi://" can be used. The port number is optional.

When using an ldap or ldaps scheme it is recommended to use an IP address to avoid failures when domain name services are unavailable.

Multiple URIs can be separated by spaces.

LDAP server URI:

ldap://bds-pam.ru-central1.internal/

                 <Ok>                                  <Cancel>
```

Fill in LDAP server URI



```
Please enter the distinguished name of the LDAP search base. Many sites use the components of their domain names for this purpose. For example, the
domain "example.net" would use "dc=example,dc=net" as the distinguished name of the search base.

LDAP server search base:

dc=ru-central1,dc=internal

                 <Ok>                                  <Cancel>
```

Fill in LDAP server search base

In other dialog boxes, you can confirm the default settings.

3. Open the PAM configuration file */etc/nslcd.conf* for editing:

```
$ sudo vi /etc/nslcd.conf
```

The file contents are shown below. For information on available options, see GitHub.

```
# nslcd configuration file. See nslcd.conf(5)
# for details.

# The user and group nslcd should run as.
uid nslcd
gid nslcd

# The location at which the LDAP server(s) should be reachable.
uri ldap://bds-pam.ru-central1.internal/

# The search base that will be used for all queries.
base dc=ru-central1,dc=internal

# The LDAP protocol version to use.
#ldap_version 3

# The DN to bind with for normal lookups.
binddn cn=admin,dc=ru-central1,dc=internal
bindpw qwerty

# The DN used for password modifications by root.
#rootpwmoddn cn=admin,dc=example,dc=com
```

Note that the `uri` and `base` option values are set automatically based on the data entered during the `libpam-ldapd` installation. As for the `binddn` and `bindpw` options, you should fill in them manually in the file (if necessary) — by passing the DN and password of your LDAP administrator accordingly.

NOTE

If your environment allows anonymous binding to the LDAP server for searching (and the abovementioned step Disable anonymous binding was not performed), the `binddn` and `bindpw` parameters are optional. If anonymous binding is forbidden, these parameters are required; otherwise, during authentication check, you get the following error: `Authentication service cannot retrieve authentication info`.

4. Save changes to the */etc/nslcd.conf* file (if any), close the file, and restart the `nslcd` service:

```
$ sudo systemctl restart nslcd
```

5. Create the */etc/pam.d/postgresql* file to enter configuration options of the PAM module `postgresql`:

```
$ sudo vi /etc/pam.d/postgresql
```

Add the following lines to the file:

```
auth        required      pam_ldap.so
account     sufficient    pam_ldap.so
```

Save and close the file.

Check authentication

To check PAM authentication, run the following commands on the ADB coordinator host:

1. (Optional) Install the `pamtester` utility to precheck that all PAM settings are correct:

```
$ sudo apt install pamtester
```

2. (Optional) Run the `pamtester` utility by passing a name of the PAM module (`postgresql`) and the LDAP user name (`alice`) in the utility parameters:

```
$ pamtester postgresql alice authenticate
```

After running the utility, enter the `alice` user password (see step 1 in the Create a user section). The successful authentication result is shown below:

```
pamtester: successfully authenticated
```

3. Log in to the ADB coordinator host under the `gpadmin` user:

```
$ sudo su - gpadmin
```

4. Connect to the `adb` database using `psql`:

```
$ psql adb
```

5. Create the `alice` role:

```
CREATE ROLE alice LOGIN;
```

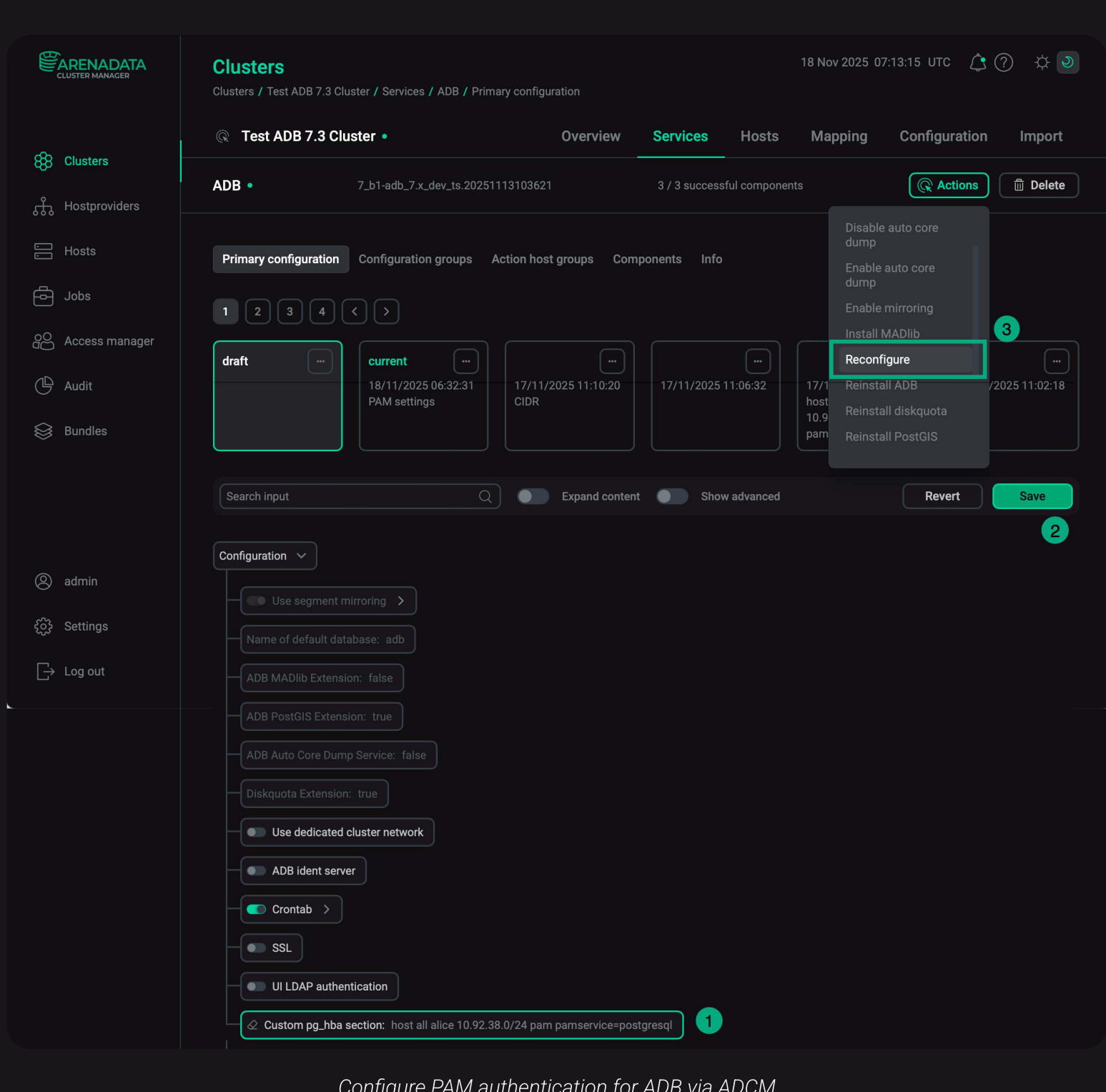
Exit `psql`.

6. Open the ADB service configuration page in ADCM. In the ***Custom pg_hba section*** field, add information about PAM authentication using the following format:

```
host    all                     alice               10.92.38.0/24                        pam pamservice=postgresql
```

The abovementioned line allows the `alice` role to connect to all ADB databases using the `postgresql` PAM module from all hosts with IP addresses in the `10.92.38.0` — `10.92.38.255` range. For production environments, you should change allowed IP addresses and database names in the appropriate way.

To apply changes, click ***Save*** and run the ***Reconfigure*** ADB service action.



Custom pg_hba section: host all alice 10.92.38.0/24 pam pamservice=postgresql

Configure PAM authentication for ADB via ADCM

7. Using `psql`, connect to the `adb` database under the `alice` user:

```
$ psql -d adb -U alice -h 10.92.38.37
```

After running the utility, enter the `alice` user password (see step 1 in the Create a user section). As a result of successful authentication, `psql` prompts you to enter next commands:

```
psql (12.22)
Type "help" for help.
adb=>
```

Use MIT Kerberos KDC in ADB

Daria Barysheva

Contents

To Table of Contents

Step 1. Install and configure MIT Kerberos KDC
Step 2. Create principals for ADB in MIT Kerberos KDC
Step 3. Configure the ADB coordinator and check authentication
Step 4. Map Kerberos principals to ADB roles
Step 5. Configure Kerberos authentication for client applications
Step 6. Configure JDBC connections

Step 1. Install and configure MIT Kerberos KDC

TIP

For information on Kerberos authentication, see the MIT Kerberos documentation.

Below you can find an example of installing and configuring MIT Kerberos KDC on the `bds-kerberos.ru-central1.internal` host with Ubuntu 22.04. The realm name is `RU-CENTRAL1.INTERNAL`. Note that the Kerberos realm name should be uppercase:

1. Install Kerberos server and client packages:

```
$ sudo apt install krb5-kdc krb5-admin-server
```

2. Edit the following Kerberos configuration files using the `vi` (or `vim`) editor.

   - In the /etc/krb5.conf file, fill in the `realms`, `domain_realm` sections and the `default_realm` field of the `libdefaults` section:

```
$ sudo vi /etc/krb5.conf
```

   The file example is given below. In the `kdc` and `admin_server` fields of the `realms` section, specify the host where Kerberos KDC is being installed (with a port number). You can use IP addresses instead of host names:

```
[libdefaults]
 default_realm = RU-CENTRAL1.INTERNAL
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
 default_tgs_enctypes = aes128-cts des3-hmac-sha1 des-cbc-crc des-cbc-md5
 default_tkt_enctypes = aes128-cts des3-hmac-sha1 des-cbc-crc des-cbc-md5
 permitted_enctypes = aes128-cts des3-hmac-sha1 des-cbc-crc des-cbc-md5

[realms]
 RU-CENTRAL1.INTERNAL = {
  kdc = bds-kerberos.ru-central1.internal:88
  admin_server = bds-kerberos.ru-central1.internal:749
  default_domain = ru-central1.internal
 }

[domain_realm]
 .ru-central1.internal = RU-CENTRAL1.INTERNAL
 ru-central1.internal = RU-CENTRAL1.INTERNAL

[appdefaults]
 pam = {
    debug = false
    ticket_lifetime = 36000
    renew_lifetime = 36000
    forwardable = true
    krb4_convert = false
 }
```

   - In the /etc/krb5kdc/kdc.conf file, update the realm name in the `realms` section:

```
$ sudo vi /etc/krb5kdc/kdc.conf
```

   The file example:

```
[kdcdefaults]
    kdc_ports = 750,88

[realms]
    RU-CENTRAL1.INTERNAL = {
        database_name = /var/lib/krb5kdc/principal
        admin_keytab = FILE:/etc/krb5kdc/kadm5.keytab
        acl_file = /etc/krb5kdc/kadm5.acl
        key_stash_file = /etc/krb5kdc/stash
        kdc_ports = 750,88
        max_life = 10h 0m 0s
        max_renewable_life = 7d 0h 0m 0s
        #master_key_type = aes256-cts
        #supported_enctypes = aes256-cts:normal aes128-cts:normal
        default_principal_flags = +preauth
    }
```

3. Use the kdb5_util utility to create a Kerberos KDC database. This database is used to store keys for the Kerberos realms that are managed by the current KDC server. Use the `-s` option to create a stash file: without that file, you will be requested a password every time the KDC server starts.

```
$ sudo kdb5_util create -s
```

   After running the command, enter and confirm a password:

```
Loading random data
Initializing database '/var/kerberos/krb5kdc/principal' for realm 'RU-CENTRAL1.INTERNAL',
master key name 'K/M@RU-CENTRAL1.INTERNAL'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

4. Create the `gpadmin/admin` principal with administrative privileges in Kerberos KDC. To do this, use the kadmin.local utility:

```
$ sudo kadmin.local -q "addprinc gpadmin/admin"
```

   After running the command, enter and confirm a password:

```
Authenticating as principal root/admin@RU-CENTRAL1.INTERNAL with password.
WARNING: no policy specified for gpadmin/admin@RU-CENTRAL1.INTERNAL; defaulting to no policy
Enter password for principal "gpadmin/admin@RU-CENTRAL1.INTERNAL":
Re-enter password for principal "gpadmin/admin@RU-CENTRAL1.INTERNAL":
Principal "gpadmin/admin@RU-CENTRAL1.INTERNAL" created.
```

5. To grant the appropriate permissions to the created principal, edit the /etc/krb5kdc/kadm5.acl file:

```
$ sudo vi /etc/krb5kdc/kadm5.acl
```

   For example, the following file record means that any principal in the `RU-CENTRAL1.INTERNAL` realm with a name ending with `/admin` has all administrative privileges except key extraction:

```
*/admin@RU-CENTRAL1.INTERNAL    *
```

6. Start Kerberos services by running the following commands:

```
$ sudo systemctl start krb5-admin-server
$ sudo systemctl start krb5-kdc
```

7. To check the status of Kerberos services, you can use these commands:

```
$ systemctl status krb5-admin-server
$ systemctl status krb5-kdc
```

   The result of the first command is shown below:

```
● krb5-admin-server.service - Kerberos 5 Admin Server
     Loaded: loaded (/lib/systemd/system/krb5-admin-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2025-11-18 08:03:48 UTC; 2h 7min ago
   Main PID: 140489 (kadmind)
      Tasks: 1 (limit: 19050)
     Memory: 644.0K
        CPU: 38ms
     CGroup: /system.slice/krb5-admin-server.service
             └─140489 /usr/sbin/kadmind -nofork
```

8. To make Kerberos start automatically upon the host restart, run the commands:

```
$ sudo systemctl enable krb5-admin-server
$ sudo systemctl enable krb5-kdc
```

   The result of the first command is shown below:

```
Synchronizing state of krb5-admin-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable krb5-admin-server
```

Step 2. Create principals for ADB in MIT Kerberos KDC

In addition to the principal with administrative permissions (the creation of which is described above), you need to add Kerberos principals for all ADB users who will use Kerberos authentication. Since ordinary users do not need administrative access to the Kerberos server in most cases, you can use the kadmin utility to manage them (instead of `kadmin.local`). Examples in the following sections will use only one principal for demonstration purposes — `gpadmin/admin@RU-CENTRAL1.INTERNAL` (with one corresponding role in ADB — `gpadmin/admin`).

You should also add a service principal to Kerberos KDC, which corresponds not to the role, but to the `postgres` process on the ADB coordinator host. This principal is required when using Kerberos authentication with ADB. A principal name is formed according to the following template:

```
postgres/<coordinator_hostname>@<REALM>
```

where:

- `<coordinator_hostname>` — a name of the ADB coordinator host, which can be obtained via the `hostname` command. If this command returns the fully qualified domain name (FQDN), use it as the parameter value.
- `<REALM>` — a realm name in Kerberos.

Below is an example of adding a service principal:

```
$ sudo kadmin.local -q "addprinc postgres/bds-mdw@RU-CENTRAL1.INTERNAL"
```

Result:

```
Authenticating as principal root/admin@RU-CENTRAL1.INTERNAL with password.
WARNING: no policy specified for postgres/bds-mdw@RU-CENTRAL1.INTERNAL; defaulting to no policy
Enter password for principal "postgres/bds-mdw@RU-CENTRAL1.INTERNAL":
Re-enter password for principal "postgres/bds-mdw@RU-CENTRAL1.INTERNAL":
Principal "postgres/bds-mdw@RU-CENTRAL1.INTERNAL" created.
```

When all required principals (including the service principal) are created, add them to the keytab file using the following command. This keytab file should be copied to the coordinator host of ADB later.

```
$ sudo kadmin.local -q "ktadd -k bds-kerberos.keytab postgres/bds-mdw@RU-CENTRAL1.INTERNAL gpadmin/admin@RU-CENTRAL1.INTERNAL"
```

Result:

```
Authenticating as principal root/admin@RU-CENTRAL1.INTERNAL with password.
Entry for principal postgres/bds-mdw@RU-CENTRAL1.INTERNAL with kvno 2, encryption type aes256-cts-hmac-sha1-96 added to keytab
WRFILE:bds-kerberos.keytab.
Entry for principal postgres/bds-mdw@RU-CENTRAL1.INTERNAL with kvno 2, encryption type aes128-cts-hmac-sha1-96 added to keytab
WRFILE:bds-kerberos.keytab.
Entry for principal gpadmin/admin@RU-CENTRAL1.INTERNAL with kvno 3, encryption type aes256-cts-hmac-sha1-96 added to keytab
WRFILE:bds-kerberos.keytab.
Entry for principal gpadmin/admin@RU-CENTRAL1.INTERNAL with kvno 3, encryption type aes128-cts-hmac-sha1-96 added to keytab
WRFILE:bds-kerberos.keytab.
```

Step 3. Configure the ADB coordinator and check authentication

1. Install Kerberos client libraries on the ADB coordinator host:

```
$ sudo apt install krb5-user sssd-krb5
```

2. Copy the /etc/krb5.conf file contents from the Kerberos server to the */etc/krb5.conf* file on the ADB coordinator host.

3. Copy the keytab file that was generated on the Kerberos server (*bds-kerberos.keytab* in the given example) to the */home/gpadmin/* directory on the ADB coordinator host. Set the ownership and permissions of the keytab file you copied as follows:

```
$ sudo chown gpadmin:gpadmin /home/gpadmin/bds-kerberos.keytab
$ sudo chmod 400 /home/gpadmin/bds-kerberos.keytab
```

4. Log in to the ADB coordinator host under the `gpadmin` user and save a path to the keytab file in the server configuration parameter (GUC) `krb_server_keyfile`:

```
$ sudo su - gpadmin
$ gpconfig -c krb_server_keyfile -v  '/home/gpadmin/bds-kerberos.keytab'
```

   Result:

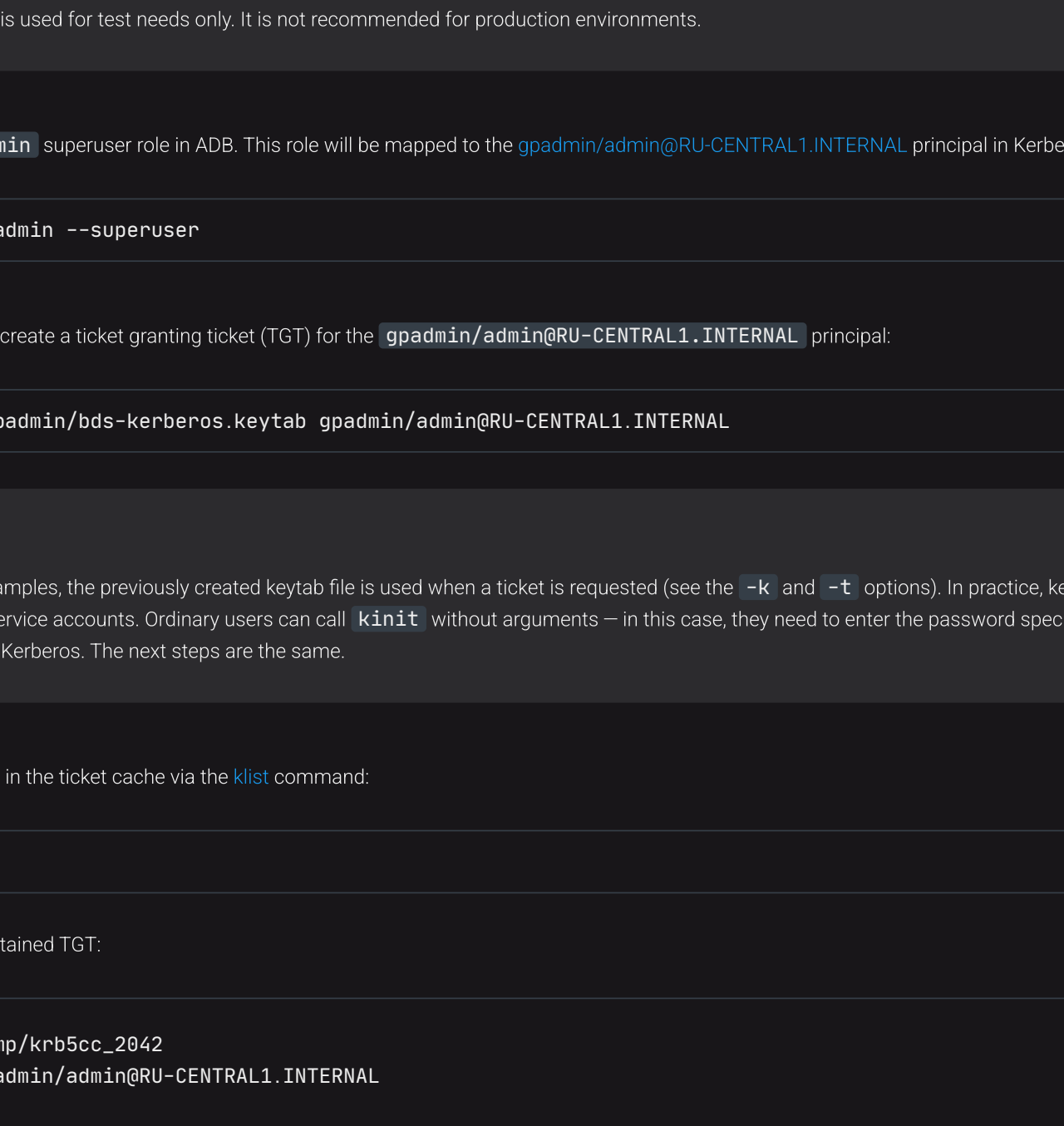
```
20240523:10:50:32:008181 gpconfig:bds-mdw:gpadmin-[INFO]:-completed successfully with parameters '-c krb_server_keyfile -v
/home/gpadmin/bds-kerberos.keytab'
```

5. Open the ADB service configuration page in ADCM. In the ***Custom pg_hba section*** field, add information about Kerberos authentication using the following format:

```
host all all 0.0.0.0/0 gss include_realm=0 krb_realm=RU-CENTRAL1.INTERNAL
```

   For more information on the *pg_hba.conf* file format that is used for Kerberos authentication, see Step 4. Map Kerberos principals to ADB roles. The abovementioned entry allows connecting to all ADB databases from all hosts for those ADB roles whose names coincide with the principal names in the `RU-CENTRAL1.INTERNAL` realm. The `include_realm=0` option means the realm should be stripped from principal names before mapping with ADB role names.

   To apply changes, click ***Save*** and run the ADB service action ***Reconfigure***.



Custom pg_hba section: host all all 0.0.0.0/0 gss include_realm=0 krb_realm=RU-CENTRAL1.INTERNAL

Configure Kerberos authentication for ADB via ADCM

NOTE

The `0.0.0.0/0` address is used for test needs only. It is not recommended for production environments.

6. Create the `gpadmin/admin` superuser role in ADB. This role will be mapped to the gpadmin/admin@RU-CENTRAL1.INTERNAL principal in Kerberos:

```
$ createuser gpadmin/admin --superuser
```

7. Using the kinit command, create a ticket granting ticket (TGT) for the `gpadmin/admin@RU-CENTRAL1.INTERNAL` principal:

```
$ kinit -k -t /home/gpadmin/bds-kerberos.keytab gpadmin/admin@RU-CENTRAL1.INTERNAL
```

NOTE

In this and the following examples, the previously created keytab file is used when a ticket is requested (see the `-k` and `-t` options). In practice, keytab files are most commonly used for service accounts. Ordinary users can call `kinit` without arguments — in this case, they need to enter the password specified for the corresponding principals in Kerberos. The next steps are the same.

8. Check the ticket existence in the ticket cache via the klist command:

```
$ klist
```

   The result contains the obtained TGT:

```
Ticket cache: FILE:/tmp/krb5cc_2042
Default principal: gpadmin/admin@RU-CENTRAL1.INTERNAL

Valid starting       Expires              Service principal
05/23/2024 15:23:38  05/24/2024 15:23:37  krbtgt/RU-CENTRAL1.INTERNAL@RU-CENTRAL1.INTERNAL
```

9. Log in to the `adb` database under the `gpadmin/admin` role via `psql`:

```
$ psql adb -U "gpadmin/admin" -h bds-mdw
```

10. Select the current user name to verify that authentication is successful:

```
SELECT current_user;
```

   Result:

```
 current_user
---------------
 gpadmin/admin
```

11. If you quit `psql` and run `klist` again, the command result will contain not only a TGT, but also a ticket granting service (TGS), which is needed to implement the Kerberos authentication protocol in ADB (see postgres/bds-mdw@RU-CENTRAL1.INTERNAL):

```
Ticket cache: FILE:/tmp/krb5cc_2042
Default principal: gpadmin/admin@RU-CENTRAL1.INTERNAL

Valid starting       Expires              Service principal
05/23/2024 15:23:38  05/24/2024 15:23:37  krbtgt/RU-CENTRAL1.INTERNAL@RU-CENTRAL1.INTERNAL
05/23/2024 15:24:03  05/24/2024 15:23:37  postgres/bds-mdw@
05/23/2024 15:24:03  05/24/2024 15:23:37  postgres/bds-mdw@RU-CENTRAL1.INTERNAL
```

NOTE

The additional entry with the empty realm name (`postgres/bds-mdw@`) is not an error. This is an artifact of the way how host referrals were introduced in krb5 later than 1.6.

Step 4. Map Kerberos principals to ADB roles

In the example listed above, the following template is used to add credentials for the Kerberos authentication to the *pg_hba.conf* file:

```
host                <database>  <user>  <address>     <auth-method>  [<auth-options>]
```

For detailed information on each attribute of this entry, as well as for other possible attributes, see The pg_hba.conf File article in the PostgreSQL documentation. Below, we will focus only on the `<auth-options>` specific for the GSSAPI protocol, which is used for Kerberos authentication. By combining these options, you can choose the most appropriate way to map Kerberos principals to ADB users:

- `include_realm` — if set to `0`, a realm name is excluded from Kerberos principal names before being passed through the mapping with ADB role names; if set to `1` — a realm name is included. The usage of `0` is not secure in multi-realm environments (unless the `krb_realm` option is used). It is recommended to set `include_realm=1` and use the `map` option to convert Kerberos principal names to ADB role names based on the *pg_ident.conf* file.
- `krb_realm` — a name of the Kerberos realm. If this parameter is set, only principals of that realm will be accepted.
- `map` — a name of the custom mapping that allows you to explicitly map Kerberos principals to ADB roles in the *pg_ident.conf* file of the ADB coordinator data directory. Each mapping requires the following format:

```
# MAPNAME       SYSTEM-USERNAME          PG-USERNAME
<map_name>       <principal_name>      <role_name>
```

  where:

  - `<map_name>` — a mapping name that should be specified in the *pg_hba.conf* file: `map=<map_name>`.
  - `<principal_name>` — a name of the Kerberos principal. Should include the realm name (after `@`) if `include_realm=1`.
  - `<role_name>` — a role name in ADB.

The following example shows how to map Kerberos principals to ADB roles via the `map` option:

1. Connect to the coordinator host under the `gpadmin` user and create a new user `myadmin`:

```
$ createuser "myadmin" --superuser;
```

2. In the coordinator data directory, open the *pg_ident.conf* file:

```
$ cd /data1/master/gpseg-1
$ vi pg_ident.conf
```

3. Add the `mymap` mapping to the file:

```
# MAPNAME       SYSTEM-USERNAME          PG-USERNAME
mymap       gpadmin/admin@RU-CENTRAL1.INTERNAL      myadmin
```

4. Open the ADB service configuration page in ADCM. In the ***Custom pg_hba section*** field, replace the previously added value of the ***Custom pg_hba section*** field by the following record:

```
host all all 0.0.0.0/0 gss include_realm=1 krb_realm=RU-CENTRAL1.INTERNAL map=mymap
```

   Note that `include_realm=1`. If the option value is `0`, the previous step would require to use another record:

```
# MAPNAME       SYSTEM-USERNAME          PG-USERNAME
mymap       gpadmin/admin      myadmin
```

5. Click ***Save*** to save changes and run the ***Reconfigure*** action of the ADB service.

CAUTION

You should run the ***Reconfigure*** action each time you edit the *pg_ident.conf* file or the ***Custom pg_hba section*** value.

6. Remove existing tickets (if any) via the kdestroy command:

```
$ kdestroy
```

7. Using the `kinit` command, create a TGT for the `gpadmin/admin@RU-CENTRAL1.INTERNAL` principal:

```
$ kinit -k -t /home/gpadmin/bds-kerberos.keytab gpadmin/admin@RU-CENTRAL1.INTERNAL
```

8. Ensure you can connect to the database using the `myadmin` role:

```
$ psql adb -U "myadmin" -h bds-mdw
```

9. After you successfully connected to ADB, quit `psql` and run `klist` to ensure the service ticket is available:

```
$ klist
```

   Result:

```
Ticket cache: FILE:/tmp/krb5cc_2042
Default principal: gpadmin/admin@RU-CENTRAL1.INTERNAL

Valid starting       Expires              Service principal
05/24/2024 07:21:57  05/25/2024 07:21:57  krbtgt/RU-CENTRAL1.INTERNAL@RU-CENTRAL1.INTERNAL
05/24/2024 07:22:15  05/25/2024 07:21:57  postgres/bds-mdw@
05/24/2024 07:22:15  05/25/2024 07:21:57  postgres/bds-mdw@RU-CENTRAL1.INTERNAL
```

When mapping principals to roles in the *pg_ident.conf* file, you can use regular expressions. For example, the following record maps the `myadmin` role to any principal that belongs to the `RU-CENTRAL1.INTERNAL` realm and has a name ending with `admin`:

```
# MAPNAME       SYSTEM-USERNAME          PG-USERNAME
mymap       /^.*admin@RU-CENTRAL1.INTERNAL$      myadmin
```

For more information on how to map users via the `map` option, see the PostgreSQL documentation.

Step 5. Configure Kerberos authentication for client applications

After Kerberos authentication is configured, client applications need Kerberos tickets for connecting to ADB. On the client side, the `kinit` utility is used to generate tickets. To make this utility work, you should copy the *krb5.conf* configuration file and the keytab file from the Kerberos side and install Kerberos client libraries on the host, where the ADB connection is being configured. The following is an example of configuring a host with Ubuntu 22.04 and connecting from that host to ADB via `psql`:

1. Install Kerberos client libraries on the host, from which you need to connect to the ADB database:

```
$ sudo apt install krb5-user sssd-krb5
```

2. Copy the /etc/krb5.conf contents from Kerberos to the */etc/krb5.conf* file on the client host.

3. Copy the keytab file previously created on the Kerberos server (*bds-kerberos.keytab*) to the home directory of the current user (*/home/gpadmin/* in the given example).

4. Connect to the host under the user, whose home directory contains the keytab file (`gpadmin` in the given example).

```
$ sudo su - gpadmin
```

5. Request a TGT for the `gpadmin/admin@RU-CENTRAL1.INTERNAL` principal via the `kinit` command:

```
$ kinit -k -t /home/gpadmin/bds-kerberos.keytab gpadmin/admin@RU-CENTRAL1.INTERNAL
```

6. Check that you can connect to the ADB database using the `myadmin` role, which creation was described in Step 4. Map Kerberos principals to ADB roles:

```
$ psql adb -U "myadmin" -h bds-mdw
```

7. After you successfully connected to ADB, quit `psql` and run `klist` to ensure the service ticket is available:

```
$ klist
```

   Result:

```
Ticket cache: FILE:/tmp/krb5cc_2042
Default principal: gpadmin/admin@RU-CENTRAL1.INTERNAL

Valid starting       Expires              Service principal
05/30/2024 13:07:02  05/31/2024 13:07:00  krbtgt/RU-CENTRAL1.INTERNAL@RU-CENTRAL1.INTERNAL
05/30/2024 13:07:15  05/31/2024 13:07:00  postgres/bds-mdw@
05/30/2024 13:07:15  05/31/2024 13:07:00  postgres/bds-mdw@RU-CENTRAL1.INTERNAL
```

Kerberos authentication errors

- If the specified ADB role cannot be mapped to any principal in Kerberos KDC, the following error occurs when you try to connect to ADB using that role:

```
psql: FATAL:  authentication failed for user "gpadmin2/admin": valid until timestamp expired
```

- After a ticket expires, the following error is generated:

```
psql: GSSAPI continuation error: Unspecified GSS failure.  Minor code may provide more information
GSSAPI continuation error: Ticket expired
```

- If the specified role can be mapped to the Kerberos principal successfully, but that principal has no previously generated tickets, the following error occurs:

```
psql: GSSAPI continuation error: Unspecified GSS failure.  Minor code may provide more information
GSSAPI continuation error: No Kerberos credentials available (default cache: FILE:/tmp/krb5cc_2042)
```

Step 6. Configure JDBC connections

The JDBC access from Java applications to ADB databases with Kerberos authentication is performed via the Java authentication and authorization service (JAAS). Below you can find how to configure and run a simple Java application on a host with the Ubuntu 22.04 operating system. The application selects table data from ADB. The PostgreSQL JDBC driver is used to connect to ADB:

1. Connect to the `adb` database on the coordinator host and create a test table that will be used in your Java application:

```
CREATE TABLE book_type(id INT PRIMARY KEY, name TEXT UNIQUE NOT NULL)
DISTRIBUTED REPLICATED;
```

2. Add some data to the test table:

```
INSERT INTO book_type VALUES(1,'novel'),(2,'non-fiction'),(3,'fantastic');
```

3. On the host where you plan to run the Java application, follow all steps described in Step 5. Configure Kerberos authentication for client applications. At the end, check that you are log in under the user for whom the tickets were created (`gpadmin` in the given example):

```
$ sudo su - gpadmin
```

4. Ensure that Java is installed:

```
$ java -version
```

   In the given example, Java 11 is used:

```
openjdk version "11.0.23" 2024-04-16 LTS
OpenJDK Runtime Environment (Red_Hat-11.0.23.0.9-2.el7_9) (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.23.0.9-2.el7_9) (build 11.0.23+9-LTS, mixed mode, sharing)
```

5. Create the *.java.login.config* file in the home directory of the current user (*/home/gpadmin* in the given example):

```
$ vi .java.login.config
```

   The file contents are listed below. The purpose of all file options can be found in the Class Krb5LoginModule documentation.

```
pgjdbc {
  com.sun.security.auth.module.Krb5LoginModule required
  doNotPrompt=true
  useTicketCache=true
  debug=true
  client=true;
};
```

6. Create the application file:

```
$ vi test.java
```

   The application code is shown below. This program connects to the `adb` database on the `bds-mdw` coordinator host using the PostgreSQL JDBC driver. Then, the `name` column is read from all tuples of the `book_type` table:

```
import java.sql.*;

public class Test {
    public static void main(String []args) throws Exception {

        Class.forName("org.postgresql.Driver");
        String url = "jdbc:postgresql://bds-mdw:5432/adb?kerberosServerName=postgres&jaasApplicationName=pgjdbc&user=myadmin";

        try ( Connection conn = DriverManager.getConnection(url) ){
            try ( Statement statement = conn.createStatement() ) {
                try (ResultSet rs = statement.executeQuery( "SELECT name FROM book_type") ){
                    while (rs.next())
                        System.out.println( "Get String: " + rs.getString(1));
                }
            }
        }
    }
}
```

URLs for JDBC connections with Kerberos authentication

If Kerberos authentication is used in the ADB cluster, URLs for JDBC connections to ADB databases require the following format:

```
jdbc:postgresql://<coordinator_hostname>:5432/<database_name>?kerberosServerName=postgres&jaasApplicationName=pgjdbc&user=
<role_name>
```

where:

- `<coordinator_hostname>` — a name of the ADB coordinator host, which can be obtained via the `hostname` command. It is important that this name matches the name that was specified in the service principal creation step — otherwise, a Java application will fail with the following error:

```
KrbException: Server not found in Kerberos database (7) - LOOKING_UP_SERVER
```

- `<database_name>` — a name of the ADB database.
- `<role_name>` — a name of the ADB role, which was previously mapped to the Kerberos principal. In the given example, the `myadmin` role is used. Its creation is described in Step 4. Map Kerberos principals to ADB roles.

7. Upload the JAR file of the PostgreSQL JDBC driver and specify its location in the `CLASSPATH` environment variable:

```
$ export CLASSPATH=.:/home/gpadmin/postgresql-42.7.3.jar
```

8. Run the application:

```
$ java test.java
```

   The successful result is shown below:

```
Debug is  true storeKey false useTicketCache true useKeyTab false doNotPrompt true ticketCache is /tmp/krb5cc_2042 isInitiator
true KeyTab is null refreshKrb5Config is false principal is null tryFirstPass is false useFirstPass is false storePass is false
clearPass is false
Acquire TGT from Cache
Principal is gpadmin/admin@RU-CENTRAL1.INTERNAL
Commit Succeeded

Get String: novel
Get String: non-fiction
Get String: fantastic
```

Accounts in ADB

Daria Barysheva

Contents

To Table of Contents

- Linux users
- Arenadata DB roles
- Accounts used in the ADB Control databases
- ADB Control UI users
- ADBM API user
- ADCM user
- LDAP accounts
  - ADB
  - ADB Control

This article lists all accounts that are used in Arenadata DB.

Linux users

The following accounts are created in an operating system during ADB installation (*/etc/passwd*).

Account	Description	Default password
adcm	The user that is used by ADCM to deploy clusters	Not set
gpadmin	The Linux user for ADB that is needed to start/stop a cluster and run all console utilities. Also, it has access to all cluster databases as a superuser (with the `trust` method in *pg_hba.conf*). You can change this username via the ***System user name*** setting in configuration parameters of the ADB service	Not set
pxf	The user for PXF. Assigned `nologin` in */etc/passwd*	Not set

Arenadata DB roles

The following accounts are created in the Arenadata DB database during ADB installation (via the `CREATE ROLE` command).

Account	Description	Default password	How to change a password
gpadmin	The ADB cluster administrator. This role has access to all cluster databases as a superuser	Not set	Via the `psql` utility: `ALTER ROLE gpadmin WITH ENCRYPTED PASSWORD '<new_password>';` There is no ability to set/change a password via ADCM
adcc	The role that is used by ADB Control for connecting to ADB. Such access is required to obtain information about the cluster topology and to cancel requests initiated by users. The username can be changed on the ADB Control configuration page during the primary ADB Control installation. Not possible to change after. For more information, see Configuration parameters (***ADB Control → UI component parameters → ADB user name***)	Not set by default. Cannot be set during ADB installation. A password is stored in `pg_shadow`	1. Run the following command via the `psql` utility: `ALTER ROLE adcc WITH ENCRYPTED PASSWORD '<new_password>';` 2. Change the password on the ***Configuration → Clusters*** tab in the ADB Control UI. For more information, see Edit a cluster connection. There is no ability to set/change a password via ADCM

Accounts used in the ADB Control databases

The following accounts are created in the ADB Control databases (containerized or external) during ADB Control installation.

Account	Description	Default password	How to change a password
postgres	The role name in the PostgreSQL database, which is used in ADB Control to store information on data queries (see Query DB). The username can be changed on the ADB Control configuration page during ADB Control installation. For more information, see Configuration parameters (***ADB Control → Database parameters → Username*** or ***ADB Control → External database parameters → Username*** for external PostgreSQL)	The default value is `123`. Can be set on the ADB Control configuration page during ADB Control installation. For more information, see Configuration parameters (***ADB Control → Database parameters → Password*** or ***ADB Control → External database parameters → Password*** for external PostgreSQL)	For external PostgreSQL, you can change a password via ADCM anytime. For containerized PostgreSQL, you can set a password only during the primary ADB Control installation
default	The account name in the ClickHouse database, which is used in ADB Control to store information on metrics (see Metrics DB). The username can be changed on the ADB Control configuration page during ADB Control installation. For more information, see Configuration parameters (***ADB Control → ADQM database parameters → Username*** or ***ADB Control → External ADQM parameters → Username*** for external ClickHouse)	Not set by default. Can be set on the ADB Control configuration page during ADB Control installation. For more information, see Configuration parameters (***ADB Control → ADQM database parameters → Password*** or ***ADB Control → External ADQM parameters → Password*** for external ClickHouse). If set, a password is stored in the ADCM vault and in ClickHouse	For external ClickHouse, you can change a password via ADCM anytime. For containerized ClickHouse, you can set a password only during the primary ADB Control installation

ADB Control UI users

The following users are created during ADB Control installation.

Account	Description	Default password	How to change a password

admin	The ADB Control owner, which is used to work with ADB Control UI	The default value is `1234`.
		When you first log in to ADB Control, you should change a password. For more information, see Connect to ADB Control.
		A password is stored in the `adcc.user` table of the PostgreSQL database (containerized or external):
		`$ psql -d adcc`
		`SELECT * FROM adcc.user;`
			You can change a password on the ***Users → Management*** tab in the ADB Control UI. For more information, see Edit a user

adcm	The service API user (hidden administrator of the ADB Control UI)	Can be set on the ADB Control configuration page during ADB Control installation. For more information, see Configuration parameters (***ADB Control → UI component parameters → Service password***). If not set, a secret password is generated. For backward compatibility, it is the same across all clusters. A password is stored in encrypted form in the `adcc_database_1` container: `SELECT password FROM adcc.system_user WHERE username = 'adcm';`	Password can be reset on the ADB Control configuration page (***ADB Control → UI component parameters → Service password***)
adbm	The service API user (hidden administrator of the ADB Control UI) for collecting information about storage and cluster modifications that is displayed on the ***Audit*** page	Can be set on the ADBM configuration page. For more information, see Configuration parameters (***ADBM → Password for adbm system-user***). A password is stored in encrypted form in the `adcc_database_1` container: `SELECT password FROM adcc.system_user WHERE username = 'adbm';`	Password can be changed on the ADBM configuration page (***ADBM → Password for adbm system-user***)

ADBM API user

The following user is created during ADBM installation.

Account	Description	Default password	How to change a password

adcc	The user that is used for authentication in the ADBM CLI API	Stored as plaintext in the */opt/adbm/auth.properties* file. The default value is `1234`:
		`user.adcc = 1234`
		File permissions:
		`-rw-r--r-- 1 root root`
			Update the */opt/adbm/auth.properties* file. The ADBM restart is required

ADCM user

The following user is created during ADCM installation.

Account	Description	Default password	How to change a password
admin	ADCM admin	admin	After ADCM installation is complete, you can change a user password in the ADCM web UI. For more information, see Check the ADCM web-interface

LDAP accounts

When configuring LDAP authentication, you need to enter and store LDAP admin accounts. Passwords are stored as plaintext in specific files that should be secured with OS file permissions.

ADB

LDAP authentication in ADB is configured according to the LDAP authentication in ADB article. To specify the LDAP admin account and its password, fill in the ***SA Login*** and ***Password*** fields on the ADB configuration page.

LDAP credentials are stored in the following file on the ADB coordinator host: *$COORDINATOR_DATA_DIRECTORY/pg_hba.conf*.

File permissions:

```
-rw-r----- 1 gpadmin gpadmin
```

Synchronization of ADB users and groups with LDAP is performed automatically via the pg-ldap-sync utility. For more information, see Synchronize ADB and LDAP.

IMPORTANT

After syncing with the LDAP server, two ADB local roles are still required since they cannot be replaced with LDAP users (see Arenadata DB roles):

- `gpadmin`
- `gpmon`

All other users (including superusers) can be controlled via the LDAP server.

ADB Control

LDAP authentication in ADB Control is configured according to the LDAP authentication article. To specify the LDAP admin account and its password, fill in the ***Login*** and ***Password*** fields on the ADB Control configuration page.

LDAP credentials are stored in the following files on the ADB Control host:

- */opt/adcc/docker-compose.yaml*. File permissions:

```
-rw------- 1 root root
```

- */opt/adcc/adcc-ui-server-application.yml*. File permissions:

```
-rw-r--r--  1 root root
```

If LDAP authentication is configured, you need to map user groups of the selected LDAP server to roles in ADB Control. You can manage such mappings on the ***Users → LDAP group mapping*** tab in the ADB Control web interface. For more information, see Configure role mappings with LDAP groups.

IMPORTANT

After you assign the `Administrator` or `Owner` role to specific LDAP users and delete or block the default local user `admin` (see ADB Control UI users) in ADB Control, all authentication can be controlled via the LDAP server.

Cluster actions

Daria Barysheva

Contents

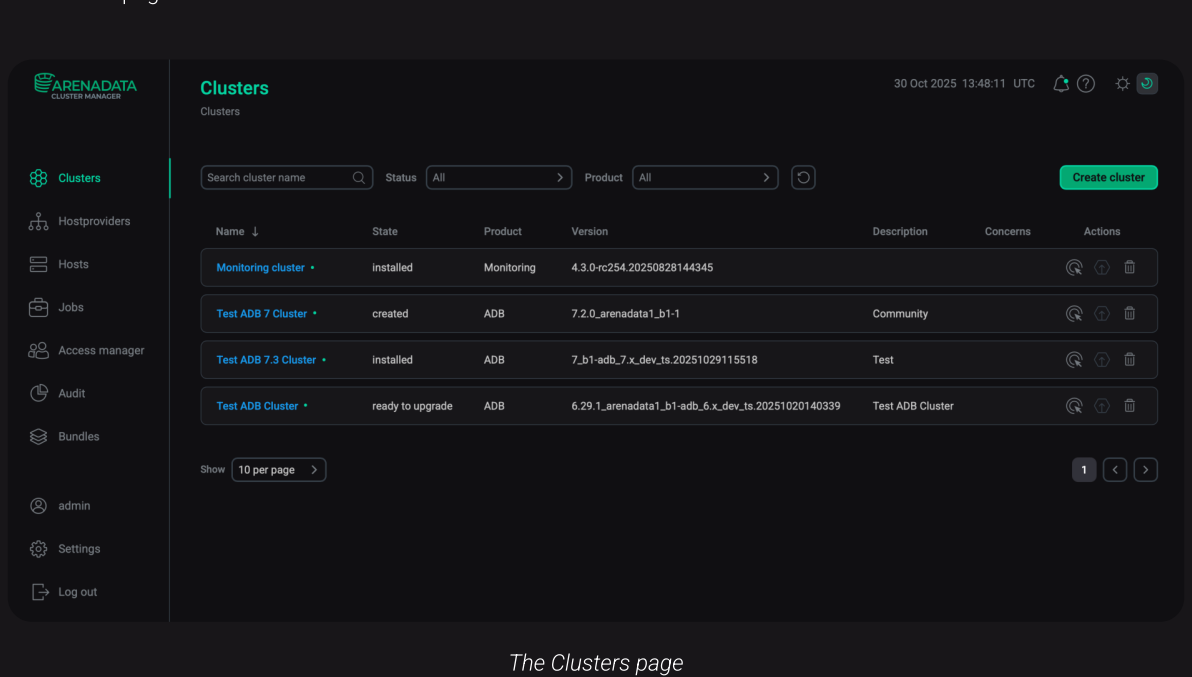
To Table of Contents

- Overview
- Activate standby
- Activate standby postprocess
- Init Standby Master
- Move Host Segment
- Expand
  - Prepare
  - Run the action
- Reinstall
- Reinstall statuschecker
- Reconfigure Vault integration
- Reconfigure parameter archiving
- Check
- Start
- Stop
- Upgrade

This article describes the actions available for the ADB cluster in the ADCM UI.

Overview

You can find cluster actions on the ***Clusters*** page.



ARENADATA CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
30 Oct 2025 13:48:11 UTC
Search cluster name
Status All
Product All
Create cluster
Show 10 per page

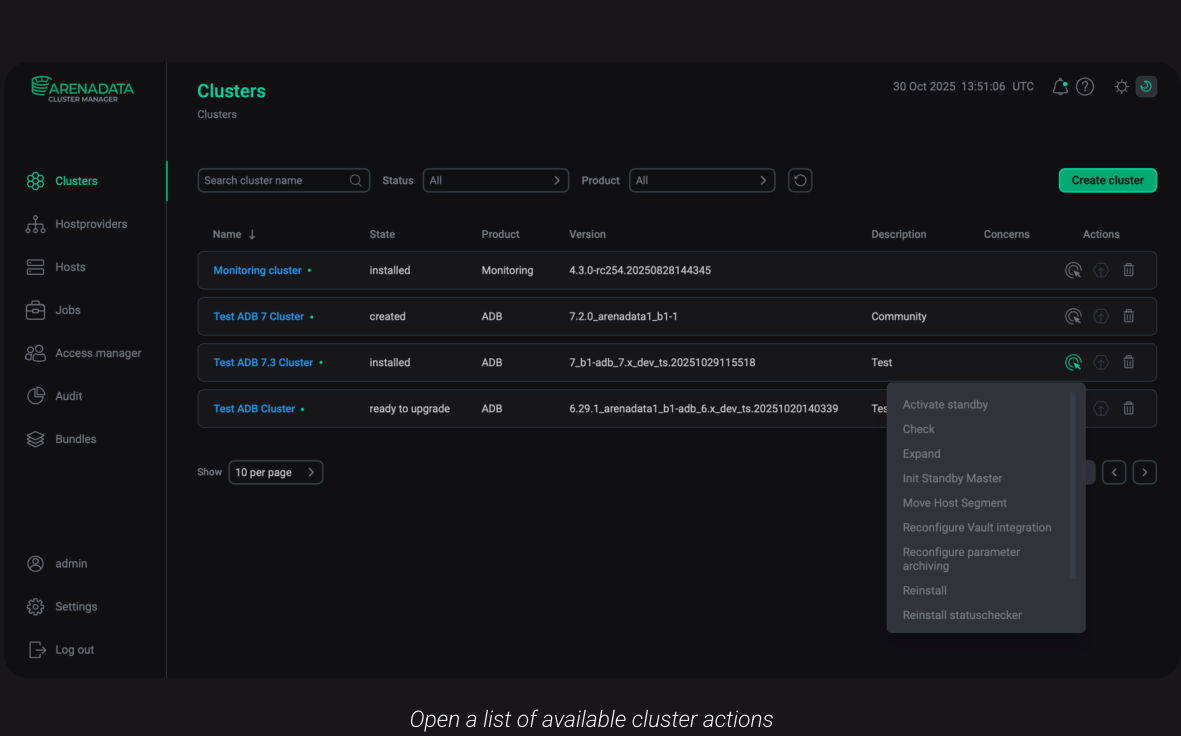
Name	State	Product	Version	Description	Concerns	Actions
Monitoring cluster	installed	Monitoring	4.3.0-rc254.20250828144345			
Test ADB 7 Cluster	created	ADB	7.2.0_arenadata1_b1-1	Community		
Test ADB 7.3 Cluster	installed	ADB	7_b1-adb_7.x_dev_ts.20251029115518	Test		
Test ADB Cluster	ready to upgrade	ADB	6.29.1_arenadata1_b1-adb_6.x_dev_ts.20251020140339	Test ADB Cluster		

The Clusters page

Refer to ADCM documentation for an overview of the ***Clusters*** page and common actions.

The ***Actions*** column shows icons for managing the cluster:

- — opens the drop-down list that offers actions to manage the cluster.



ARENADATA CLUSTER MANAGER
Clusters
30 Oct 2025 13:51:06 UTC
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
Activate standby
Check
Expand
Init Standby Master
Move Host Segment
Reconfigure Vault integration
Reconfigure parameter archiving
Reinstall
Reinstall statuschecker

Name	State	Product	Version	Description	Concerns	Actions
Monitoring cluster	installed	Monitoring	4.3.0-rc254.20250828144345			
Test ADB 7 Cluster	created	ADB	7.2.0_arenadata1_b1-1	Community		
Test ADB 7.3 Cluster	installed	ADB	7_b1-adb_7.x_dev_ts.20251029115518	Test		
Test ADB Cluster	ready to upgrade	ADB	6.29.1_arenadata1_b1-adb_6.x_dev_ts.20251020140339	Tes		

Open a list of available cluster actions

- — indicates whether a new version of a bundle is available and allows you to upgrade the cluster.
- — deletes information about the cluster from ADCM (it does not remove ADB or make any changes to hosts that belong to the cluster).

A set of cluster actions (available after clicking the icon) depends on the current ADB cluster status.

Available actions depending on the ADB status

Status	Condition	Available actions
created	The ADB cluster was created via ADCM, but not installed yet	Precheck Install
installed	The ADB cluster was successfully installed via ADCM	Activate standby Check Expand Init Standby Master Move Host Segment Reconfigure Vault integration Reconfigure parameter archiving Reinstall Reinstall statuschecker Start Stop
activating_standby	As a result of the Activate standby action, the standby coordinator was activated, but some errors occurred during the subsequent configuration process	Activate standby postprocess

For information on the ***Precheck*** and ***Install*** actions, see Install a cluster. Other actions available for an already installed cluster are listed below.

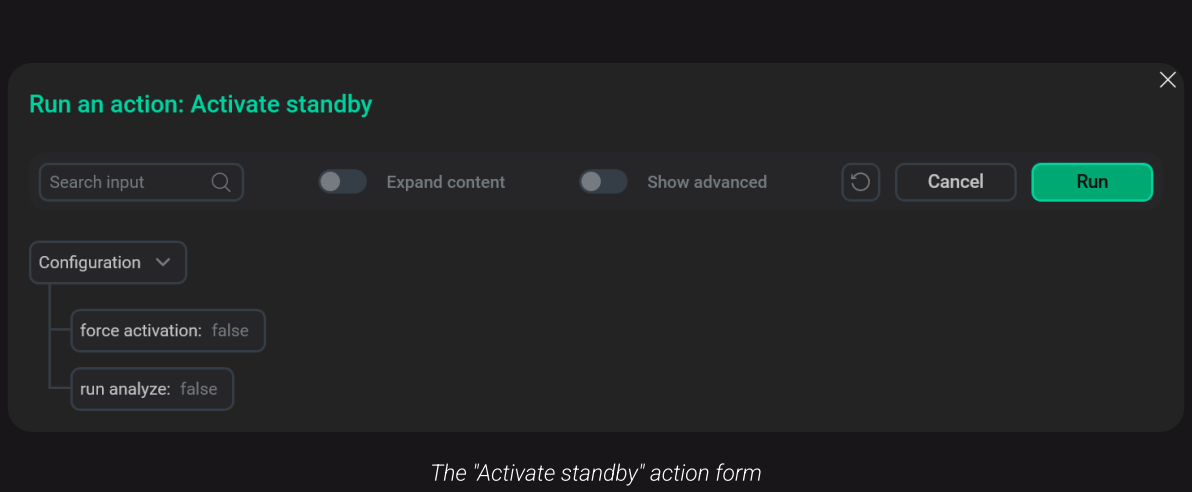
Activate standby

The ***Activate standby*** action activates the standby coordinator in case of failures on the current coordinator.

> IMPORTANT
>
> - You can activate the standby coordinator when a host of the current coordinator is unavailable. To switch to standby if there are no errors on the coordinator host (it is available), you should first stop the ADB service and set the ***force activation*** flag in the ***Activate standby*** action parameters. Otherwise, the action fails.
> - Before running the ***Activate standby*** action, you should start the ADB Control and ADBM services since their reconfiguration is needed during the action launch. Otherwise, the action fails (but you can fix it by starting services and running the Activate standby postprocess action).

After you select the ***Activate standby*** action, a dialog box opens in which you can fill in the following options:

- ***force activation*** — a flag that indicates whether to force switching to the standby coordinator when the host of the active coordinator is available, but the coordinator process inside the operating system is stopped;
- ***run analyze*** — a flag that specifies whether to run `ANALYZE` after switching to standby in all databases except `template0`, `template1`, and `postgres`.

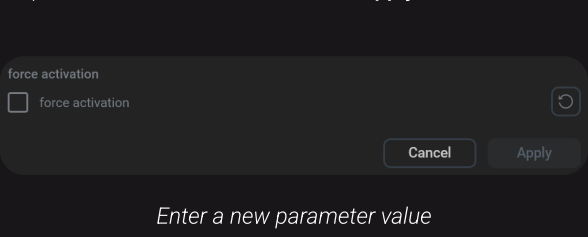


Run an action: Activate standby
Search input
Expand content
Show advanced
Cancel
Run
Configuration

force activation: false
run analyze: false

The "Activate standby" action form

To edit any parameter, click its current value. In the window that opens, enter a new value and click ***Apply***.



force activation
force activation
Cancel
Apply

Enter a new parameter value

To run the ***Activate standby*** action, click ***Run*** in the action form. Then, confirm the action in the standard confirmation dialog.

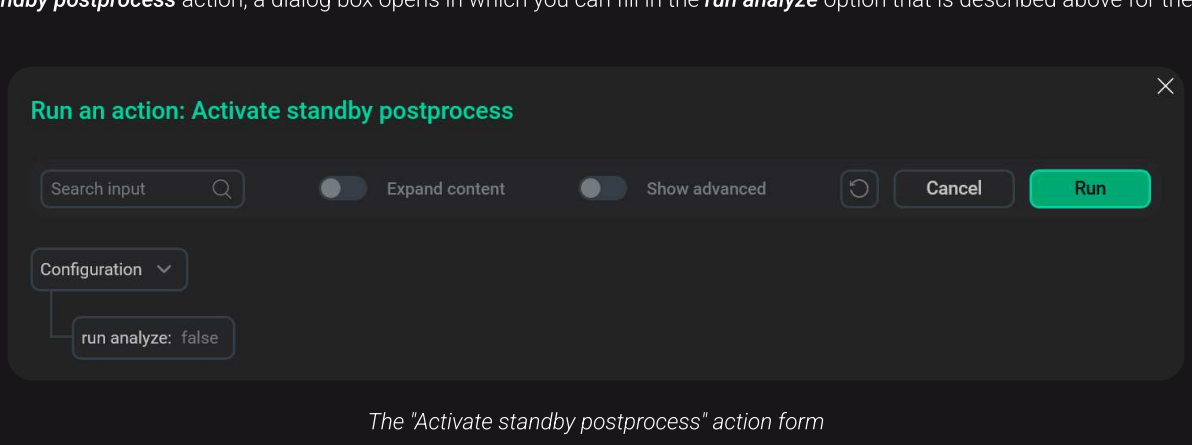
While the action is running, the ADB cluster has the `activating_standby` status. If the action succeeds, the cluster status changes to `installed`. If the standby coordinator is activated, but some errors occur during the subsequent configuration process — the cluster remains in the `activating_standby` status, and the Activate standby postprocess becomes available in the list of cluster actions. In this case, you need to determine the error cause on the ***Jobs*** page, make fixes, and run the ***Activate standby postprocess*** action.

> **Postprocessing**
>
> - To create `cron` tasks on the new coordinator, run the following actions after the ***Activate standby*** action is completed:
>   - Reconfigure parameter archiving for the ADB cluster — to have the ability to export ADB settings according to the schedule specified via the Parameter archiving/Parameter archiving job setting cluster parameters.
>   - ***Reconfigure*** for the ADB service — to have the ability to run scripts that interact with the arenadata_toolkit schema.
> - After the host of the previous coordinator becomes available (or immediately after running the ***Activate standby*** action if switching to standby was not caused by the coordinator host unavailability), follow the steps on that host:
>   - Delete the `cron` tasks under the ADB administrator (`gpadmin` by default). This is necessary to prevent attempts to access the outdated coordinator.
>
> ```
> $ sudo su - gpadmin
> $ crontab -r
> ```
>
>   - Rename or remove the coordinator data catalog (*/data1/master* by default) to avoid accidentally running an outdated coordinator and be able to place a new standby coordinator later.
> - After the standby coordinator is activated, an old coordinator stops participating in the ADB cluster operations. The standby coordinator is no longer available, as it has become a coordinator. To add a new standby coordinator, run the Init Standby Master action.

Activate standby postprocess

The ***Activate standby postprocess*** action becomes available if the standby coordinator is activated during the Activate standby action, but some errors occur during the subsequent configuration process. After fixing errors, run ***Activate standby postprocess*** to finish processing.

After you select the ***Activate standby postprocess*** action, a dialog box opens in which you can fill in the ***run analyze*** option that is described above for the Activate standby action.



Run an action: Activate standby postprocess
Search input
Expand content
Show advanced
Cancel
Run
Configuration

run analyze: false

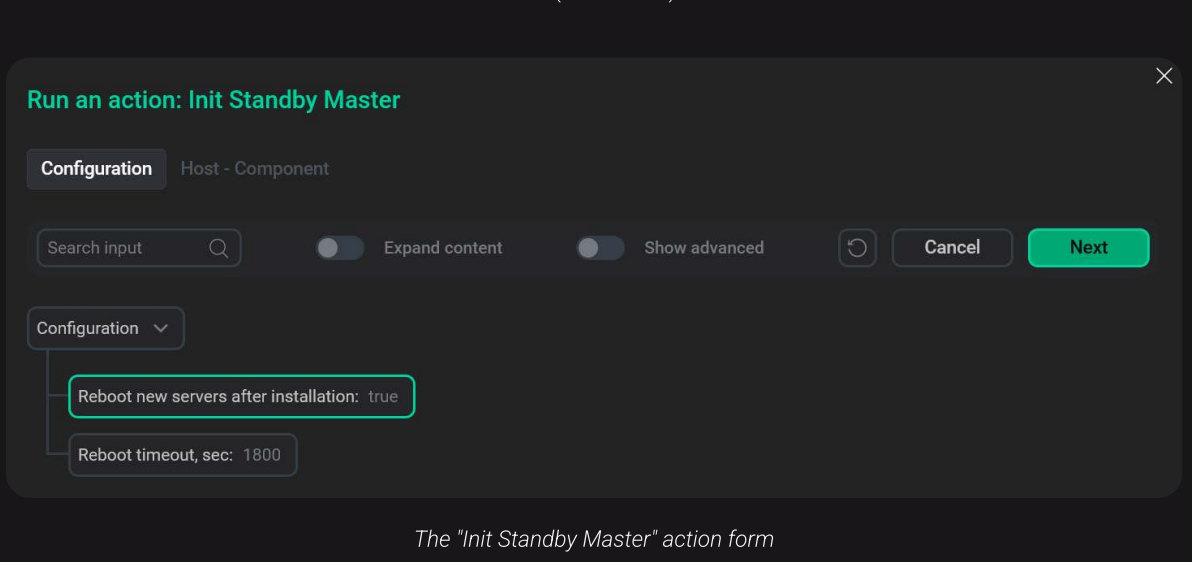
The "Activate standby postprocess" action form

To run the ***Activate standby postprocess*** action, click ***Run*** in the action form. Then, confirm the action in the standard confirmation dialog.

Init Standby Master

The ***Init Standby Master*** action allows you to add or remove the standby coordinator. After you select the action, follow the steps:

1. On the ***Configuration*** tab of the window that opens, fill in the options:
   - ***Reboot new servers after installation*** — a flag that indicates whether to reboot ADB hosts automatically after the ***Init Standby Master*** action is completed. If the flag is reset, you need to restart manually.
   - ***Reboot timeout, sec*** — a timeout that is used to wait for the reboot of ADB hosts (in seconds).

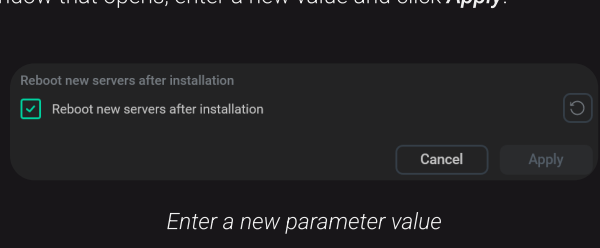


Run an action: Init Standby Master
Configuration
Host - Component
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Reboot new servers after installation: true
Reboot timeout, sec: 1800

The "Init Standby Master" action form

To edit any parameter, click its current value. In the window that opens, enter a new value and click ***Apply***.

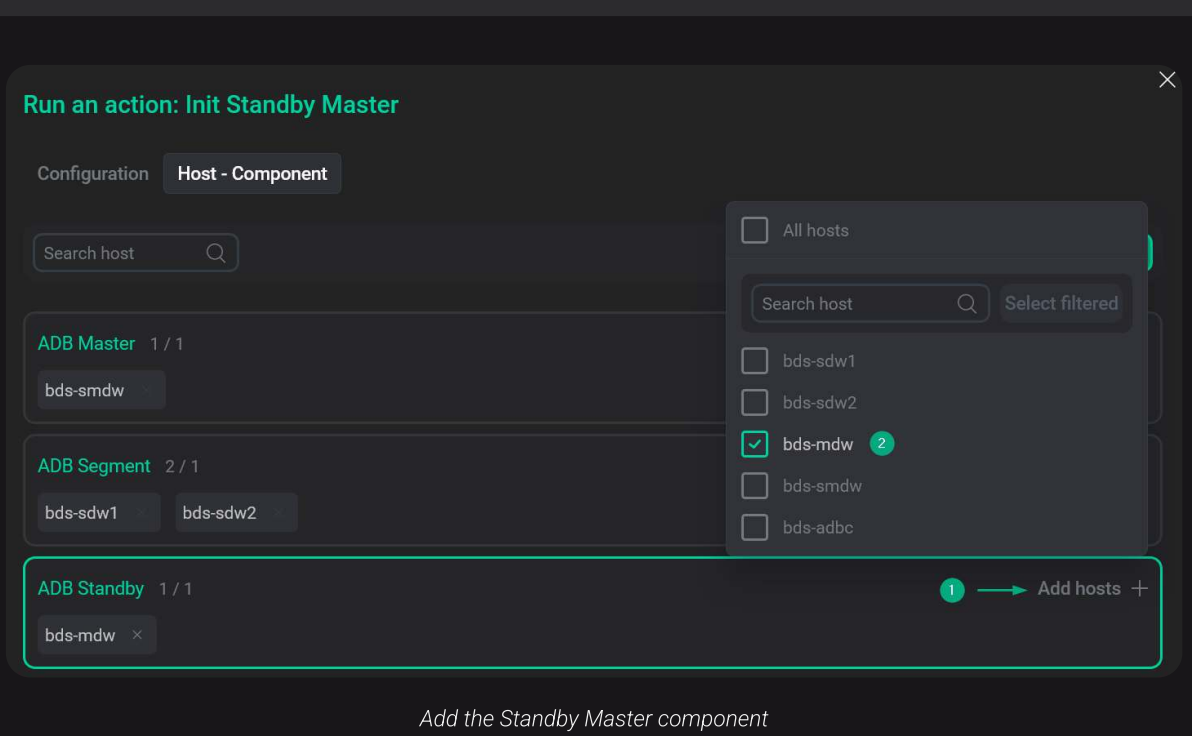


Reboot new servers after installation
Reboot new servers after installation
Cancel
Apply

Enter a new parameter value

2. Click ***Next***.
3. On the ***Host - Component*** tab that opens, perform one of the following actions:
   - When adding standby — place the ***ADB Standby*** component on a separate host. To do this, click ***Add hosts*** in the component section and select a host in the dialog that opens.

> NOTE
>
> You can place the standby coordinator on the host that was used for the previous coordinator (before applying the Activate standby action). But you should first rename or delete the data catalog of the outdated coordinator (*/data1/master* by default).

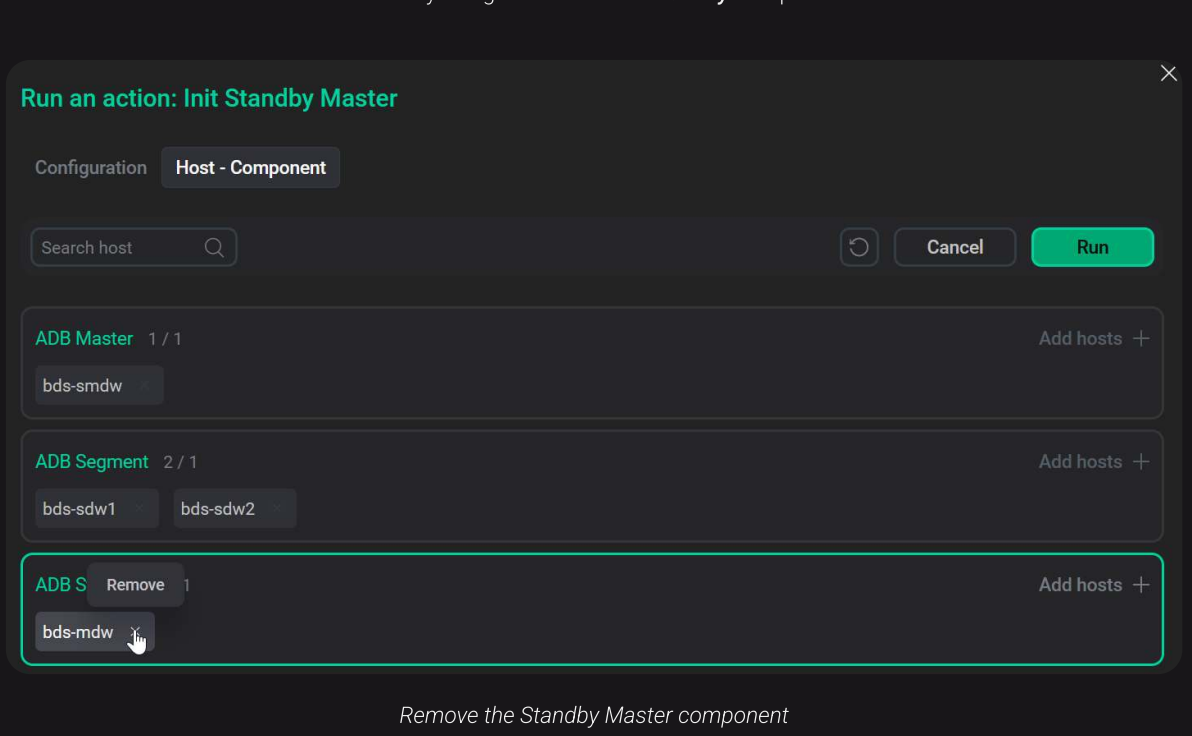


Run an action: Init Standby Master
Configuration
Host - Component
Search host
All hosts
Search host
Select filtered
bds-sdw1
bds-sdw2
bds-mdw
2
bds-smdw
bds-adbc
1
Add hosts

ADB Master 1 / 1: bds-smdw
ADB Segment 2 / 1: bds-sdw1, bds-sdw2
ADB Standby 1 / 1: bds-mdw

Add the Standby Master component

   - When removing standby — click ***Remove*** for the host that is currently assigned to the ***ADB Standby*** component.



Run an action: Init Standby Master
Configuration
Host - Component
Search host
Cancel
Run
Add hosts
Add hosts
Remove
Add hosts

ADB Master 1 / 1: bds-smdw
ADB Segment 2 / 1: bds-sdw1, bds-sdw2
ADB S… 1: bds-mdw

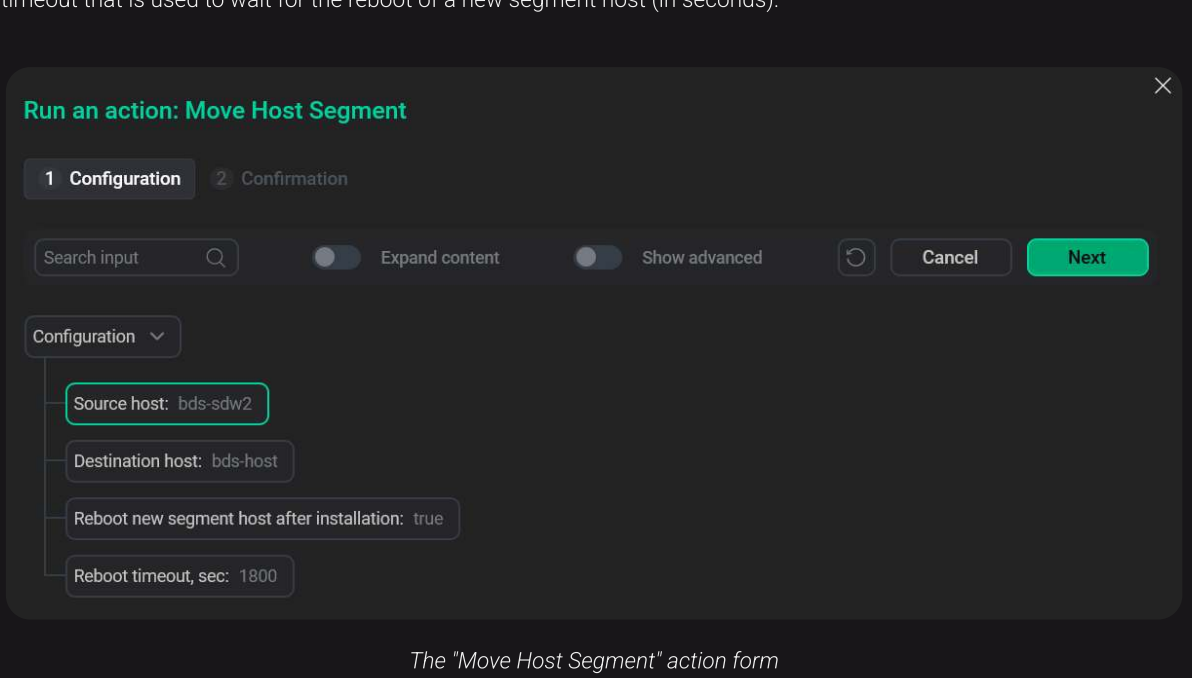
Remove the Standby Master component

4. Click ***Run***.
5. Confirm the action in the standard confirmation dialog by clicking ***Run***.

Move Host Segment

The ***Move Host Segment*** action is available in the Enterprise Edition of ADB. This action allows you to move segments from one ADB cluster host to another. After you select the action, follow the steps:

1. On the ***Configuration*** tab of the window that opens, fill in the options:
   - ***Source host*** — a source host from which you need to move data segments. You can select one of the ADB cluster hosts where the ADB Segment component is installed.
   - ***Destination host*** — a target host for data segments that should be moved from the source host. You can select one of the ADB cluster hosts that is not mapped to the ADB Segment component.
   - ***Reboot new segment host after installation*** — a flag that indicates whether to reboot a new ADB segment host after the ***Move Host Segment*** action is completed. If the flag is reset, you need to restart the host manually to apply configuration settings.
   - ***Reboot timeout, sec*** — a timeout that is used to wait for the reboot of a new segment host (in seconds).

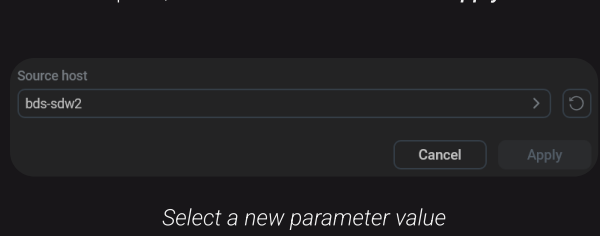


Run an action: Move Host Segment
1 Configuration
2 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Source host: bds-sdw2
Destination host: bds-host
Reboot new segment host after installation: true
Reboot timeout, sec: 1800

The "Move Host Segment" action form

To edit any parameter, click its current value. In the window that opens, enter a new value and click ***Apply***.



Source host
bds-sdw2
Cancel
Apply

Select a new parameter value

2. Click ***Next***.
3. Confirm the action in the standard confirmation dialog by clicking ***Run***.

> NOTE
>
> In addition to the movement of segments, the ***Move Host Segment*** cluster action performs the following for a new segment host:
>
> - Environment preparation, including installation of ADB binaries, application of necessary configurations, etc.
> - Installation of the components that are added to all segment hosts explicitly or implicitly by other services (for example, PXF, ADB Control/ADBM agents, monitoring, etc.).

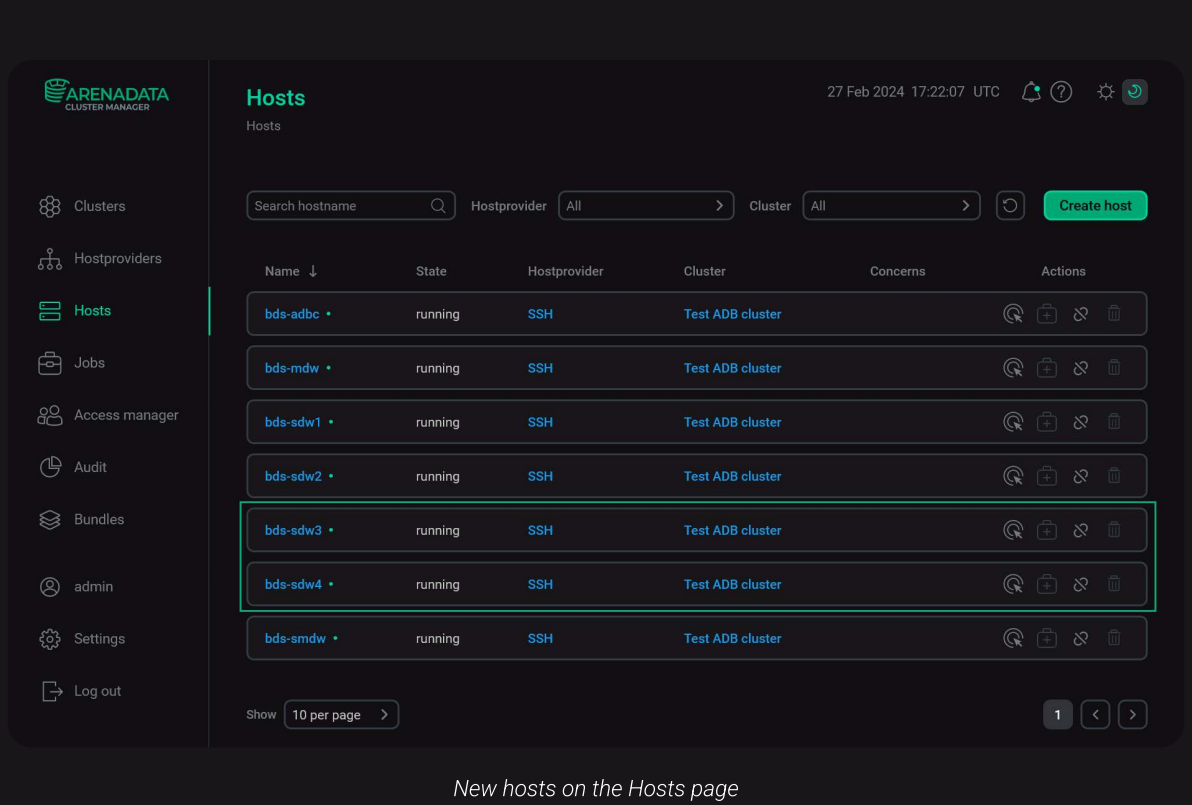
Expand

The ***Expand*** action allows you to add new segment hosts to the ADB cluster.

Prepare

Before starting the cluster expansion:

1. Create and configure new hosts on the ***Hosts*** page in ADCM. Use one of the available hostproviders.
2. If a host was created using the SSH hostprovider, apply the ***Check connection*** and ***Install statuschecker*** actions. Ensure that all actions are completed successfully and all hosts are available.
3. Add new hosts to the ADB cluster.



ARENADATA CLUSTER MANAGER
Hosts
27 Feb 2024 17:22:07 UTC
Search hostname
Hostprovider All
Cluster All
Create host
Show 10 per page

Name	State	Hostprovider	Cluster	Concerns	Actions
bds-adbc	running	SSH	Test ADB cluster		
bds-mdw	running	SSH	Test ADB cluster		
bds-sdw1	running	SSH	Test ADB cluster		
bds-sdw2	running	SSH	Test ADB cluster		
bds-sdw3	running	SSH	Test ADB cluster		
bds-sdw4	running	SSH	Test ADB cluster		
bds-smdw	running	SSH	Test ADB cluster		

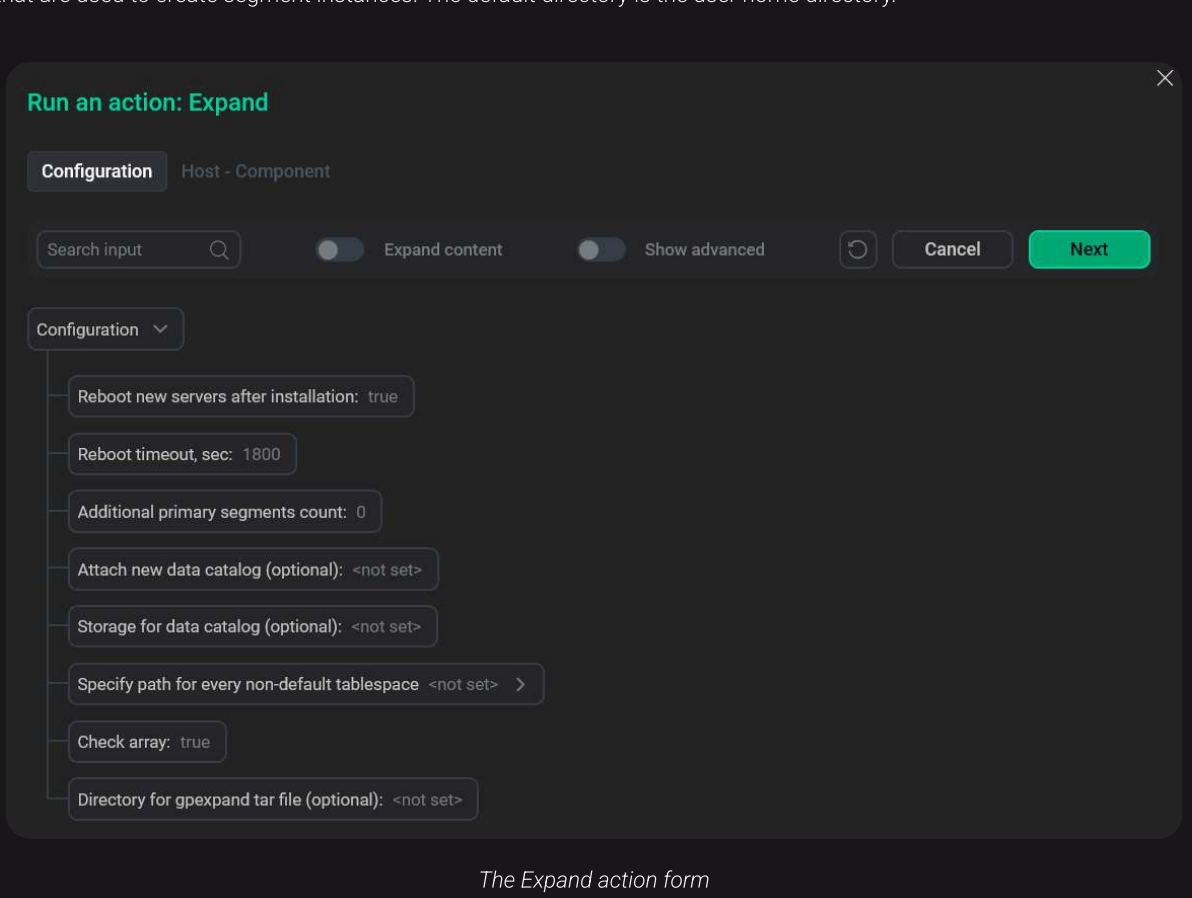
New hosts on the Hosts page

> CAUTION
>
> If the ADB cluster uses mirroring, the number of new hosts should be sufficient to apply one of the mirroring policies: spread or group. If the number of new hosts is greater than the number of segments per host in the current cluster configuration, the spread mirroring will be applied during the cluster expansion, otherwise — group. The group mirroring policy requires at least 2 new hosts, otherwise the action fails with the following error: `Not enough hosts for grouped mirroring. You must have at least 2`.

Run the action

After selecting the ***Expand*** action, follow the steps:

1. On the ***Configuration*** tab of the window that opens, switch on the ***Show advanced*** toggle and fill in the following options:
   - ***Reboot new servers after installation*** — a flag that indicates whether to reboot new ADB hosts automatically after the ***Expand*** action is completed. A restart is required to apply some of the settings that are changed during the installation process. If the flag is reset, you need to restart manually.
   - ***Reboot timeout, sec*** — a timeout that is used to wait for the reboot of ADB hosts (in seconds).
   - ***Additional primary segments count*** — a number of primary segments that should be added to all segment hosts (including existing hosts). For example, if existing hosts have two primary segments, using the parameter value `2` will initialize two additional primary segments on existing hosts and four primary segments on new hosts. In addition, mirror segments will be added for these new primary segments if mirroring is enabled. The default parameter value is `0`, which means that new hosts are configured with the same number of primary segments as existing hosts have.
   - ***Attach new data catalog (optional)*** — a directory that will be used to store new segments. All directories used as data catalogs should contain an equal number of segments to avoid data skew. Expansion will fail if segments are unevenly distributed across data catalogs. If the parameter is empty, additional segments are evenly distributed across existing data catalogs.
   - ***Storage for data catalog (optional)*** — a name of the block storage device that should be mounted to the directory specified via the ***Attach new data catalog (optional)*** parameter. For example, `sdc` (without the `/dev` prefix).
   - ***Specify path for every non-default tablespace*** — paths to the directories that are used to store tablespace data. Paths should be specified if the ***Attach new data catalog (optional)*** parameter is filled in.
   - ***Check array*** — a flag that indicates whether to check that all hosts have an equal number of primary and mirror segments and the same segment name suffixes in the `-<n>` format (`-1`, `-2`, `-3`, etc.).
   - ***Directory for gpexpand tar file (optional)*** — a fully qualified path to the directory on segment hosts where the `gpexpand` utility copies a temporary TAR file. This archive contains database files that are used to create segment instances. The default directory is the user home directory.

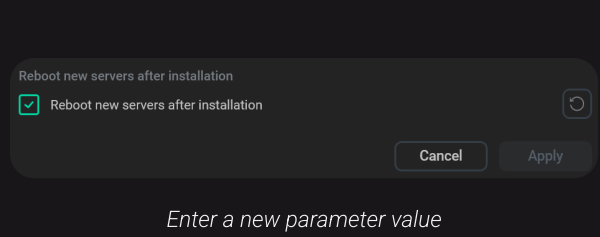


Run an action: Expand
Configuration
Host - Component
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Reboot new servers after installation: true
Reboot timeout, sec: 1800
Additional primary segments count: 0
Attach new data catalog (optional): <not set>
Storage for data catalog (optional): <not set>
Specify path for every non-default tablespace <not set>
Check array: true
Directory for gpexpand tar file (optional): <not set>

The Expand action form

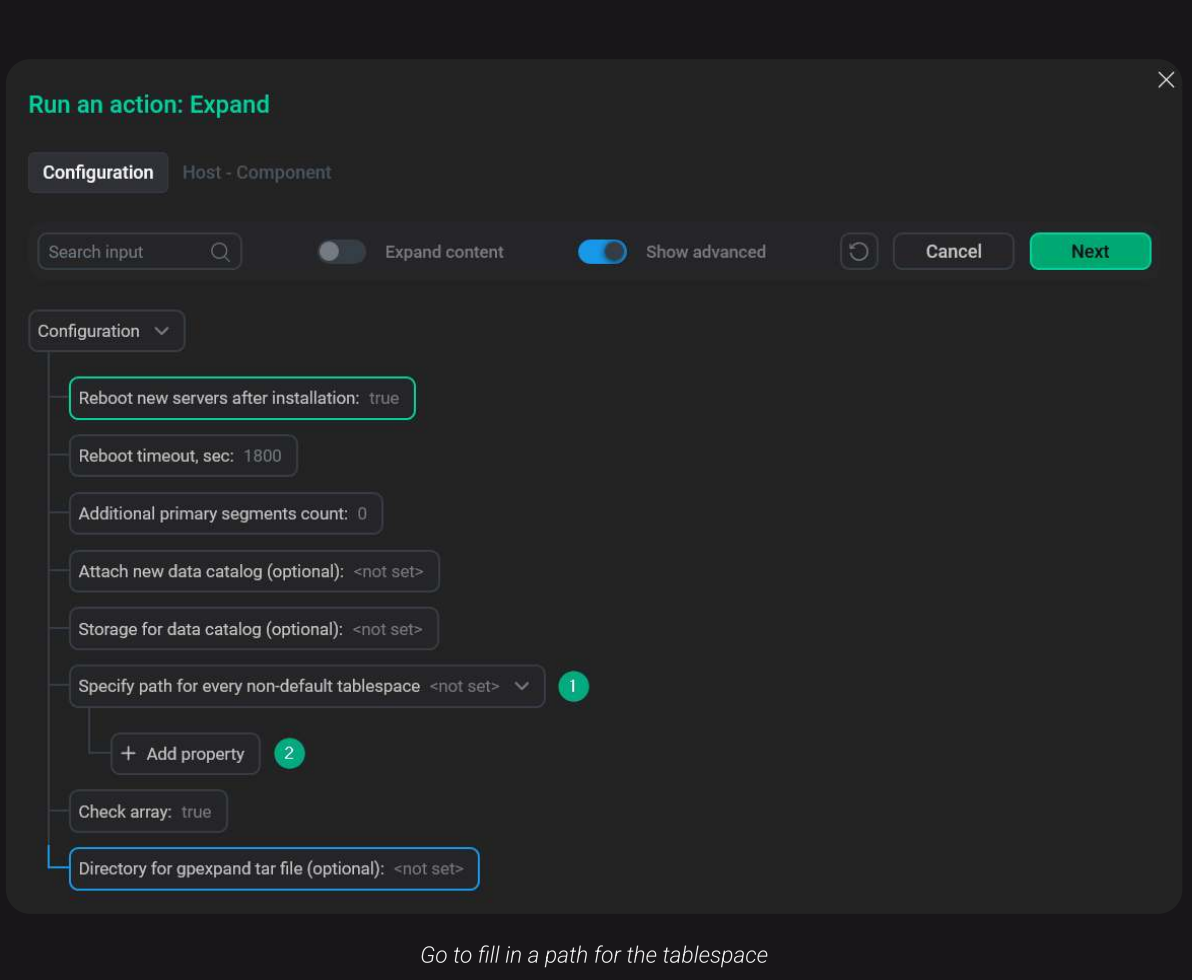
To edit any parameter (except the ***Specify path for every non-default tablespace*** field described below), click its current value. In the window that opens, enter a new value and click ***Apply***.



Reboot new servers after installation
Reboot new servers after installation
Cancel
Apply

Enter a new parameter value

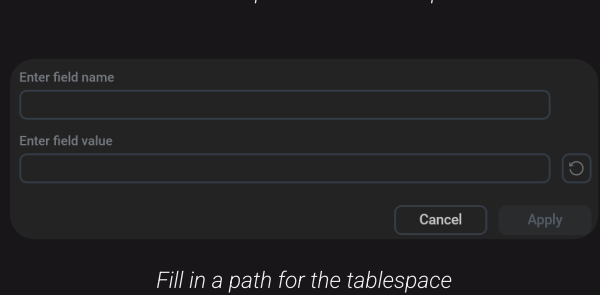
To edit the ***Specify path for every non-default tablespace*** parameter value (like any other parameter that allows adding several values), expand a node with the parameter name and click ***Add property***. In the dialog that opens, enter a tablespace name in the ***Enter field name*** field and a directory path in the ***Enter field value*** field. Click ***Apply*** to save changes.



Run an action: Expand
Configuration
Host - Component
Search input
Expand content
Show advanced
Cancel
Next
Configuration
1
Add property
2

Reboot new servers after installation: true
Reboot timeout, sec: 1800
Additional primary segments count: 0
Attach new data catalog (optional): <not set>
Storage for data catalog (optional): <not set>
Specify path for every non-default tablespace <not set>
Check array: true
Directory for gpexpand tar file (optional): <not set>

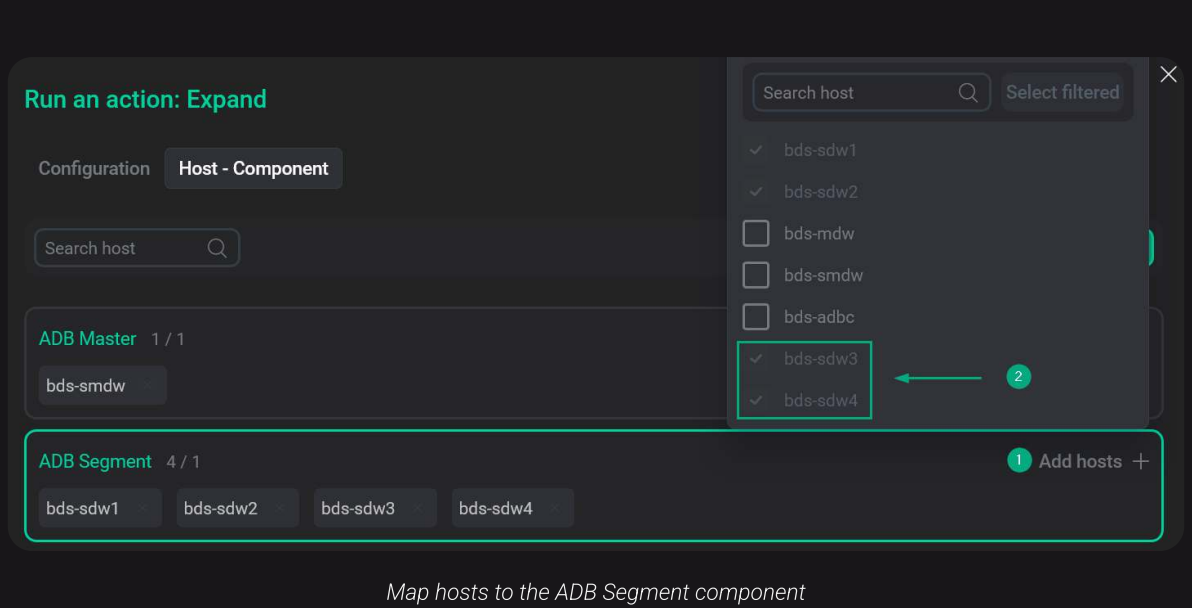
Go to fill in a path for the tablespace



Enter field name
Enter field value
Cancel
Apply

Fill in a path for the tablespace

2. Click ***Next***.
3. On the ***Host - Component*** tab that opens, assign new segment hosts to the ***ADB Segment*** component. To do this, click ***Add hosts*** in the component section and select hosts in the dialog that opens.



Run an action: Expand
Configuration
Host - Component
Search host
Search host
Select filtered
bds-sdw1
bds-sdw2
bds-mdw
bds-smdw
bds-adbc
bds-sdw3
bds-sdw4
2
1
Add hosts

ADB Master 1 / 1: bds-smdw
ADB Segment 4 / 1: bds-sdw1, bds-sdw2, bds-sdw3, bds-sdw4

Map hosts to the ADB Segment component

> IMPORTANT
>
> If you use PXF or Chrony in the ADB cluster, you should also map their components (***PXF*** or ***NTP Slave***) to new segment hosts for correct cluster operation.

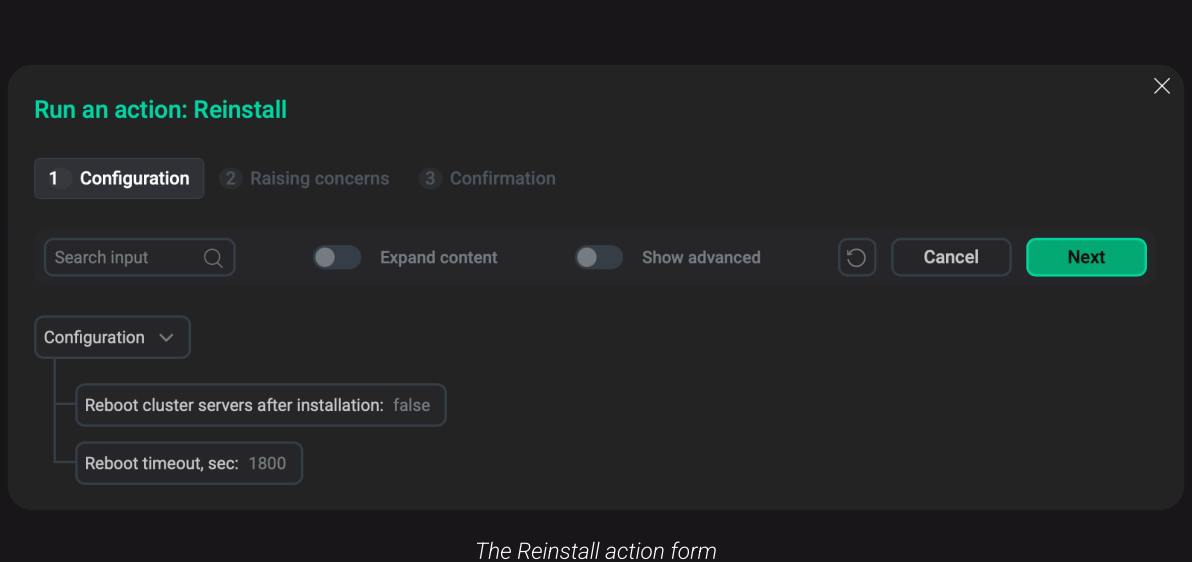
4. After the mapping of components is completed, click ***Run*** to launch the ***Expand*** action.
5. Confirm the action in the standard confirmation dialog by clicking ***Run***.

During the ***Expand*** action, the necessary packages are installed and configured on new hosts after some checks. Then, the `gpexpand` schema is created in the `postgres` database. Objects of this schema will be used to store metadata and statistics on the tables that require redistribution over segments after the cluster expansion. After expanding a cluster, run the ***Redistribute*** action, which becomes available in the ADB service if the ***Expand*** cluster action finished successfully. Redistribution is necessary because otherwise the tables will continue to operate within the same segment hosts without any performance improvement.

Reinstall

The ***Reinstall*** action reinstalls the ADB cluster. After you select the action, a dialog box opens in which you can fill in the following options:

- ***Reboot cluster servers after installation*** — a flag that indicates whether to reboot ADB hosts automatically after the ***Reinstall*** action is completed. If the flag is reset, you need to restart manually.
- ***Reboot timeout, sec*** — a timeout that is used to wait for the reboot of ADB hosts (in seconds).

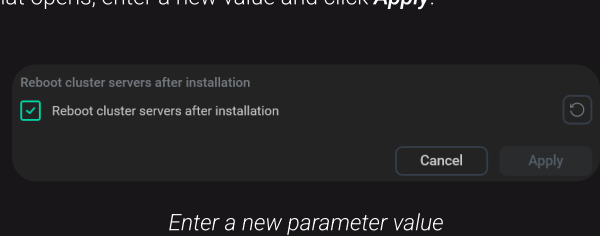


Run an action: Reinstall
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Reboot cluster servers after installation: false
Reboot timeout, sec: 1800

The Reinstall action form

To edit any parameter, click its current value. In the window that opens, enter a new value and click ***Apply***.



Reboot cluster servers after installation
Reboot cluster servers after installation
Cancel
Apply

Enter a new parameter value

To run the ***Reinstall*** action, click ***Run*** in the action form. Then, confirm the action in the standard confirmation dialog.

Reinstall statuschecker

The ***Reinstall statuschecker*** action reconfigures and restarts the statuschecker for all cluster services. Use this action when migrating a cluster to a new ADCM server.

After you select the action, the standard confirmation dialog opens. Click ***Run*** to confirm the action. No additional parameters are required.

Reconfigure Vault integration

The ***Reconfigure Vault integration*** action is available in the Enterprise Edition of ADB. It is used to apply changes to the Vault integration parameters. These parameters are available on the ***Configuration*** tab of the cluster page and allow you to store secrets of ADB services in HashiCorp Vault.

After you select the action, the standard confirmation dialog opens. Click ***Run*** to confirm the action. No additional parameters are required.

> IMPORTANT
>
> - Before editing Vault integration parameters, ensure that Vault is installed and configured.
> - Each time you edit and save parameters in the ***Vault integration*** section, run the cluster action Reconfigure Vault integration.
> - Currently, Vault can be used to store authentication parameters of ADB Control and ADBM.
> - If you need to store credentials for the ADB Control Query DB and Metrics DB in Vault, set up these databases as external.

Reconfigure parameter archiving

The ***Reconfigure parameter archiving*** action is used to apply changes to the Parameter archiving/Parameter archiving job setting parameters. These parameters are available on the ***Configuration*** tab of the cluster page and allow you to export ADB settings according to the specified schedule.

After you select the action, the standard confirmation dialog opens. Click ***Run*** to confirm the action. No additional parameters are required.

> IMPORTANT
>
> Each time you edit and save the ***Parameter archiving*** and ***Parameter archiving job setting*** values, run the ***Reconfigure parameter archiving*** action.

Check

The ***Check*** action verifies if all hosts, components, and services are configured according to the ADB cluster requirements (like the Precheck action does) and additionally checks whether the installed cluster components operate correctly.

After you select the action, the standard confirmation dialog opens. Click ***Run*** to confirm the action. No additional parameters are required.

Start

The ***Start*** action starts all services of the stopped ADB cluster.

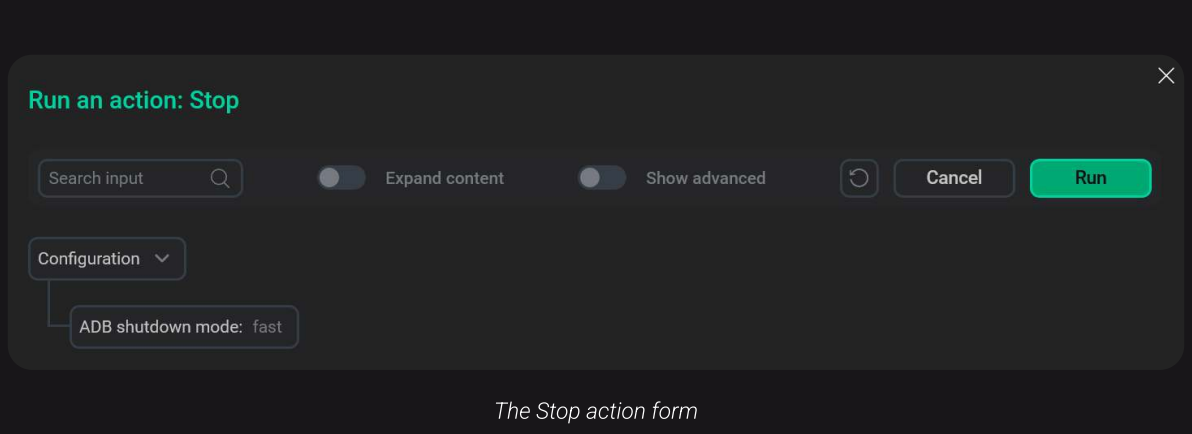
After you select the action, the standard confirmation dialog opens. Click ***Run*** to confirm the action. No additional parameters are required.

You can view the state of the services on the Services tab.

Stop

The ***Stop*** action stops all services of the running ADB cluster. After you select the action, a dialog box opens in which you can set the ***ADB shutdown mode***:

- `fast` — the ADB cluster will be stopped after all transactions are interrupted and rolled back, and all active connections are closed. This is the default mode.
- `smart` — the ADB cluster will be stopped only if there are no active client connections. Otherwise, the action fails with a warning.
- `immediate` — the ADB cluster will be stopped after all transactions are aborted and postgres processes are killed. This mode does not allow a database server to complete transaction processing and cleanup any temporary or in-process work files. It is not recommended to use the `immediate` mode since it can damage databases in some cases.

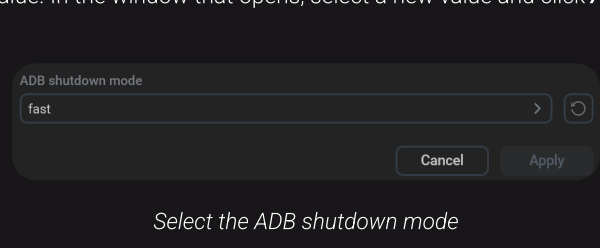


Run an action: Stop
Search input
Expand content
Show advanced
Cancel
Run
Configuration

ADB shutdown mode: fast

The Stop action form

To edit the ***ADB shutdown mode*** parameter, click its current value. In the window that opens, select a new value and click ***Apply***.



ADB shutdown mode
fast
Cancel
Apply

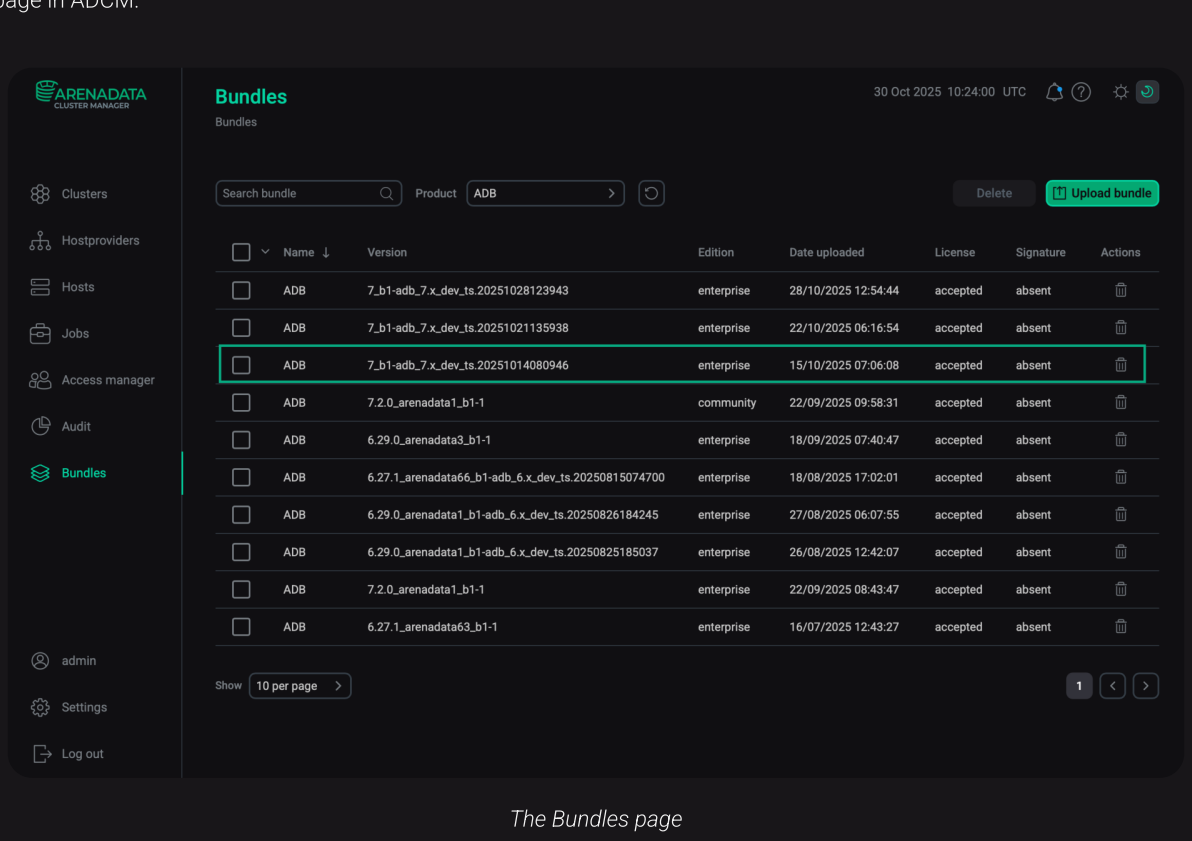
Select the ADB shutdown mode

To run the ***Stop*** action, click ***Run*** in the action form. Then, confirm the action in the standard confirmation dialog.

Upgrade

To upgrade an ADB cluster version, follow the steps:

1. Upload a new ADB cluster bundle (see the ***Download a cluster bundle*** and ***Upload a cluster bundle to ADCM*** steps in the Create a cluster article). As a result, the bundle should be displayed on the ***Bundles*** page in ADCM.

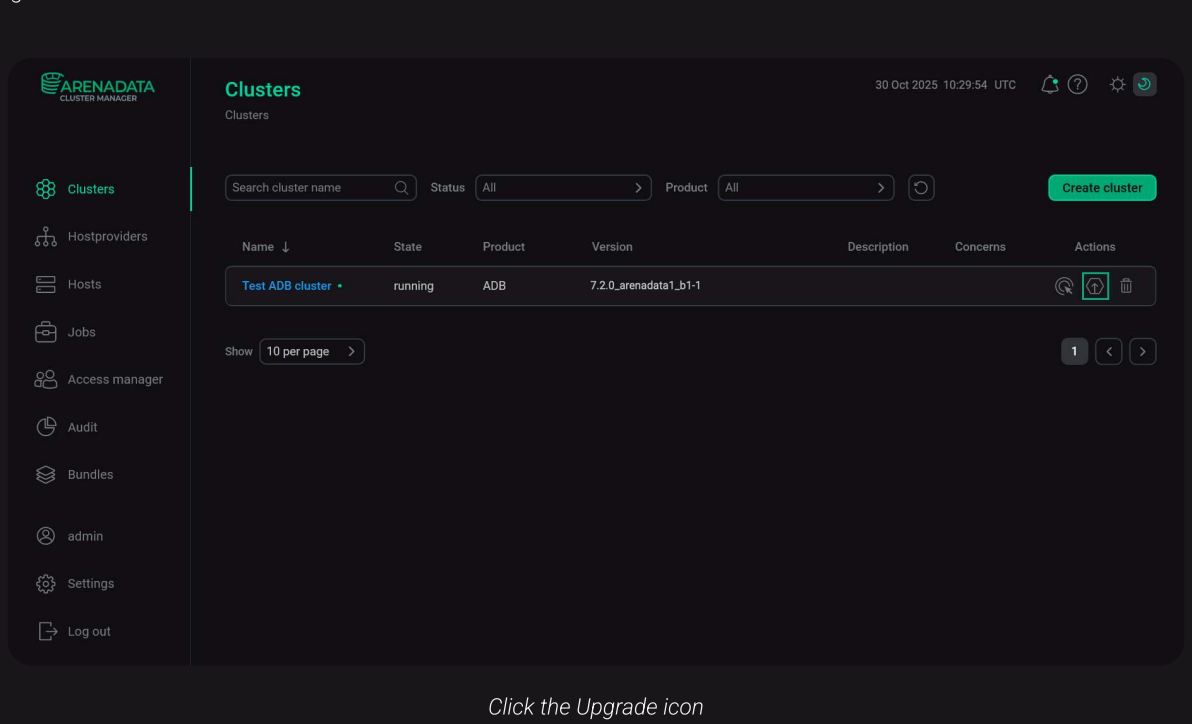


ARENADATA CLUSTER MANAGER
Bundles
30 Oct 2025 10:24:00 UTC
Search bundle
Product ADB
Delete
Upload bundle
Show 10 per page

Name	Version	Edition	Date uploaded	License	Signature	Actions
ADB	7_b1-adb_7.x_dev_ts.20251028123943	enterprise	28/10/2025 12:54:44	accepted	absent	
ADB	7_b1-adb_7.x_dev_ts.20251021135938	enterprise	22/10/2025 06:16:54	accepted	absent	
ADB	7_b1-adb_7.x_dev_ts.20251014080946	enterprise	15/10/2025 07:06:08	accepted	absent	
ADB	7.2.0_arenadata1_b1-1	community	22/09/2025 09:58:31	accepted	absent	
ADB	6.29.0_arenadata3_b1-1	enterprise	18/09/2025 07:40:47	accepted	absent	
ADB	6.27.1_arenadata66_b1-adb_6.x_dev_ts.20250815074700	enterprise	18/08/2025 17:02:01	accepted	absent	
ADB	6.29.0_arenadata1_b1-adb_6.x_dev_ts.20250826184245	enterprise	27/08/2025 06:07:55	accepted	absent	
ADB	6.29.0_arenadata1_b1-adb_6.x_dev_ts.20250825185037	enterprise	26/08/2025 12:42:07	accepted	absent	
ADB	7.2.0_arenadata1_b1-1	enterprise	22/09/2025 08:43:47	accepted	absent	
ADB	6.27.1_arenadata63_b1-1	enterprise	16/07/2025 12:43:27	accepted	absent	

The Bundles page

2. On the ***Clusters*** page, click the icon for the ADB cluster that you need to upgrade. That icon becomes active after successfully loading the bundle in the previous step. The bundle version should be higher than the current cluster version.

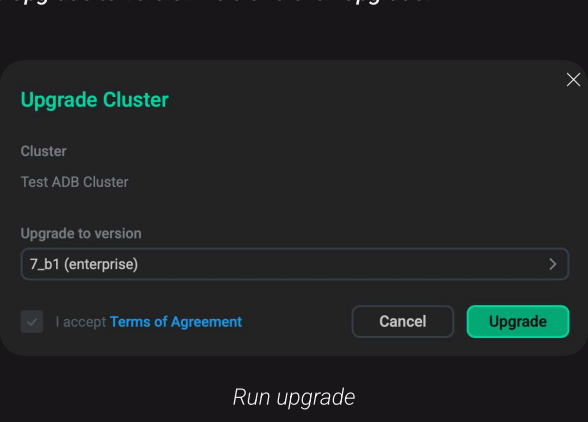


ARENADATA CLUSTER MANAGER
Clusters
30 Oct 2025 10:29:54 UTC
Search cluster name
Status All
Product All
Create cluster
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	running	ADB	7.2.0_arenadata1_b1-1			

Click the Upgrade icon

3. In the window that opens, select the bundle version in the ***Upgrade to version*** field and click ***Upgrade***.

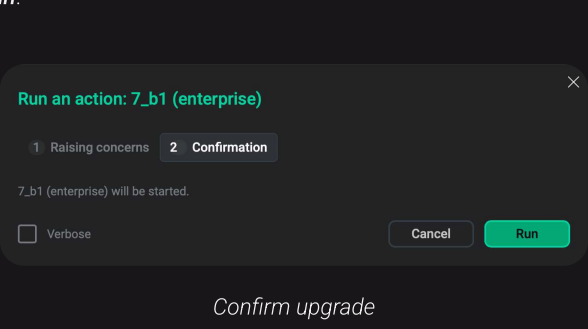


Upgrade Cluster
I accept Terms of Agreement
Cancel
Upgrade

Cluster: Test ADB Cluster
Upgrade to version: 7_b1 (enterprise)

Run upgrade

4. In the dialog that opens, confirm the action by clicking ***Run***.

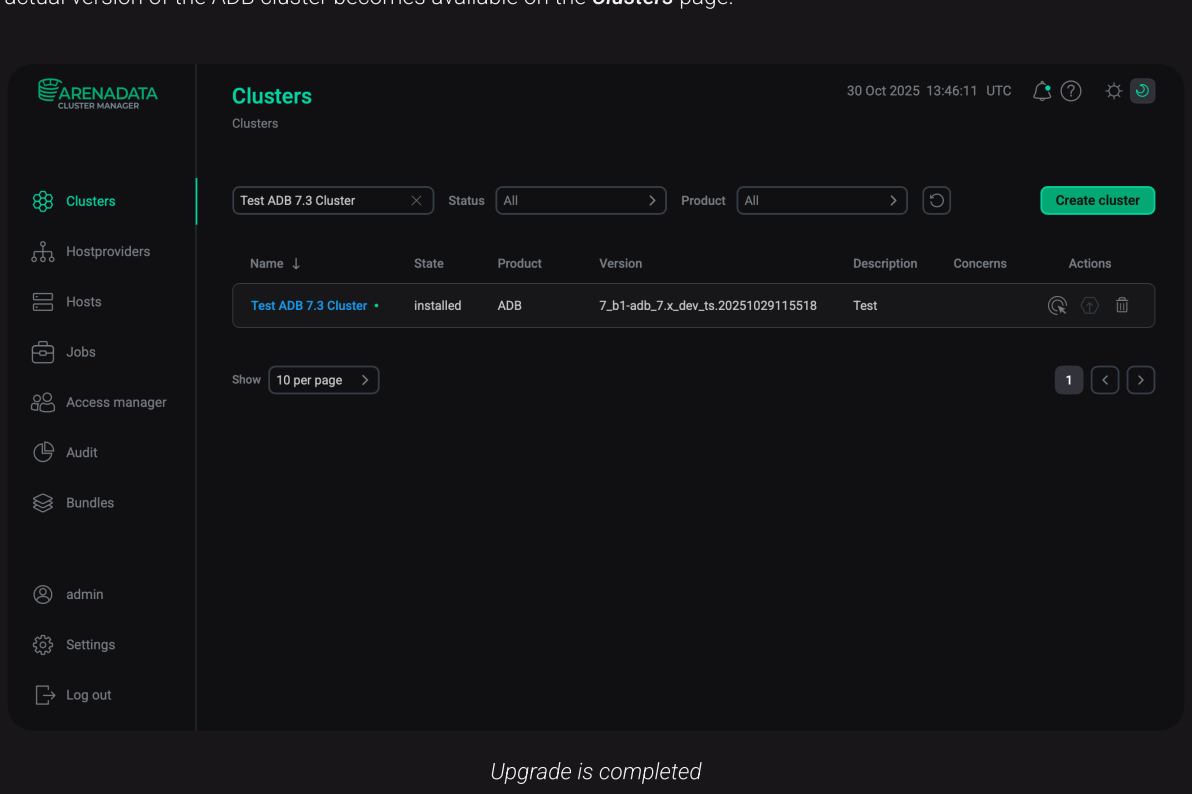


Run an action: 7_b1 (enterprise)
1 Raising concerns
2 Confirmation
Verbose
Cancel
Run

7_b1 (enterprise) will be started.

Confirm upgrade

As a successful result, the actual version of the ADB cluster becomes available on the ***Clusters*** page.



ARENADATA CLUSTER MANAGER
Clusters
30 Oct 2025 13:46:11 UTC
Test ADB 7.3 Cluster
Status All
Product All
Create cluster
Show 10 per page

Name	State	Product	Version	Description	Concerns	Actions
Test ADB 7.3 Cluster	installed	ADB	7_b1-adb_7.x_dev_ts.20251029115518	Test		

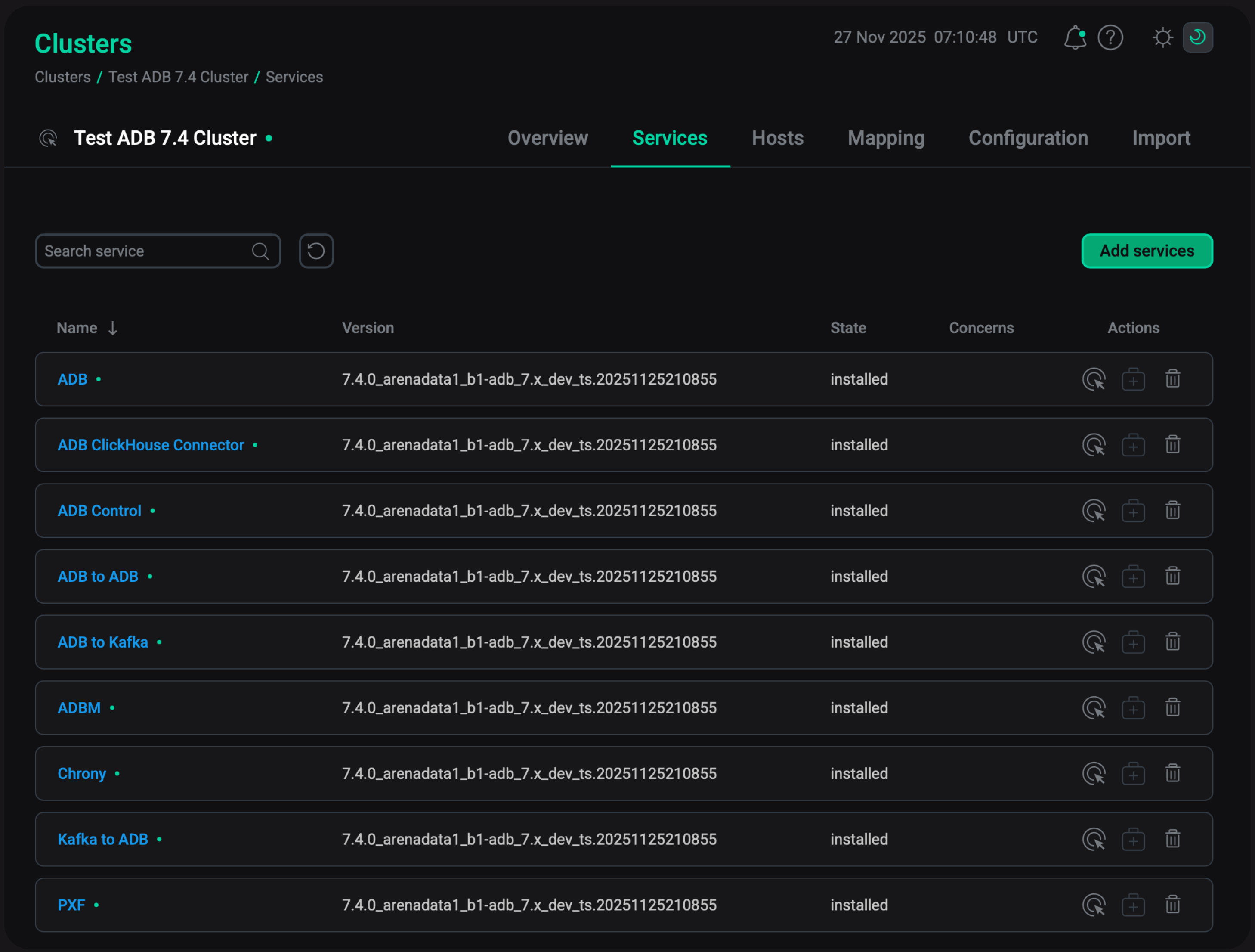
Upgrade is completed

Service actions

Anton Vasilev

To Table of Contents

ADCM UI provides actions to manage services. The actions are available on the ***Services*** tab. To open this tab, click an ADB cluster name on the ***Clusters*** page and switch to the ***Services*** tab.


Clusters
Clusters / Test ADB 7.4 Cluster / Services
27 Nov 2025 07:10:48 UTC
Test ADB 7.4 Cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add services

Name	Version	State	Concerns	Actions
ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
ADB ClickHouse Connector	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
ADB Control	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
ADB to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
ADB to Kafka	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
ADBM	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
Chrony	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
Kafka to ADB	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		
PXF	7.4.0_arenadata1_b1-adb_7.x_dev_ts.20251125210855	installed		

ADB cluster services in the ADCM interface

Refer to ADCM documentation for an overview of the ***Services*** tab and common actions.

Refer to the following articles that describe actions of ADB cluster services:

- ADB
- ADB Control
- ADBM
- PXF
- Connectors
- Chrony

ADB service actions

Anton Vasilev

Contents

To Table of Contents

- Create database
- Create role
- Enable auto core dump
- Disable auto core dump
- Enable mirroring
- Install diskquota
- Install MADlib
- Install PostGIS
- Reconfigure
- Redistribute
  - Rank tables for redistribution
  - Run redistribution
  - Monitor redistribution
- Reinstall ADB
- Run SQL
- Start
- Stop
- Manage tablespace

This article describes the actions available for the ADB service in the ADCM web interface.

Create database

Recreates the default database. The default name is `adb`.

Under normal circumstances, the action is not required because the database is created automatically during ADB installation. You can use this action though if you've accidentally deleted the default database.

After the action completes successfully, you can check that the database exists:

1. On the ADB coordinator host, log in as the `gpadmin` user:

```
$ sudo su - gpadmin
```

2. List all databases:

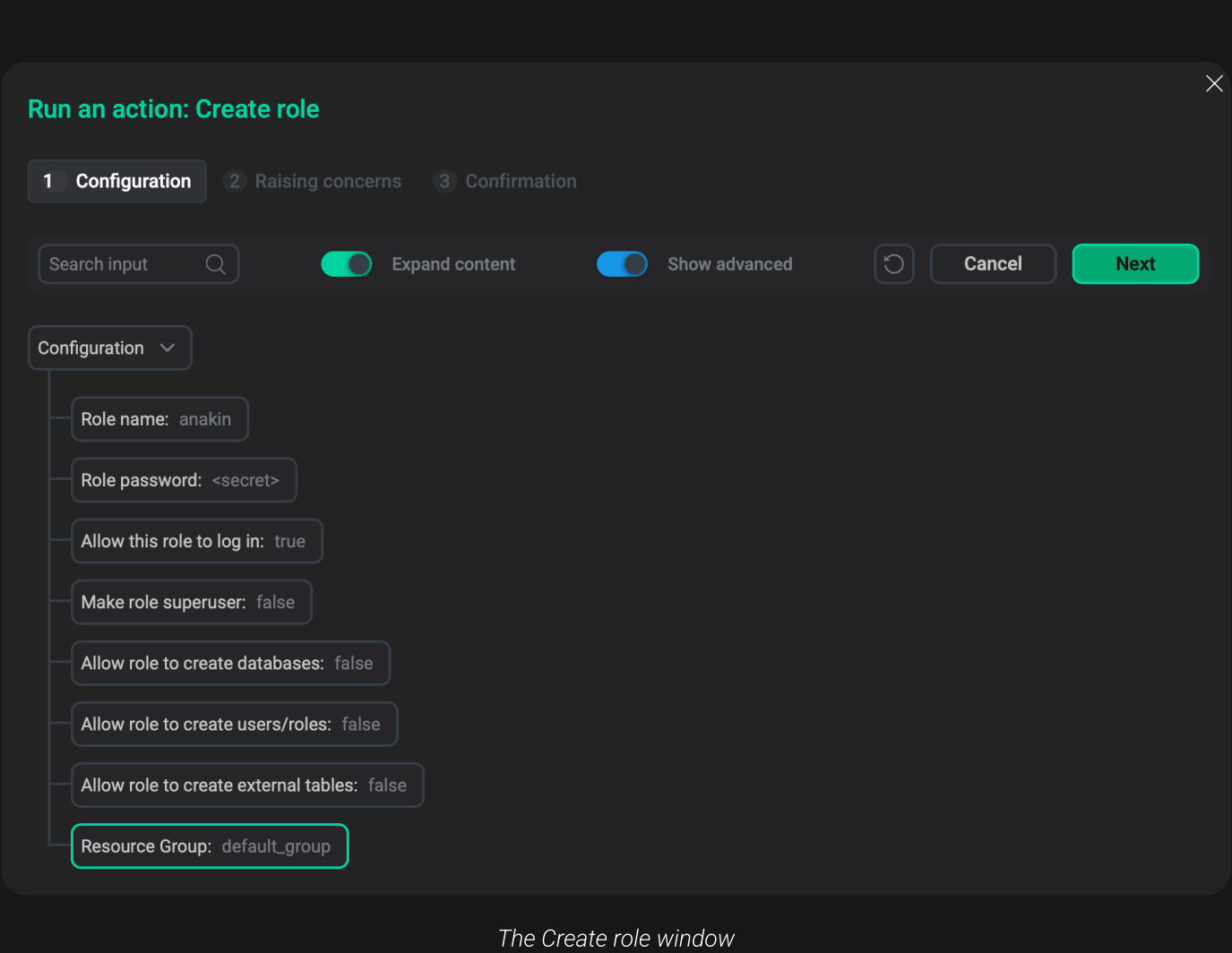
```
$ psql -l
```

The output should contain the `adb` database:

```
                              List of databases
   Name    |  Owner  | Encoding |  Collate   |   Ctype    |  Access privileges
-----------+---------+----------+------------+------------+---------------------
 adb       | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =Tc/gpadmin        +
           |         |          |            |            | gpadmin=CTc/gpadmin
 postgres  | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +
           |         |          |            |            | gpadmin=CTc/gpadmin
 template1 | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +
           |         |          |            |            | gpadmin=CTc/gpadmin
(5 rows)
```

Create role

Adds a new role to the database cluster.



Role name: anakin
Role password: <secret>
Allow this role to log in: true
Make role superuser: false
Allow role to create databases: false
Allow role to create users/roles: false
Allow role to create external tables: false
Resource Group: default_group

The Create role window

On the ***Configuration*** tab of the window that opens, configure the following parameters.

Parameter	Description	Default value
Role name	A unique name for the new role	—
Role password	The password of the role. Can't be left empty	—
Allow this role to log in	Determines whether a role is allowed to log in. When set to `true`, the role can be thought of as a user	true
Make role superuser	Determines whether a role is a superuser	false
Allow role to create databases	Determines whether the role can create new databases	false
Allow role to create users/roles	Determines whether the role can create new roles	false
Allow role to create external tables	Determines whether the role can create external tables	false
Resource Group	The name of an existing resource group to which a new role should be assigned. The value cannot be empty since each user should belong to a resource group	default_group

Enable auto core dump

Installs the Auto Core Dump service (`systemd-coredump`) on the coordinator, standby coordinator, and every segment host. The action uses the following values of the service configuration options:

- `Storage=external` — core dumps will be stored in */var/lib/systemd/coredump/*.
- `MaxUse=1024M` — remove old core dumps when their size exceeds 1 GB.

To check that the service was successfully installed and started, run the following command on the coordinator, standby coordinator, or any segment host:

```
$ systemctl status systemd-coredump.socket
```

The service should have the `active (listening)` status:

```
systemd-coredump.socket - Process Core Dump Socket
     Loaded: loaded (/lib/systemd/system/systemd-coredump.socket; static)
     Active: active (listening) since Thu 2025-09-04 10:24:43 UTC; 21min ago
       Docs: man:systemd-coredump(8)
     Listen: /run/systemd/coredump (SequentialPacket)
   Accepted: 0; Connected: 0;
     CGroup: /system.slice/systemd-coredump.socket
```

Disable auto core dump

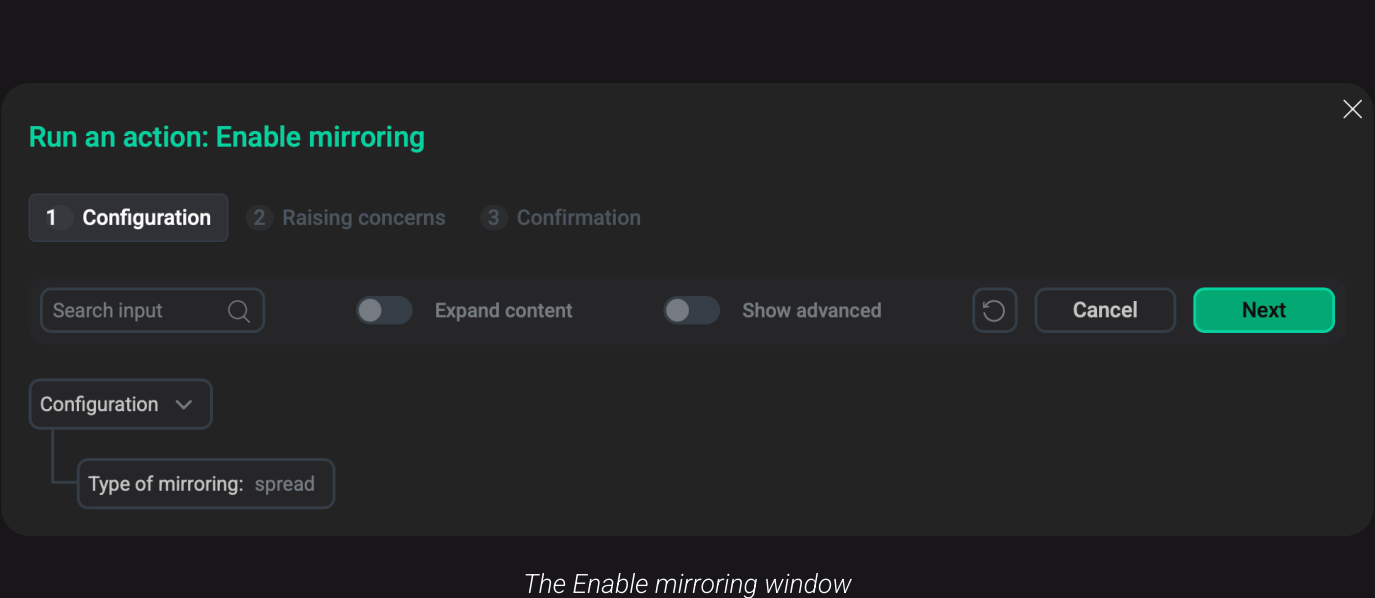
Removes the Auto Core Dump service (`systemd-coredump`) and its configuration files.

Enable mirroring

Enables synchronous replication for primary data segments by adding mirrors to the cluster.

On the ***Configuration*** tab of the window that opens, in the ***Type of mirroring*** list, select the mirroring type: `spread` or `group`.

Spread mirroring is used by default. If you select spread mirroring, the action verifies that the number of segment hosts in your cluster is greater than the number of segments configured per host and that the number of segments per host can be divided by the number of the data directories on the host without remainder. If these conditions are not met, the action will fail.



Type of mirroring: spread

The Enable mirroring window

Install diskquota

Installs the Diskquota ADB extension. This extension allows users to limit the disk space used by schemas and roles.

The action creates the `diskquota` database and the `diskquota` schema and registers `diskquota` as a database extension.

You can check the `diskquota` extension by running:

```
SELECT * FROM pg_extension WHERE extname = 'diskquota';
```

The output should be similar to:

```
  oid  |  extname  | extowner | extnamespace | extrelocatable | extversion |   extconfig   | extcondition
-------+-----------+----------+--------------+----------------+------------+---------------+--------------
 20312 | diskquota |       10 |         2200 | t              | 2.3        | {20314,20321} | {"",""}
(1 row)
```

After `diskquota` is installed, the ***Uninstall diskquota*** and ***Reinstall diskquota*** actions become available.

Install MADlib

Installs the MADlib library.

The action creates the `madlib` extension and schema.

Check the installed version by running:

```
SELECT madlib.version();
```

The output should be similar to:

```
         version
-------------------------------------------------------------------------------------------------------
 MADlib version: 2.1.0, git revision: 6.22.1_arenadata41-244-g5ab0b4c4, cmake configuration time: Tue Oct 14 14:
46:34 UTC 2025, build type: RelWithDebInfo, build system: Linux-4.15.0-175-generic, C compiler: gcc 11, C++ comp
iler: g++ 11
(1 row)
```

After MADlib is installed, the ***Uninstall MADlib*** and ***Reinstall MADlib*** actions become available.

Install PostGIS

Installs the PostGIS extension.

To check the version of installed PostGIS, run the following command in your database:

```
SELECT PostGIS_Lib_Version();
```

The output should be similar to the following:

```
 postgis_lib_version
---------------------
 3.3.2
(1 row)
```

After PostGIS is installed, the ***Uninstall PostGIS*** and ***Reinstall PostGIS*** actions become available.

Reconfigure

Updates the service configuration according to the configuration settings.

The action restarts the cluster (using `gpstop -arM fast`) if any of the following configuration parameters are changed:

- ***Interconnect properties → Interconnect type***
- Any option under ***SSL***

Otherwise, the action reloads the configuration (using `gpstop -au`).

Redistribute

The ***Redistribute*** action redistributes tables to balance existing data over the newly expanded ADB cluster. The action is available only after the Expand cluster action has completed successfully. The action uses the Greenplum utility gpexpand.

It is recommended to rank the database tables before starting the action, i.e. specify the order in which tables should be redistributed.

Rank tables for redistribution

A priority of the table processing during redistribution is determined by the `rank` column value in the `gpexpand.status_detail` table. Tables with the lowest rank are redistributed first. To rank tables, follow the steps:

1. Connect to the `postgres` database of the ADB cluster under the `gpadmin` user (e.g. via `psql`).
2. For all rows of the `gpexpand.status_detail` table, set the initial value for the `rank` column that will be considered as maximum (and will match the lowest processing priority), e.g. `100`:

```
UPDATE gpexpand.status_detail SET rank = 100;
```

3. Decrease the `rank` value for those tables that should be redistributed first. For example, as a result of the following queries, the ***Redistribute*** action will process the `public.test` table first, then the `public.test2` table, and then other tables in `gpexpand.status_detail`.

```
UPDATE gpexpand.status_detail SET rank = 10 WHERE fq_name = 'public.test';
UPDATE gpexpand.status_detail SET rank = 20 WHERE fq_name = 'public.test2';
```

TIP

- Use a fully qualified table name (including a schema name) as the `fq_name` column value.
- It is recommended to use the step `10` when ranking tables (instead of `1`) — to provide the ability to quickly increase a table priority without affecting other tables.
- Assign the highest priority (the smallest `rank` value) to those tables that are used more often than others.
- To exclude any table from the redistribution process, you can remove it from the `gpexpand.status_detail` table.

Run redistribution

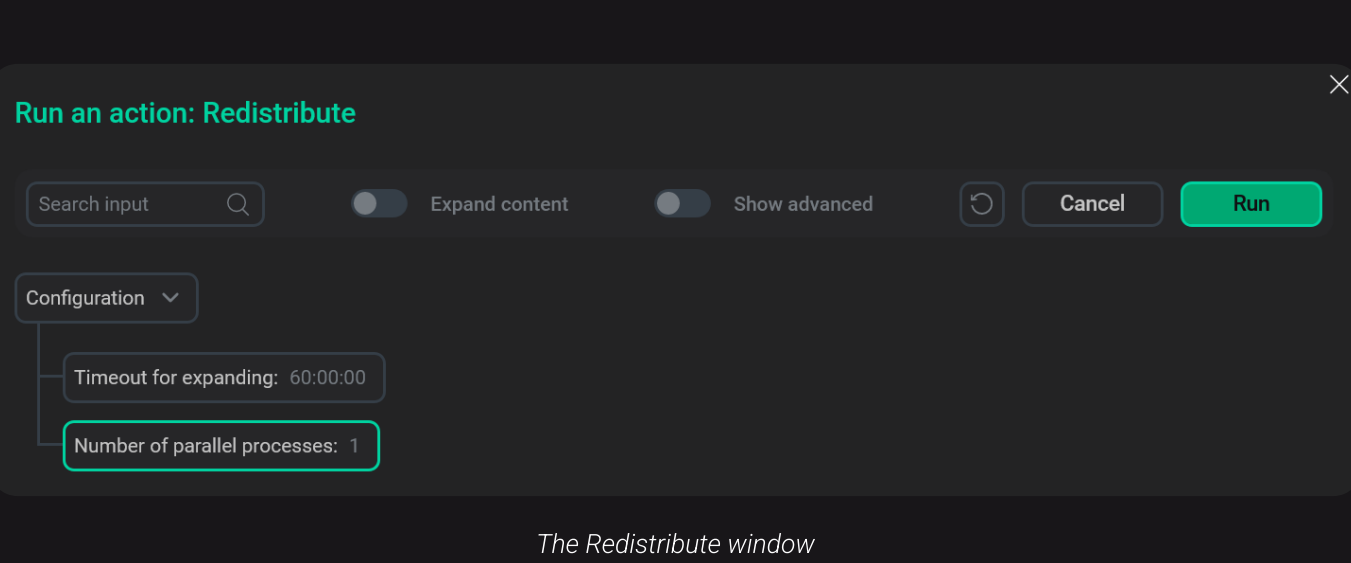
To run data redistribution, select the ***Redistribute*** action. It is recommended to perform redistribution during low-use hours when table locks do not have a significant impact on the cluster operations.

IMPORTANT

- To perform redistribution, segment hosts should have enough disk space to temporarily store a copy of the largest table/partition.
- During redistribution, each table or partition is locked (with the `ACCESS EXCLUSIVE` lock type) and becomes unavailable for read and write operations. Once its redistribution is completed, normal operations for this particular table resume.
- New tables and partitions are distributed across all segments like under normal operating conditions.

After you select the ***Redistribute*** action, a dialog box opens in which you can set the following fields:

- ***Timeout for expanding*** — the maximum redistribution duration in hours, minutes, and seconds.
- ***Number of parallel processes*** — defines the number of tables to redistribute simultaneously (which maps to the `-n` option value for the `gpexpand` utility). Valid values are 1 — 96. Each table redistribution process requires two database connections: one to alter the table and another to update the table status in the `gpexpand` schema. Before increasing ***Number of parallel processes***, check the current value of the server configuration parameter `max_connections` to ensure that the maximum connection limit is not exceeded.



Timeout for expanding: 60:00:00
Number of parallel processes: 1

The Redistribute window

To run the ***Redistribute*** action, click ***Run*** in the dialog box. If redistribution of all tables is completed within the time specified in the ***Timeout for expanding*** field, the `gpexpand` schema is removed from the `postgres` database. Otherwise, you can retry redistribution by running the ***Redistribute*** action again.

Monitor redistribution

To track the redistribution progress, you can run SQL queries against the following objects of the `gpexpand` schema in the `postgres` database:

- `gpexpand.status` — the table that stores a history of redistribution status changes.

Structure

Column	Description
status	A status of the ADB redistribution. Possible values: - `SETUP` — preparations for the cluster expansion are started during the ***Expand*** action. - `SETUP DONE` — preparations for the cluster expansion are completed during the ***Expand*** action. - `EXPANSION STARTED` — table redistribution is started during the ***Redistribute*** action. - `EXPANSION STOPPED` — table redistribution is stopped. The additional launch of the ***Redistribute*** action is required. - `COMPLETED` — redistribution of all tables is successfully completed after one or more launches of the ***Redistribute*** action.
updated	A timestamp of the status change

Query example

```
SELECT * FROM gpexpand.status;
```

Result:

```
      status       |          updated
-------------------+----------------------------
 EXPANSION STARTED | 2024-02-28 08:02:31.026412
 SETUP DONE        | 2024-02-27 17:40:55.160054
 SETUP             | 2024-02-27 17:40:50.830922
(3 rows)
```

- `gpexpand.expansion_progress` — a view that shows the current redistribution status: how many tables/bytes have been successfully redistributed and how many are left. Also, the view provides calculations of the estimated redistribution rate and estimated time to completion. Calculations restart each time you run the ***Redistribute*** action — after the first table redistribution is completed. Statistics on tables/bytes are refreshed as they are redistributed.

NOTE

In ADB, the `gpexpand` utility is used with the `--simple-progress` option to improve performance by reducing the amount of progress information written to the `gpexpand` tables. Due to this, only the `Tables Expanded` and `Tables Left` metrics have values in the `gpexpand.expansion_progress` table.

Structure

Column	Description
name	A metric name: - `Bytes Left` — amount of data that is not yet redistributed (in bytes). - `Bytes Done` — amount of data that is successfully redistributed (in bytes). - `Estimated Expansion Rate` — the estimated rate of data redistribution (with a measurement unit). - `Estimated Time to Completion` — the estimated time required for redistribution completion (in hours, minutes, and seconds). - `Tables Expanded` — a number of tables that are successfully redistributed. - `Tables Left` — a number of tables that are not yet redistributed.
value	A value of the metric that is specified in the `name` column

Query example

```
SELECT * FROM gpexpand.expansion_progress;
```

Result:

```
      name       | value
-----------------+-------
 Tables Expanded | 90
 Tables Left     | 7
(2 rows)
```

- `gpexpand.status_detail` — the table that stores the current redistribution status for each table/partition.

Structure

Column	Description
table_oid	A table object identifier (OID)
dbname	A name of the database to which the table belongs
fq_name	A fully qualified name of the table (with a schema name)
root_partition_oid	For a partitioned table, the OID of the root partition. Otherwise, `None`
rank	A rank that determines the priority of table processing during redistribution. Tables with the lowest rank are processed first
external_writable	Indicates whether the table is `EXTERNAL WRITABLE` (such tables require a different syntax when using `gpexpand`)
status	The current status of the table redistribution: - `NOT STARTED`; - `IN PROGRESS`; - `COMPLETED`; - `NO LONGER EXISTS` — the table does not exist anymore.
expansion_started	A timestamp of the table redistribution start. This column is only populated after the table is successfully redistributed
expansion_finished	A timestamp of the table redistribution completion
source_bytes	The disk size associated with the source table (in bytes). Due to the possible data bloat in heap tables and differing numbers of segments after expansion, it is not guaranteed that the final number of bytes will be equal to the source number. Values of the `source_bytes` column are used to estimate the redistribution duration. In ADB, the `source_bytes` value is equal to `0` for all tables due to using the `--simple-progress` option

Query example

```
SELECT dbname, fq_name, status, expansion_started, expansion_finished, source_bytes FROM gpexpand.status_detail;
```

The output fragment:

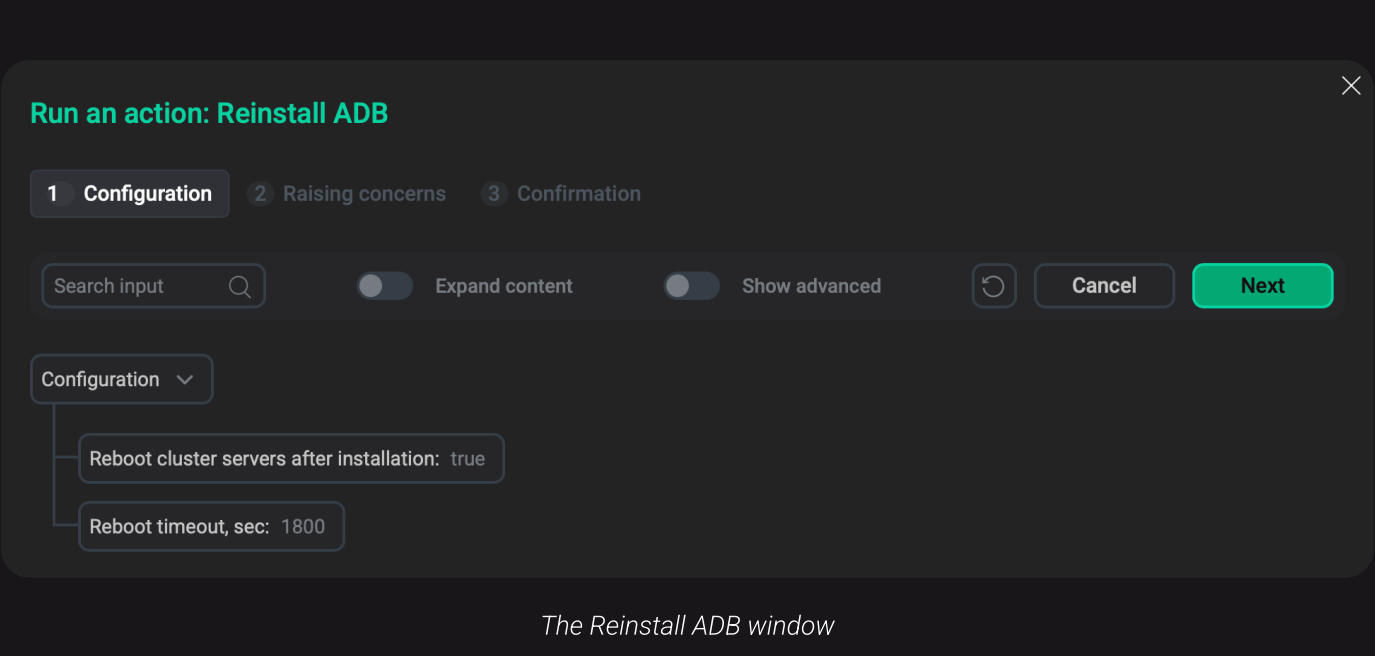
```
  dbname   |                        fq_name                        |   status    |     expansion_started      |
expansion_finished     | source_bytes
-----------+-------------------------------------------------------+-------------+----------------------------+------------
----------------+--------------
 adb       | kadb.offsets                                          | NOT STARTED |                            |
|            0
 adb       | arenadata_toolkit.db_files_history_1_prt_p202310      | NOT STARTED |                            |
|            0
 gpperfmon | public.database_history_1_prt_r1013160842             | NOT STARTED |                            |
|            0
 gpperfmon | public.log_alert_history_1_prt_r598663655             | NOT STARTED |                            |
|            0
 gpperfmon | public.log_alert_history_1_prt_r1113542580            | NOT STARTED |                            |
|            0
 adb       | public.kafka_ssl                                      | NOT STARTED |                            |
|            0
 adb       | public.adb_to_kafka_table3                            | NOT STARTED |                            |
|            0
 adb       | arenadata_toolkit.db_files_current                    | NOT STARTED |                            |
|            0
 adb       | arenadata_toolkit.db_files_history_1_prt_p202312      | NOT STARTED |                            |
|            0
 gpperfmon | public.database_history_1_prt_r2098142994             | NOT STARTED |                            |
|            0
 gpperfmon | public.network_interface_history_1_prt_1              | NOT STARTED |                            |
|            0
 adb       | public.ext_adb_to_kafka_sasl_gssapi                   | NOT STARTED |                            |
|            0
 adb       | public.adb_to_kafka_table7                            | NOT STARTED |                            |
|            0
 adb       | arenadata_toolkit.db_files_history_1_prt_default_part | NOT STARTED |                            |
|            0
 gpperfmon | public.queries_history_1_prt_r81170841                | NOT STARTED |                            |
|            0
 gpperfmon | public.queries_history_1_prt_r419258856               | NOT STARTED |                            |
|            0
 gpperfmon | public.diskspace_history_1_prt_r526486693             | NOT STARTED |                            |
|            0
 adb       | public.test2                                          | COMPLETED   | 2024-02-28 08:02:37.00728  | 2024-02-28
08:02:38.048592 |            0
 gpperfmon | public.system_history_1_prt_r1569528921               | COMPLETED   | 2024-02-28 08:02:41.066986 | 2024-02-28
08:02:44.182122 |            0
 gpperfmon | public.database_history_1_prt_r932188937              | COMPLETED   | 2024-02-28 08:02:44.305985 | 2024-02-28
08:02:44.697928 |            0
 gpperfmon | public.segment_history_1_prt_1                        | COMPLETED   | 2024-02-28 08:02:44.841786 | 2024-02-28
08:02:45.426392 |            0
 diskquota | arenadata_toolkit.db_files_history_1_prt_default_part | COMPLETED   | 2024-02-28 08:02:46.096083 | 2024-02-28
08:02:48.04711  |            0
 adb       | diskquota.target                                      | COMPLETED   | 2024-02-28 08:02:48.166249 | 2024-02-28
08:02:48.602075 |            0
 adb       | arenadata_toolkit.db_files_history_1_prt_p202402      | COMPLETED   | 2024-02-28 08:02:48.73606  | 2024-02-28
08:02:49.505108 |            0
```

Reinstall ADB

Reinstalls the ADB service.

On the ***Configuration*** tab of the window that opens, configure the following parameters:

- ***Reboot cluster servers after installation*** — a flag that indicates whether to reboot ADB hosts automatically after the ***Reinstall ADB*** action is completed. The default value is `false` meaning you need to restart hosts manually.
- ***Reboot timeout, sec*** — a timeout that is used to wait for the reboot of ADB hosts (in seconds). The default value is `1800`. The value must be at least `600`.



Reboot cluster servers after installation: true
Reboot timeout, sec: 1800

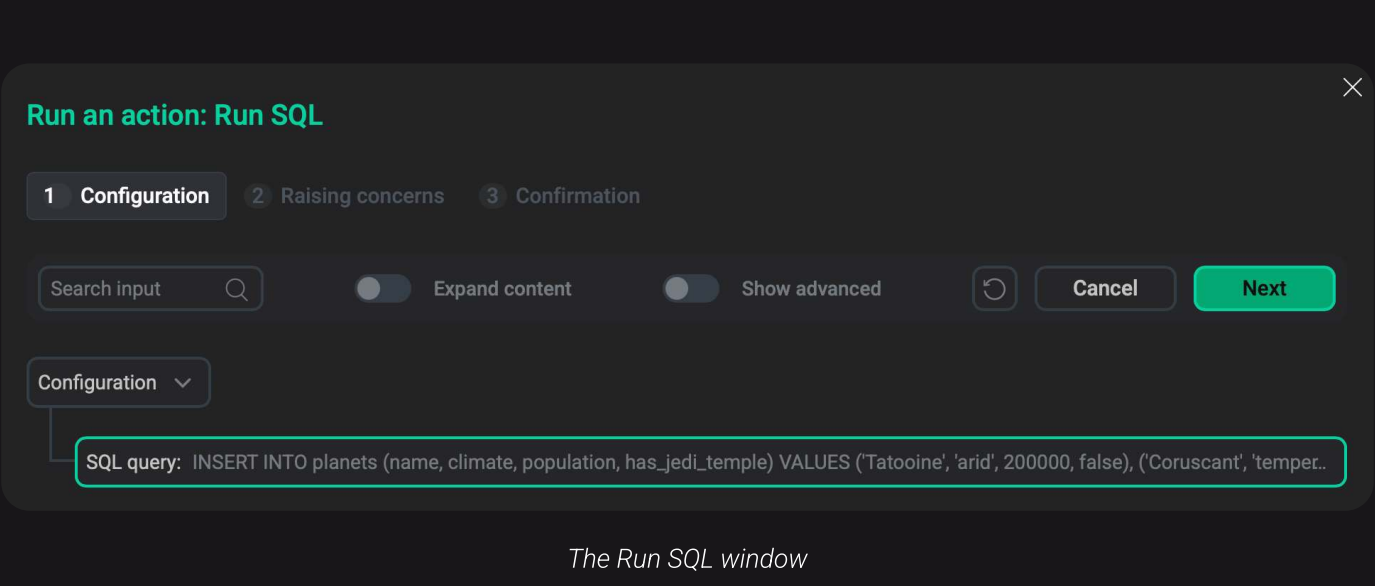
The Reinstall ADB window

Run SQL

Runs any custom SQL command on behalf of the `gpadmin` user.

The command is executed in the `adb` database unless another database is specified as the default database for client connections.

In the window that opens, in the ***SQL query*** field, enter your command.



SQL query: INSERT INTO planets (name, climate, population, has_jedi_temple) VALUES ('Tatooine', 'arid', 200000, false), ('Coruscant', 'temper...

The Run SQL window

To view the action result, on the ***Jobs*** page click the action name and open the ***Ansible [check]*** tab — the result will be in the ***SQL query*** line.

Start

Starts the ADB service.

Stop

Stops the ADB service. After you select the action, a dialog box opens in which you can set the ***ADB shutdown mode***:

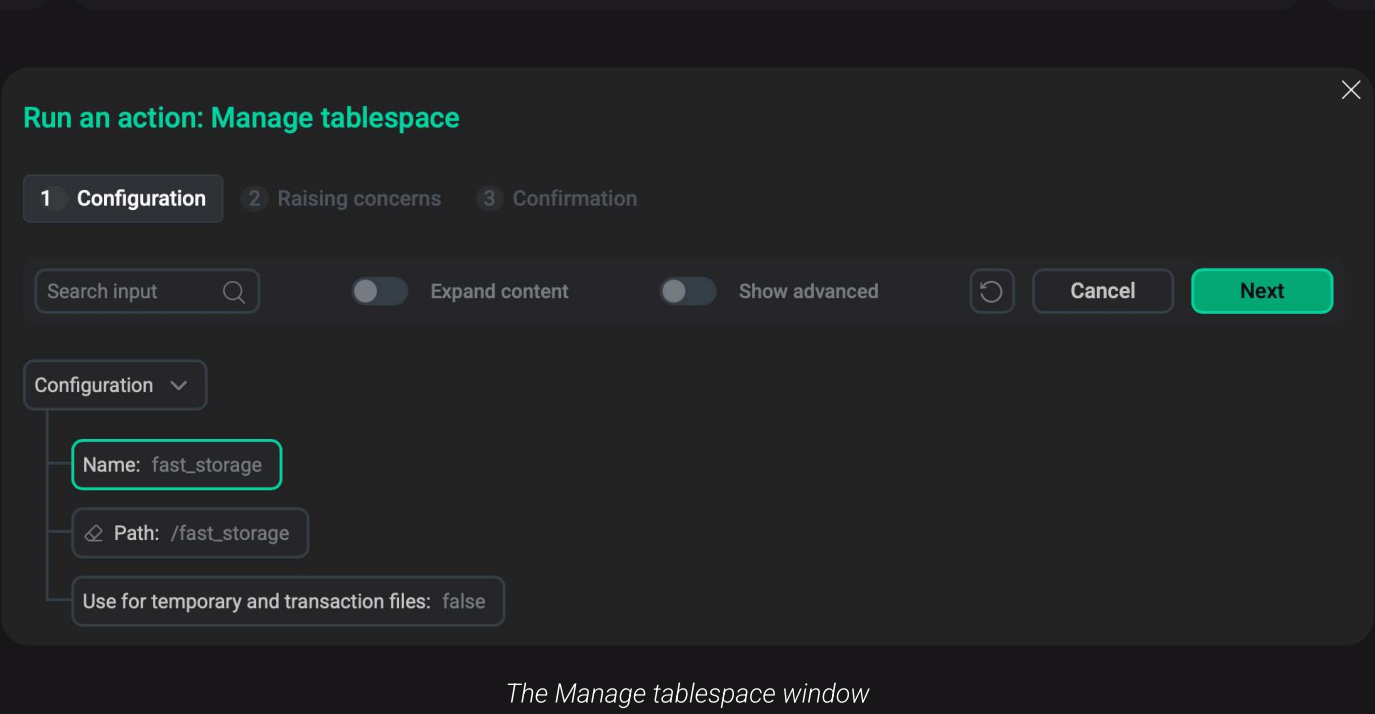
- `fast` — the ADB service will be stopped after all transactions are interrupted and rolled back, and all active connections are closed. This is the default mode.
- `smart` — the ADB service will be stopped only if there are no active client connections. Otherwise, the action fails with a warning.
- `immediate` — the ADB service will be stopped after all transactions are aborted and PostgreSQL processes are killed. This mode does not allow a database server to complete transaction processing and clean up any temporary or in-process work files. It is not recommended to use the `immediate` mode since it can damage databases in some cases.

Manage tablespace

Creates a tablespace or modifies existing tablespaces.

Parameter	Description	Default value
Name	A unique name for the new tablespace. The name cannot start with the `pg_` prefix. Alternatively, you can specify the name of an existing tablespace to modify its parameters	pg_default
Path	The path to the directory that will be used for the tablespace. The parameter is required if you create a new tablespace. The directory must exist on the coordinator host and on every segment host; the action does not create it	—

Use for temporary and transaction files	Use the tablespace to store temporary objects. The action adds the tablespace to the `temp_tablespaces` configuration parameter.
	To check tablespaces currently used to store temporary objects, you can run the following command:
	`$ gpconfig -s temp_tablespaces`
	The output should be similar to:
	`Values on all segments are consistent`
	`GUC               : temp_tablespaces`
	`Coordinator value: fast_storage`
	`Segment     value: fast_storage`
		false



Name: fast_storage
Path: /fast_storage
Use for temporary and transaction files: false

The Manage tablespace window

ADB Control service actions

Anton Vasilev

Contents

To Table of Contents

Install
Reinstall
Reconfigure & Restart
Start
Stop
Uninstall

This article describes the actions available for the ADB Control service in the ADCM web interface.

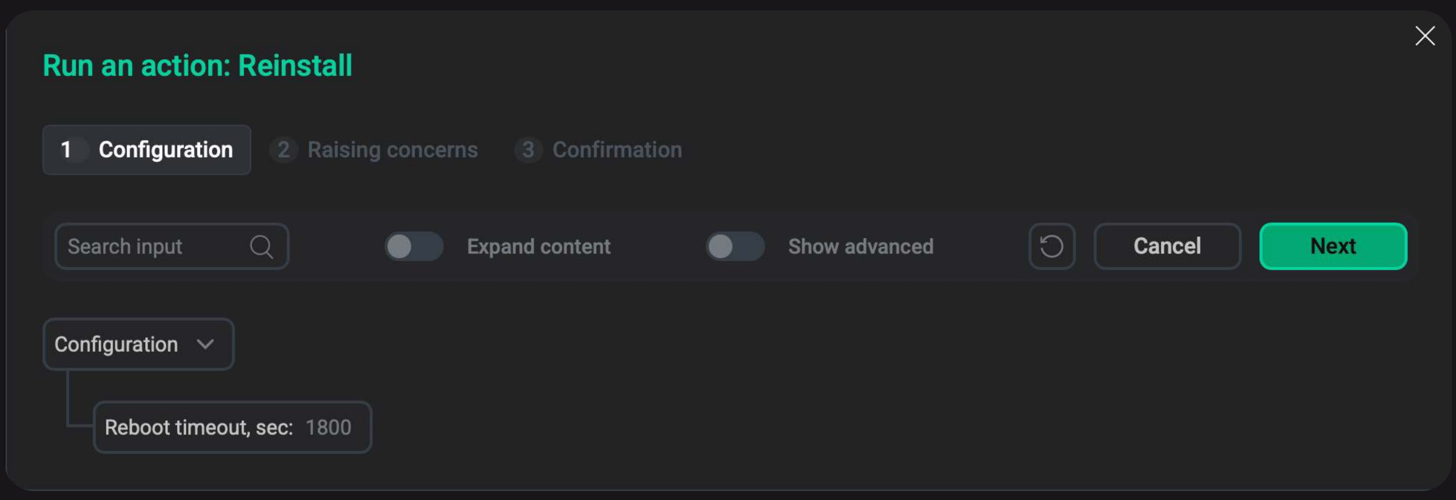
Install

Installs the ADB Control service. This action is available only if a service has been added but not yet installed.

Reinstall

Reinstalls the ADB Control service.

On the ***Configuration*** tab of the window that opens, in the ***Reboot timeout, sec*** field, enter the timeout to wait for hosts to restart (in seconds). The default value is `1800`. The value must be at least `600`.


Run an action: Reinstall
1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Reboot timeout, sec: 1800

The Reinstall window

Reconfigure & Restart

Updates the service configuration according to the configuration settings and restart the service.

In the window that opens, use the ***Unlock ADB Control db*** flag to determine what to do with database locks that may occur if a migration failed. When the flag is set to `true`, ADB Control releases the locks and makes its database available for migrations. The default value is `false`: preserve locks to avoid potentially conflicting migrations.

Start

Starts the ADB Control service.

Stop

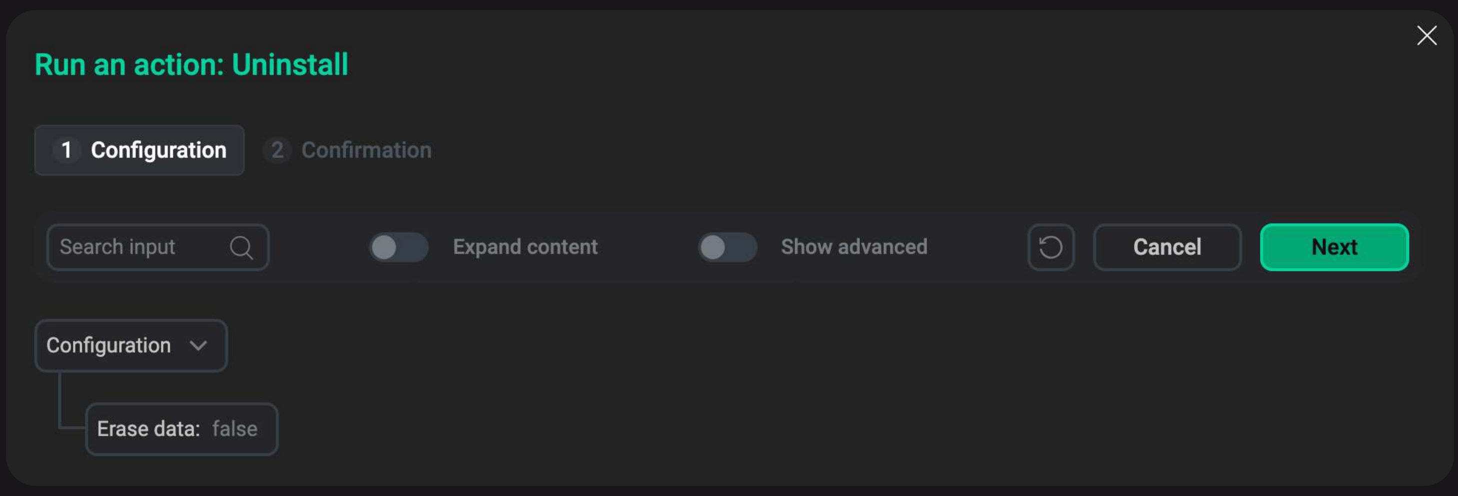
Stops the ADB Control service.

Uninstall

Uninstalls the ADB Control service. In the window that opens, use the ***Erase data*** flag to select whether you need to delete files associated with a service.

NOTE

The ADBM service depends on ADB Control. If you want to uninstall ADB Control, you have to uninstall ADBM first.


Run an action: Uninstall
1 Configuration
2 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

Erase data: false

The Uninstall window

ADBM service actions

Anton Vasilev

Contents

To Table of Contents

Install

Reinstall

Reconfigure & Restart

Start

Stop

Uninstall

This article describes the actions available for the Arenadata DB Backup Manager (ADBM) service in the ADCM web interface.

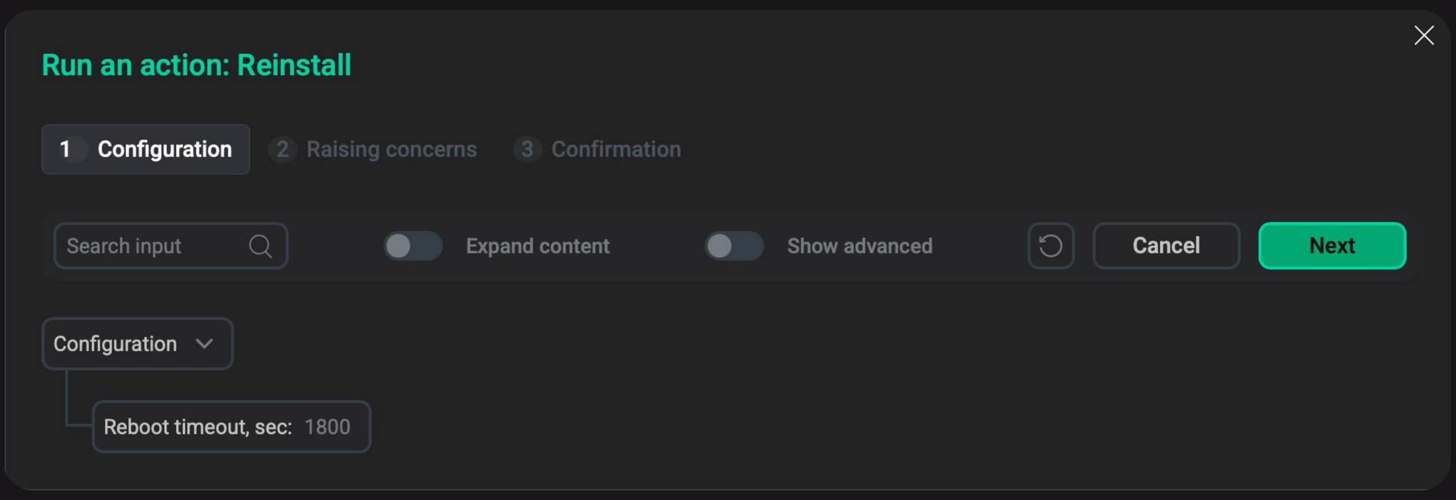
Install

Installs the ADBM service. This action is available only if a service has been added but not yet installed.

Reinstall

Reinstalls the ADBM service.

On the ***Configuration*** tab of the window that opens, in the ***Reboot timeout, sec*** field, enter the timeout to wait for hosts to restart (in seconds). The default value is `1800`. The value must be at least `600`.



Run an action: Reinstall

Reboot timeout, sec: 1800

1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

The Reinstall window

Reconfigure & Restart

Updates the service configuration according to the configuration settings and restart the service.

In the window that opens, use the ***Unlock adbm db*** flag to determine what to do with database locks that may occur if a migration failed. When the flag is set to `true`, ADBM releases the locks and makes its database available for migrations. The default value is `false`: preserve locks to avoid potentially conflicting migrations.

Start

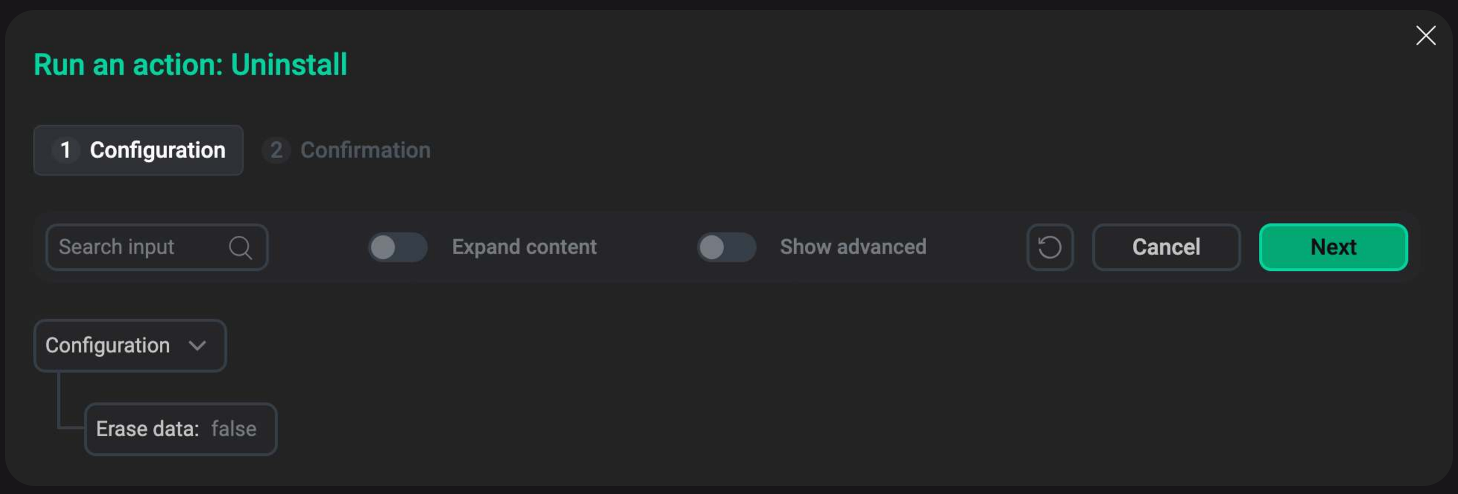
Starts the ADBM service.

Stop

Stops the ADBM service.

Uninstall

Uninstalls the ADBM service. In the window that opens, use the ***Erase data*** flag to select whether you need to delete files associated with a service.



Run an action: Uninstall

Erase data: false

1 Configuration
2 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

The Uninstall window

PXF service actions

Anton Vasilev

Contents

To Table of Contents

Install

Reconfigure

Restart

Start

Stop

Uninstall

This article describes the actions available for the PXF service in the ADCM web interface.

Install

Installs the PXF service. This action is available only if a service has been added but not yet installed.

Reconfigure

Updates the service configuration according to the configuration settings. If a changed parameter requires a restart, the action restarts the service.

Restart

Restarts the PXF service.

Start

Starts the PXF service.

Stop

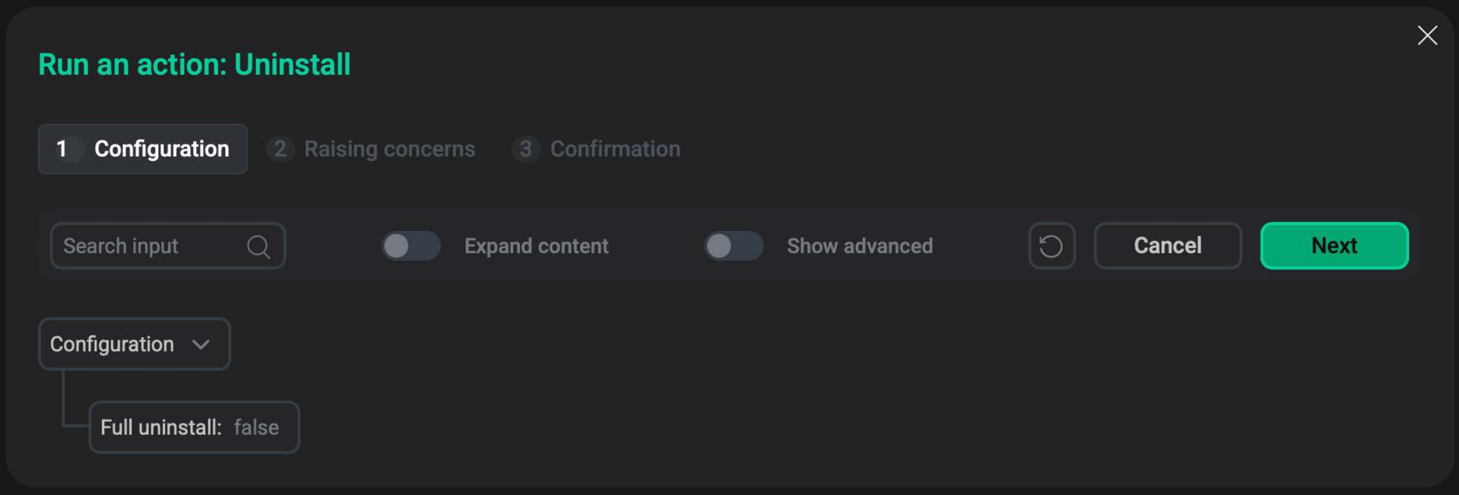
Stops the PXF service.

Uninstall

Uninstalls the PXF service.

> **NOTE**
>
> The ADB to Kafka and ADB ClickHouse connectors depend on the PXF service. You cannot uninstall PXF if you use these connectors.

In the window that opens, set the ***Full uninstall*** flag to `true` if you want to delete all PXF-related data: configuration files, the `pxf` user, the `pxf` and `pxf_fdw` database extensions, and objects that depend on these extensions. The default value is `false`.



Run an action: Uninstall

Full uninstall: false

1 Configuration
2 Raising concerns
3 Confirmation
Search input
Expand content
Show advanced
Cancel
Next
Configuration

The PXF → Uninstall window

Connector service actions

Anton Vasilev

Contents

To Table of Contents

Install

Reinstall

Uninstall

The actions described below are available for the following connector services:

- ADB ClickHouse Connector
- ADB to Kafka Connector
- Kafka to ADB Connector
- ADB to ADB Connector

Install

Installs the connector service. This action is available only if a service has been added but not yet installed.

Reinstall

Reinstalls the connector service.

Uninstall

Uninstalls the connector service.

Chrony service actions

Anton Vasilev

Contents

To Table of Contents

Install

Add/remove chronys component

Reinstall

Uninstall

The Chrony service synchronizes a system clock within the cluster. Segment hosts get the time from the coordinator host. This article describes the actions available for the Chrony service in the ADCM web interface.

Install

Installs the Chrony service. This action is available only if a service has been added but not yet installed.

After installation, you can check the current time sources that are used in a host with the following command:

```
$ chronyc sources
```

For example, if the command is run on a segment host, the output shows that the coordinator host is used for synchronization and the standby coordinator is used as a backup source:

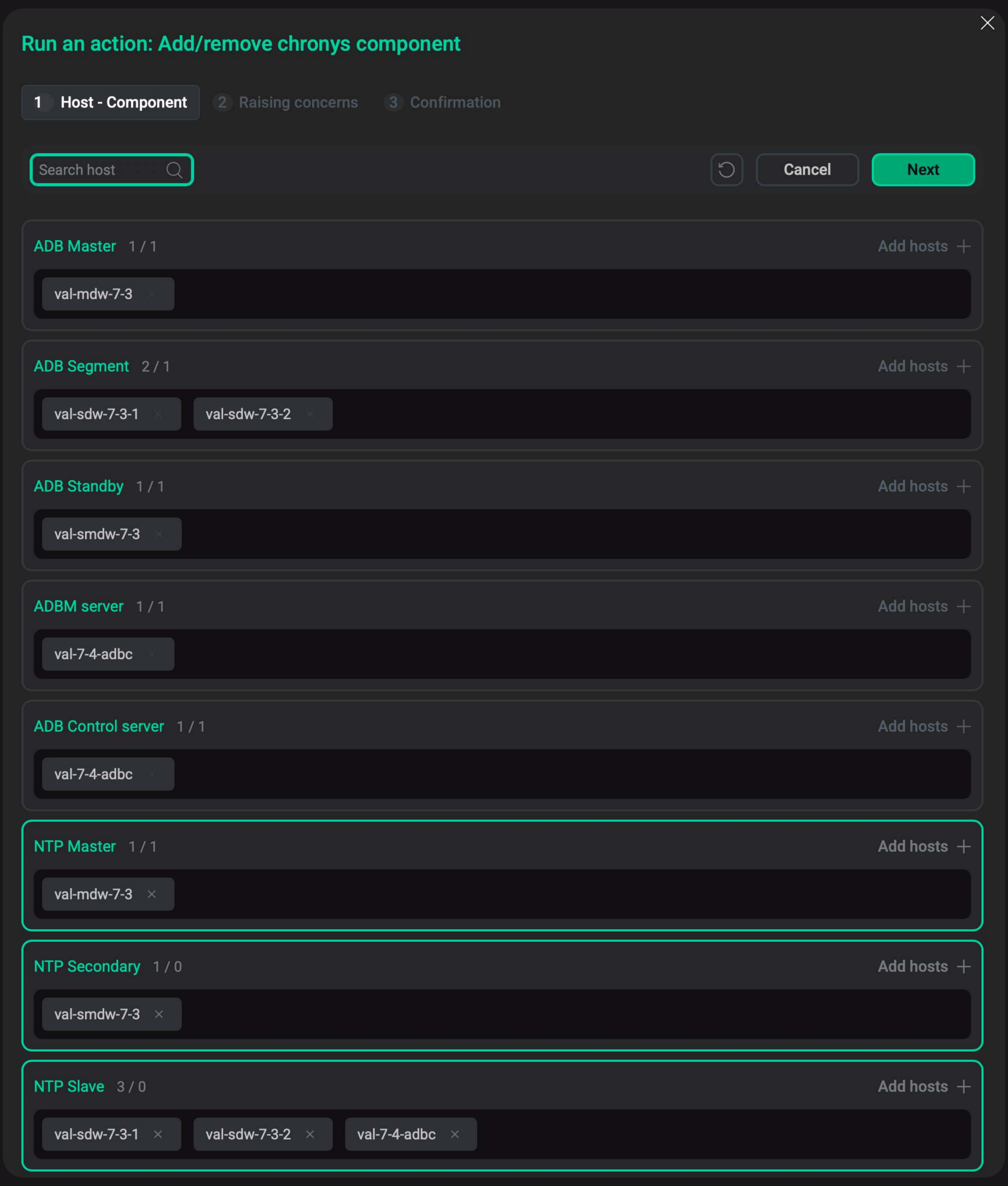
```
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* mdw                       10   9   377   411    -82us[  -98us] +/-  271us
^- smdw                      11   9   377    76   -606us[ -606us] +/- 1326us
```

Add/remove chronys component

Allows you to change host-component mapping for Chrony: install it on the selected hosts or remove it.

In the window that opens, click ***Add hosts*** in the ***NTP Master***, ***NTP Secondary***, or ***NTP Slave*** section and select hosts on which you want to install Chrony.

> **NOTE**
>
> If you use Chrony, it should be installed on every cluster host.



Run an action: Add/remove chronys component

ADB Master 1 / 1
val-mdw-7-3

ADB Segment 2 / 1
val-sdw-7-3-1
val-sdw-7-3-2

ADB Standby 1 / 1
val-smdw-7-3

ADBM server 1 / 1
val-7-4-adbc

ADB Control server 1 / 1
val-7-4-adbc

NTP Master 1 / 1
val-mdw-7-3

NTP Secondary 1 / 0
val-smdw-7-3

NTP Slave 3 / 0
val-sdw-7-3-1
val-sdw-7-3-2
val-7-4-adbc

The Add/remove chronys component window

Reinstall

Reinstalls the Chrony service.

Uninstall

Uninstalls the Chrony service.

Configure DBMS via GUCs

Daria Barysheva

Contents

To Table of Contents

- Overview
- Set parameters
  - session or system
  - coordinator or local
  - restart or reload
  - superuser
  - read only
- Get parameters
- Examples

Overview

Server configuration parameters (also known as **Grand Unified Configuration, GUC**) allow ADB administrators to manage the DBMS state and configuration. Many of these parameters have the same names and behaviors as in PostgreSQL.

> NOTE
>
> To see the full list of server configuration parameters, refer to Greenplum documentation.

The main GUC features are listed below:

- Parameter names and values are case-insensitive.
- Every parameter takes a value of one of the following types: boolean, integer, floating point, enum, or string.
- Boolean values may be specified as:
  - `true`: `ON`, `TRUE`, `YES`, `1`;
  - `false`: `OFF`, `FALSE`, `NO`, `0`.
- Enum parameters are specified in the same way as string parameters, but are limited by the specified set of values.
- For those parameters that specify the memory size, you can use the following size units:
  - `kB` — kilobytes;
  - `MB` — megabytes;
  - `GB` — gigabytes.

  Remember that the multiplier for memory units is `1024`, not `1000`.
- For those parameters that specify time values, you can use the following time units:
  - `ms` — milliseconds;
  - `s` — seconds;
  - `min` — minutes;
  - `h` — hours;
  - `d` — days.
- When specifying a memory or time parameter, enclose its value in single quotes `'` and do not use any spaces between the value and the unit name. For example:

```
SET effective_cache_size = '20GB';
```

Set parameters

Every server configuration parameter belongs to one of the categories (see the Set Classifications column in the Greenplum documentation), which define limitations on who and how can change this parameter. All categories are described in detail below.

session or system

This classification defines whether the parameter can be changed "on the fly" (without the DBMS restart and the configuration file reload):

- **session** — parameters of this type can be changed "on the fly" at one of the following levels:
  - session:

    ```
    SET <parameter> = <value>;
    ```
  - database and role:

    ```
    ALTER ROLE <role_name> IN DATABASE <database_name> SET <parameter> = <value>;
    ```
  - role:

    ```
    ALTER ROLE <role_name> SET <parameter> = <value>;
    ```
  - database:

    ```
    ALTER DATABASE <database_name> SET <parameter> = <value>;
    ```

  You can also define session parameters at the system level — via the *postgresql.conf* file (using the `gpconfig` utility). In this case, session parameters cannot be applied "on the fly" and require a system restart or a configuration file reload (like system parameters that are described below).

  If a session parameter is set at multiple levels, then the most granular setting has a higher precedence. For example, the parameter value that is defined at the session level overrides all other settings. The value specified at the database and role level overrides other level settings except the session one, and so on. The system level has the lowest priority.
- **system** — parameters of this type can only be changed via the *postgresql.conf* file (using the `gpconfig` utility). You cannot alter these parameters "on the fly". To define a system parameter value, run one of the following commands and then perform a restart or reload operation (depending on the parameter classification):
  - Set the parameter value on coordinator only. Use this command for coordinator parameters:

```
$ gpconfig -c <parameter> -v <value> --coordinatoronly
```

> CAUTION
>
> This command cannot be applied to the `gp_default_storage_options` parameter, despite this parameter belonging to the coordinator category. If you try to set different values for this parameter on coordinator and segment hosts, you get the following error: `gp_default_storage_options value cannot be different on coordinator and segments`. Use the next command instead.

  - Set the parameter value on all segments (including coordinator):

```
$ gpconfig -c <parameter> -v <value>
```

  - Set different values of the specified parameter on segment hosts and coordinator:

```
$ gpconfig -c <parameter> -v <value_segment> -m <value_coordinator>
```

  - Comment the parameter out in all configuration files, thus restoring its default value:

```
$ gpconfig -r <parameter>
```

coordinator or local

This classification applies only to parameters which values are changed via the *postgresql.conf* file (system or session at the system level):

- **coordinator** — parameters of this type should be set in the configuration file of the coordinator instance. The values for these parameters are then passed to or ignored by segments at run time.
- **local** — parameters of this type should be set in the configuration file of every segment (including coordinator). Each segment looks through its own configuration file to get the parameter value. Local parameters always require a DBMS restart for changes to be applied.

restart or reload

This classification applies only to parameters which values are changed via the *postgresql.conf* file (system or session at the system level):

- **restart** — parameters of this type require a DBMS restart:

```
$ gpstop -r -a
```

- **reload** — parameters of this type do not require a DBMS restart and can be refreshed by just reloading the server configuration file:

```
$ gpstop -u -a
```

superuser

Parameters of this type can only be set by a database superuser. Regular users cannot change values of such parameters.

read only

Parameters of this type cannot be set by any user (neither a regular database user nor a superuser). You can read the current values of such parameters but not alter them.

Get parameters

To show the current parameter value, use one of the following ways:

- If you use `psql` or any other client program, run the `SHOW <parameter>;` command. For example:

```
SHOW effective_cache_size;
```

  The result can be like:

```
 effective_cache_size
----------------------
 16GB
(1 row)
```

> IMPORTANT
>
> For session parameters that are defined at multiple levels, the `SHOW` command takes into account the priority of settings. The session level has the highest precedence, the system level — the lowest one.

- To get the parameter value that is stored in the *postgresql.conf* file (at the system level), log out from `psql` or any other client and run the following command: `gpconfig -s <parameter>`. This command shows the parameter values separately for coordinator and segment hosts. It also checks whether values on all segments are consistent. Example:

```
$ gpconfig -s effective_cache_size
```

  Result:

```
Values on all segments are consistent
GUC              : effective_cache_size
Coordinator value: 16GB
Segment     value: 16GB
```

Examples

Session parameter (session level)

Below is an example of changing the `optimizer` parameter. This parameter has the `coordinator, session, reload` classification. To change the parameter value at the session level, do the following:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

   The output is:

```
psql (12.12)
Type "help" for help.
```

3. Get the current parameter value:

```
SHOW optimizer;
```

   Result:

```
 optimizer
-----------
 on
(1 row)
```

4. Change the parameter value to `off` at the session level:

```
SET optimizer = off;
```

   Result:

```
SET
```

5. Get the current parameter value again:

```
SHOW optimizer;
```

   The value has changed:

```
 optimizer
-----------
 off
(1 row)
```

6. Log out of `psql` (or any other client you use):

```
\q
```

7. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

8. Get the current parameter value again:

```
SHOW optimizer;
```

   In another session, the parameter still has a default value:

```
 optimizer
-----------
 on
(1 row)
```

Session parameter (database level)

Below is an example of changing the `gp_external_max_segs` parameter. This parameter has the `coordinator, session, reload` classification. To change the parameter value at the database level, do the following:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

   The output is:

```
psql (12.12)
Type "help" for help.
```

3. Get the current parameter value:

```
SHOW gp_external_max_segs;
```

   Result:

```
 gp_external_max_segs
----------------------
 64
(1 row)
```

4. Change the parameter value to `32` for the `adb` database:

```
ALTER DATABASE adb SET gp_external_max_segs = 32;
```

   Result:

```
ALTER DATABASE
```

5. Log out of `psql` (or any other client you use):

```
\q
```

6. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

7. Get the current parameter value again:

```
SHOW gp_external_max_segs;
```

   The value has changed:

```
 gp_external_max_segs
----------------------
 32
(1 row)
```

> TIP
>
> When changing the session configuration parameter at the database level, do not forget to restart the current database session. Otherwise, you cannot see the changes.

8. Connect to any other database:

```
\c books_store
```

   Result:

```
You are now connected to database "books_store" as user "gpadmin".
```

9. Check the current parameter value again:

```
SHOW gp_external_max_segs;
```

   In another database, the parameter still has a default value:

```
 gp_external_max_segs
----------------------
 64
(1 row)
```

System parameter (coordinator and reload)

Below is an example of changing the `gp_connection_send_timeout` parameter. This parameter has the `coordinator, system, reload` classification. To change the parameter value, do the following:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Get the current parameter values:

```
$ gpconfig -s gp_connection_send_timeout
```

   Result:

```
Values on all segments are consistent
GUC              : gp_connection_send_timeout
Coordinator value: 3600
Segment     value: 3600
```

3. Change the parameter value to `1800` on coordinator:

```
$ gpconfig -c gp_connection_send_timeout -v 1800 --coordinatoronly
```

   Result:

```
20221214:19:32:05:010670 gpconfig:bds-mdw:gpadmin-[INFO]:-completed successfully with parameters '-c
gp_connection_send_timeout -v 1800 --coordinatoronly'
```

4. Get the current parameter values again:

```
$ gpconfig -s gp_connection_send_timeout
```

   They are still the same:

```
Values on all segments are consistent
GUC              : gp_connection_send_timeout
Coordinator value: 3600
Segment     value: 3600
```

5. Reload the configuration file:

```
$ gpstop -u -a
```

6. Check the current parameter values again:

```
$ gpconfig -s gp_connection_send_timeout
```

   The value is changed on coordinator:

```
Values on all segments are consistent
GUC              : gp_connection_send_timeout
Coordinator value: 1800
Segment     value: 3600
```

   As the `gp_connection_send_timeout` parameter belongs to the `coordinator` category, it is not required to change the parameter value on other segments.

System parameter (local and restart)

Below is an example of changing the `max_connections` parameter. This parameter has the `local, system, restart` classification. To change the parameter value, do the following:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Get the current parameter values:

```
$ gpconfig -s max_connections
```

   Result:

```
Values on all segments are consistent
GUC              : max_connections
Coordinator value: 250
Segment     value: 750
```

3. Change the parameter value on coordinator to `300` and on segment hosts — to `800`:

```
$ gpconfig -c max_connections -v 800 -m 300
```

   Result:

```
20221214:19:19:15:009073 gpconfig:bds-mdw:gpadmin-[INFO]:-completed successfully with parameters '-c max_connections -v 800 -m
300'
```

4. Get the current parameter values again:

```
$ gpconfig -s max_connections
```

   They are still the same:

```
Values on all segments are consistent
GUC              : max_connections
Coordinator value: 250
Segment     value: 750
```

5. Restart the system:

```
$ gpstop -r -a
```

6. Check the current parameter values again:

```
$ gpconfig -s max_connections
```

   The values are changed on coordinator and segments according to step 3:

```
Values on all segments are consistent
GUC              : max_connections
Coordinator value: 300
Segment     value: 800
```

Use ADB DDBoost plugin

Daria Barysheva

Contents

To Table of Contents

Overview
Installation and configuration
Usage examples
  gpbackup
  gprestore

Overview

ADB DDBoost is a plugin-connector for native usage of the Dell EMC Data Domain storage system with Greenplum utilities for logical data backup (gpbackup) and restore (gprestore).

When the DDBoost plugin is used with `gpbackup` and `gprestore`, it connects to the Data Domain Platform server. To integrate with Data Domain Platform, the plugin requires the *libDDBoost.so* library version 7.7, which is supplied as part of the Dell DDBoostSDK 7.7 package. This library is proprietary and should be obtained through Dell Support or Dell Data Domain representatives in your country. Due to deduplication of data provided by the library, the DDBoost plugin allows you to significantly accelerate backup operations and reduce the physical storage usage.

NOTE

- For more information on the ADB DDBoost plugin architecture, advantages, and load testing results, see Arenadata technical blog.
- The DDBoost plugin is available in the Enterprise version of ADB.

Installation and configuration

To install the ADB DDBoost plugin, follow the steps:

1. Install the *adb-ddp-plugin* package **on all cluster hosts** under the user with `sudo` privileges. Below is an example of installation via the APT package manager:

```
$ sudo apt install adb-ddp-plugin
```

As a result of the successful command execution, a plugin executable file is added to the following location: *$GPHOME/bin/adb_ddp_plugin*.

2. Put the *libDDBoost.so* library to the *lib* directory on **all cluster hosts**.
3. Connect to the ADB **coordinator host** under the `gpadmin` user:

```
$ sudo su - gpadmin
```

4. On the **coordinator host**, create a YAML configuration file in the */home/gpadmin* directory (for example, *adb_ddp_plugin.yaml*). A file structure and its description are given below.

```
executablepath: /usr/lib/ggdb/bin/adb_ddp_plugin
options:
  hostname: "testhost"
  username: "testuser"
  password: "XXXXXXXXXXXX"
  storage_unit: "testunit"
  directory: "testdir"
  write_buffer_size: "1048576"
  read_buffer_size: "1048576"
  log_level: "DEBUG"
  log_path: "/home/gpadmin"
```

Fields of the plugin configuration file

Name	Description	Default	Required
executablepath	An absolute path to the plugin in the file system of ADB hosts (along with the plugin executable file name). The plugin should be installed on all cluster hosts in the same directory. The default path to the plugin executable file is *$GPHOME/bin/adb_ddp_plugin*	—	Yes
hostname	An IP address or name of the host that provides operations with DDBoost. Can contain no more than 30 characters	—	Yes
username	A name of the user who is granted permissions to work with DDBoost. The user is configured on the DDBoost side. This is neither operating system nor ADB user. Can contain no more than 30 characters	—	Yes
password	A password of the user who is granted permissions to work with DDBoost	—	Yes
storage_unit	A name of the storage unit that is configured on the DDBoost side	—	Yes
directory	A name of the directory in the DDBoost file system. That directory is used to store all backup files created when the `gpbackup` utility is run with the DDBoost plugin. In the */<storage_unit>/<directory>* folder, the plugin automatically creates all subdirectories corresponding to the creation date and time of each backup: */<storage_unit>/<directory>/YYYYMMDD/YYYYMMDDHHmmSS/*	—	Yes
write_buffer_size	A buffer size for writing data to DDBoost (in bytes). Values from the following range are allowed: 64 <= write_buffer_size <= 1048576	—	Yes
read_buffer_size	A buffer size for reading data from DDBoost (in bytes). Values from the following range are allowed: 64 <= read_buffer_size <= 1048576	—	Yes

log_level	A log level. Possible values:
	`DEBUG`
	`INFO`
	`NOTICE`
	`WARN`
	`ERROR`
	`FATAL`
		WARN	No

log_path	An absolute directory path to write plugin logs. The *adb_ddp_plugin.log* file is created in the specified directory when the plugin is used	/home/gpadmin/gpAdminLogs	No

Usage examples

gpbackup

To use the ADB DDBoost plugin when creating backups via the `gpbackup` utility, specify the `--plugin-config <yaml_path>` command argument, where `<yaml_path>` is an absolute path to the plugin configuration file in YAML format.

The following example shows how to back up one table of the `adb` database:

1. Connect to the `adb` database on ADB coordinator under the `gpadmin` user (for example, via `psql`). Using the `psql` command `\dt`, ensure the required table (`public.test` in the given example) exists in the database:

```
\dt
```

Result:

```
                          List of relations
 Schema |      Name       | Type  |  Owner  |       Storage
--------+-----------------+-------+---------+----------------------
 public | spatial_ref_sys | table | gpadmin | heap
 public | test            | table | gpadmin | heap
 public | test2           | table | gpadmin | ao_column
 public | test3           | table | gpadmin | ao_column
(4 rows)
```

2. Disconnect from `adb` and run `gpbackup` with the `--plugin-config` parameter under the `gpadmin` user. Note that the `--include-table` parameter allows you to back up one table:

```
$ gpbackup --dbname adb --no-compression --single-data-file --include-table public.test --plugin-config
/home/gpadmin/adb_ddp_plugin.yaml
```

TIP

When the DDBoost plugin is used along with `gpbackup`, it is recommended to set the backup mode "all segment tables — one file" (`--single-data-file`). Do not use the parallel backup mode (`--jobs`) because of its extremely low write performance.

In the command output, note the backup timestamp, which will be used as the backup directory and will be needed when restoring data:

```
[INFO]:-Backup Timestamp = 20251112160114
```

If the data backup succeeds, the command output ends with the following message:

```
[INFO]:-Backup completed successfully
```

3. Ensure that the backup files are generated and stored on the selected Data Domain Platform server. To do this, open the */<storage_unit>/<directory>/YYYYMMDD/YYYYMMDDHHmmSS* directory on the server, where:
   - `<storage_unit>` and `<directory>` — values of the corresponding fields from the plugin YAML configuration file.
   - `YYYYMMDD` — backup creation date.
   - `YYYYMMDDHHmmSS` — backup creation date and time. It is a timestamp that will be used as the `--timestamp` value when restoring data from the current backup.

```
$ ls ./<storage_unit>/<directory>/20251112/20251112160114/
```

The directory contents are as follows:

```
gpbackup_0_20251112160114                   gpbackup_3_20251112160114
gpbackup_0_20251112160114_toc.yaml          gpbackup_3_20251112160114_toc.yaml
gpbackup_1_20251112160114                   gpbackup_4_20251112160114
gpbackup_1_20251112160114_toc.yaml          gpbackup_4_20251112160114_toc.yaml
gpbackup_20251112160114_config.yaml         gpbackup_5_20251112160114
gpbackup_20251112160114_metadata.sql        gpbackup_5_20251112160114_toc.yaml
gpbackup_20251112160114_plugin_config.yaml  gpbackup_6_20251112160114
gpbackup_20251112160114_report              gpbackup_6_20251112160114_toc.yaml
gpbackup_20251112160114_toc.yaml            gpbackup_7_20251112160114
gpbackup_2_20251112160114                   gpbackup_7_20251112160114_toc.yaml
gpbackup_2_20251112160114_toc.yaml
```

On the coordinator host, in the *$COORDINATOR_DATA_DIRECTORY/backups/YYYYMMDD/YYYYMMDDHHmmSS* directory, a report about the completed backup is available with information about the number of objects in the backup copy:

```
$ cat /data1/master/gpseg-1/backups/20251112/20251112160114/gpbackup_20251112160114_report
```

Example of a report for a successful one-table backup

```
Greengage Database Backup Report

timestamp key:         20251112160114
gpdb version:          7.3.0+dev.845.g704bb9a9ef build 256+git704bb9a
gpbackup version:      1.30.6

database name:         adb
command line:          gpbackup --dbname adb --no-compression --single-data-file --include-table public.test --plugin-config
/home/gpadmin/adb_ddp_plugin.yaml
compression:           None
plugin executable:     adb_ddp_plugin
backup section:        All Sections
object filtering:      Include Table Filter
includes statistics:   No
data file format:      Single Data File Per Segment
incremental:           False

start time:            Wed Nov 12 2025 16:01:14
end time:              Wed Nov 12 2025 16:01:52
duration:              0:00:38

backup status:         Success

segment count:         8

count of database objects in backup:
constraints          0
conversions          0
default privileges   0
event triggers       0
indexes              0
policies             0
rules                0
statistics_ext       0
sequences            0
tables               1
triggers             0
views                0
```

gprestore

To use the ADB DDBoost plugin when restoring data via the `gprestore` utility, specify the `--plugin-config <yaml_path>` command argument, where `<yaml_path>` is an absolute path to the plugin configuration file in YAML format.

The following example shows how to restore the `public.test` table, for which a backup was previously created:

1. Connect to the `adb` database on ADB coordinator under the `gpadmin` user (for example, via `psql`). Drop the `public.test` table:

```
DROP TABLE test;
```

2. Ensure the table does not exist via the `psql` command `\dt`:

```
\dt
```

Result:

```
                          List of relations
 Schema |      Name       | Type  |  Owner  |       Storage
--------+-----------------+-------+---------+----------------------
 public | spatial_ref_sys | table | gpadmin | heap
 public | test2           | table | gpadmin | ao_column
 public | test3           | table | gpadmin | ao_column
(3 rows)
```

3. Disconnect from `adb` and run `gprestore` with the `--plugin-config` parameter under the `gpadmin` user. Note that the `--timestamp` value should contain the backup creation timestamp in the `YYYYMMDDHHmmSS` format:

```
$ gprestore --timestamp 20251112160114 --plugin-config /home/gpadmin/adb_ddp_plugin.yaml --include-table public.test
```

If the data restore succeeds, the command output ends with the following message:

```
[INFO]:-Restore completed successfully
```

4. Connect to the `adb` database again and check that the `public.test` table is available:

```
\dt
```

Result:

```
                          List of relations
 Schema |      Name       | Type  |  Owner  |       Storage
--------+-----------------+-------+---------+----------------------
 public | spatial_ref_sys | table | gpadmin | heap
 public | test            | table | gpadmin | heap
 public | test2           | table | gpadmin | ao_column
 public | test3           | table | gpadmin | ao_column
(4 rows)
```

Work with arenadata_toolkit

Daria Barysheva

Contents

To Table of Contents

- Schema tables
- Cluster management scripts
  - Overview
  - Example of a custom function for VACUUM

When you install an ADB cluster, the `arenadata_toolkit` schema is automatically created in the default database. This schema is used to collect information on cluster operations. The schema includes several tables, which allow users to get results of the `VACUUM` / `ANALYZE` commands that are launched on a schedule, view information about data files, etc. Additionally, ADB provides a set of scripts that interact with `arenadata_toolkit` and automatically perform various cluster management operations (according to the predefined schedule).

> **NOTE**
>
> In custom databases that are added after ADB installation, the `arenadata_toolkit` schema is created automatically when the operation.py script is first run.

Schema tables

The tables included in the `arenadata_toolkit` schema are described below.

daily_operation

The `arenadata_toolkit.daily_operation` monthly-partitioned table contains information about the `VACUUM` and `ANALYZE` operations that are automatically applied to database tables on a schedule. The table contents are updated daily when the vacuum, analyze, vacuum_analyze_pg_catalog scripts are run.

Column	Description
schema_name	A schema name
table_name	A table name
action	An operation name. Possible values: - `ANALYZE` - `VACUUM`
status	An operation status. Possible values: - `SCS` — success. - `ERROR` — error.
time	An operation duration (in seconds)
processed_dttm	Operation date and time

db_files_current

The `arenadata_toolkit.db_files_current` table contains information about data files on all cluster segments with links to tables, indexes, and other database objects (if possible). The table contents reflect the state at the time of the last collect_table_stats script launch. The table is overwritten every time the script is run.

Column	Description
oid	A table object identifier (OID)
table_name	A table name
table_schema	A schema name
type	A table type. See pg_class.relkind
relamname	A table access method. See pg_am.amname
table_parent_table	A parent table name (`NULL` if missing). Available for partitions
table_parent_schema	A parent table schema (`NULL` if missing). Available for partitions
table_database	A database name
table_tablespace	A tablespace name
content	A content identifier of the segment instance where the data file is located. See gp_segment_configuration.content
segment_preferred_role	The role that a segment was originally assigned at initialization time. See gp_segment_configuration.preferred_role
hostname	A name of the host where the data file is located
address	The hostname used to access a particular segment instance on a segment host. This value may be the same as `hostname` on systems that do not have per-interface hostnames configured
file	A full path to the data file
modifiedtime	The last file modification time that is returned by the system command `stat`. The value of this timestamp is updated during the file modifications, for example when the following commands are run: `mknod`, `truncate`, `utime`, and `write` (if more than zero bytes are written). The timestamp value does not change when the inode information is being modified (file owner, group, number of hard links, access mode, etc.)
file_size	A data file size (in bytes)
tablespace_location	The file system location of the tablespace

db_files_history

The `arenadata_toolkit.db_files_history` monthly-partitioned table contains the history of data file changes on all cluster segments with links to tables, indexes, and other database objects (if possible). The table may be useful for tracking the database size changes. The table structure is identical to `arenadata_toolkit.db_files_current`, except for the additional column `collecttime` described below.

The `arenadata_toolkit.db_files_history` contents are updated daily when the collect_table_stats script is run (by copying actual data from `arenadata_toolkit.db_files_current`).

Column	Description
collecttime	Data collection time

operation_exclude

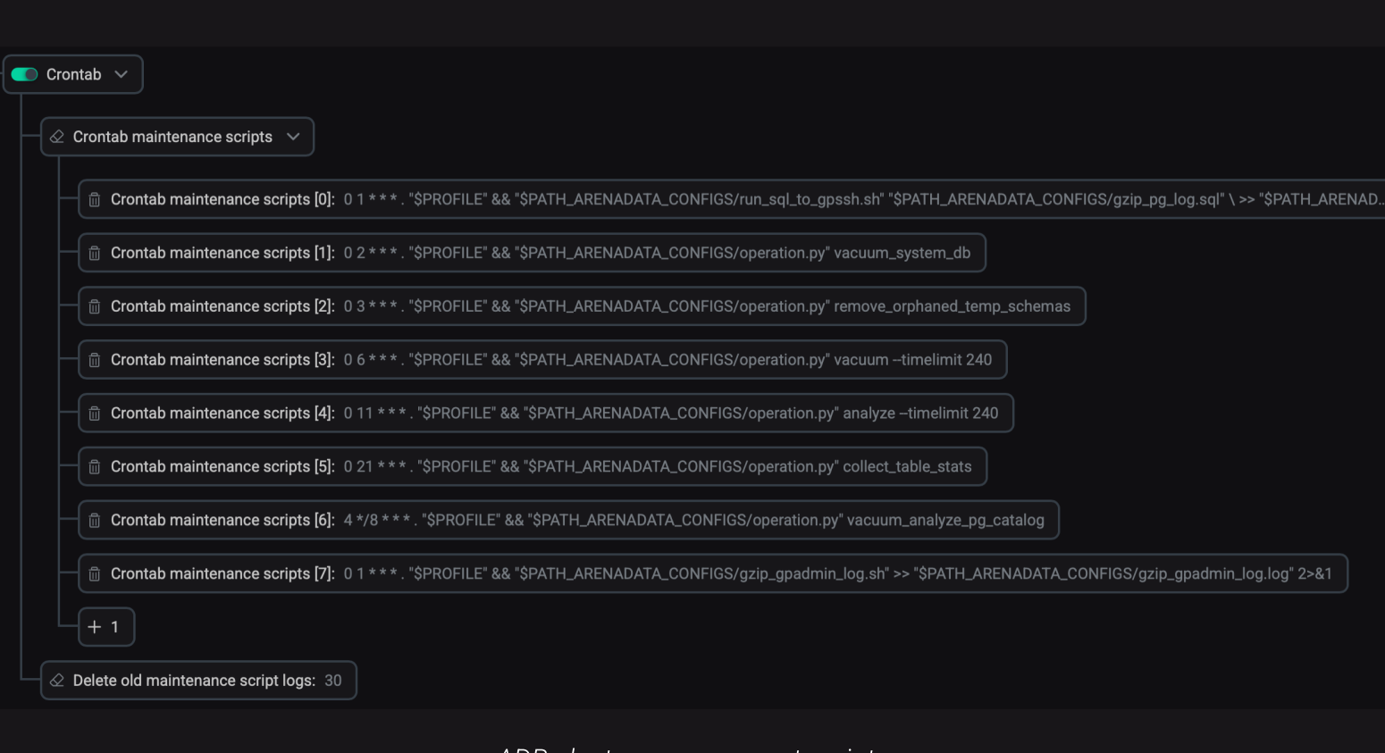
The `arenadata_toolkit.operation_exclude` table contains information on database schemas that should be skipped by the `VACUUM` and `ANALYZE` commands when the vacuum and analyze scripts are run.

Column	Description
schema_name	A schema name
backup	The flag that controls whether the row should be included in backups when using the `gpbackup`, `gprestore`, and `pg_dump` utilities. The default value is `true`, meaning the row will be restored. Rows that are automatically added by the `arenadata_toolkit` extension upon initial table creation (`gp_toolkit`, `information_schema`, `pg_aoseg`, `pg_bitmapindex`, `pg_catalog`, and `pg_toast`) have a default value of `false` to prevent them from being re-added during restoration

Cluster management scripts

Overview

ADB maintenance scripts are displayed on the ***Primary configuration*** tab of the ADB service configuration page. To view all scripts, expand the ***Crontab → Crontab maintenance scripts*** node in the tree of configuration settings.



Crontab maintenance scripts [0]: 0 1 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/run_sql_to_gpssh.sh" "$PATH_ARENADATA_CONFIGS/gzip_pg_log.sql" \ >> "$PATH_ARENAD...
Crontab maintenance scripts [1]: 0 2 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum_system_db
Crontab maintenance scripts [2]: 0 3 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" remove_orphaned_temp_schemas
Crontab maintenance scripts [3]: 0 6 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum –timelimit 240
Crontab maintenance scripts [4]: 0 11 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" analyze –timelimit 240
Crontab maintenance scripts [5]: 0 21 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" collect_table_stats
Crontab maintenance scripts [6]: 4 */8 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum_analyze_pg_catalog
Crontab maintenance scripts [7]: 0 1 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.sh" >> "$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.log" 2>&1
Delete old maintenance script logs: 30

ADB cluster management scripts

> **CAUTION**
>
> If you need to configure custom scripts, use the system `crontab` (for example, */etc/cron.d/custom_gpadmin*) under the user other than `gpadmin`. Do not use the ***Crontab → Crontab maintenance scripts*** section in ADCM for this purpose. Otherwise, after switching off the ***Crontab*** toggle and reconfiguring ADB, the custom scripts added via ADCM will be deleted.

The `cron` scheduler is used to run scripts automatically. Scripts are launched on behalf of the system user (by default, `gpadmin`). In the ADCM interface, each script is described as a separate string including:

- **Start time** — a `cron` expression that defines how often a script should be run.
- **Path to the script** — a full path to the script. Currently, two main scripts are used:
  - *run_sql_to_gpssh.sh* — sends the result of the selected SQL query as a bash command via `gpssh`. Requires a path to the SQL script as an argument.
  - *operation.py* — the Python script that performs various operations at the ADB cluster level. The operation type is defined by the first argument (for example, `vacuum` or `analyze`).

  Each of these scripts can in turn refer to other SQL queries, scripts, and functions of the `arenadata_toolkit` schema. All bash, Python, and SQL scripts are stored in the */home/gpadmin/<Arenadata_configs_directory_name>/* directory, where `<Arenadata_configs_directory_name>` is a value of the ***Advanced → Arenadata configs directory name*** ADB configuration parameter (by default, `arenadata_configs`).
- **Arguments** — input parameters that are used to run the script (if available). The arguments available for the *operation.py* script are listed below.

operation.py arguments

Argument	Description	Default value
The first positional parameter (without alias)	An operation name: - `vacuum_system_db` - `remove_orphaned_temp_schemas` - `vacuum` - `analyze` - `collect_table_stats` - `vacuum_analyze_pg_catalog` For information on all operations, see Purpose of ADB cluster management scripts	—
--loglevel	A logging level. Possible values: - `2` — INFO; - `3` — WARNING; - `4` — ERROR; - `5` — CRITICAL. For more information on logging levels, see Python documentation	2
--processes	A number of worker processes. Possible values: `1` — `10`	10
--timelimit	The limit on the script execution time (in minutes). If the limit is reached, the operation will finish processing the current object, and then the script will stop: - For `vacuum` operations, it will stop after processing the current table. - For `analyze` operations — after processing the entire schema. - For `vacuum_system_db` operations — after processing the current system database (`postgres` or `template1`). - For `collect_table_stats`, `remove_orphaned_temp_schemas`, and `vacuum_analyze_pg_catalog` operations — after processing a database. When multiple workers are used (via `--processes`), multiple objects can be held at the same time. The script will wait for each of them to be processed, and the workers won't start processing a new object after the limit is reached	—
--excludedb	A list of databases that should be excluded from the script process. This list extends databases that are skipped by *operation.py* by default: - `template0` - `template1` - `postgres`	—
--vacuum-order	The order in which the `VACUUM` command should be applied to tables from custom databases. You can use this parameter only when running the vacuum operation. Possible values: - `newest-first` — the tables for which there is no information on the `VACUUM` command in `pg_catalog.pg_stat_last_operation` are processed first. The `arenadata_toolkit.adb_vacuum_strategy_newest_first` function is used to sort tables. - `newest-last` — the tables for which there is no information on the `VACUUM` command in `pg_catalog.pg_stat_last_operation` are processed last. The `arenadata_toolkit.adb_vacuum_strategy_newest_last` function is used to sort tables. - A custom function name in the `<schema_name>.<function_name>` format. The custom function signature should be identical to the two functions mentioned above. It should return a list of rows, each containing the schema name and table name. If the function is absent in some databases, a warning message is stored in the log and the `VACUUM` command is not applied to the corresponding databases. For more information, see Example of a custom function for VACUUM.	newest-first

Script logs are stored in the */home/gpadmin/<Arenadata_configs_directory_name>/operation_log/* directory (for *operation.py*) and the *gzip_pg_log.log* file (for *gzip_pg_log.sql*). You can configure the log retention period via the ***Crontab → Delete old maintenance script logs*** configuration parameter of the ADB service. The default parameter value is 30 days.

> **TIP**
>
> It is not recommended to change scripts in any way. However, if you need to update the script start time or add/modify script arguments, follow the steps below:
>
> 1. Make necessary changes in the ***Crontab → Crontab maintenance scripts*** section on the ***Primary configuration*** tab of the ADB service configuration page.
> 2. Click ***Save***.
> 3. Apply the ***Reconfigure*** action to the ADB service.

Most scripts interact with the `arenadata_toolkit` schema. The purpose of all scripts is described below.

Purpose of ADB cluster management scripts

#	Script	Description	Start time HH:mm (UTC)
0	run_sql_to_gpssh.sh → gzip_pg_log.sql	For each cluster segment, archives CSV logs that are stored in *<data_directory>/pg_log* and have not changed for more than 3 days (where `<data_directory>` is a segment data directory)	01:00
1	operation.py → vacuum_system_db	Applies the VACUUM FREEZE command to the `postgres` and `template1` system databases (by running `vacuumdb -F`)	02:00
2	operation.py → remove_orphaned_temp_schemas	In all custom databases, drops temporary schemas that are no longer in use and whose names start with `pg_temp_` and `pg_toast_temp_`	03:00
3	operation.py → vacuum	- Runs the VACUUM command in all custom databases, except for schemas defined in the arenadata_toolkit.operation_exclude table. - Writes operation results to the arenadata_toolkit.daily_operation table.	06:00
4	operation.py → analyze	- Runs the ANALYZE command in all custom databases, except for schemas defined in the arenadata_toolkit.operation_exclude table. - Writes operation results to the arenadata_toolkit.daily_operation table.	11:00
5	operation.py → collect_table_stats	In all custom databases: 1. Collects statistics for data files on cluster segments (size, location, etc.) and tries to define file links with tables, indexes, and other database objects. To do this, the script runs a function, which gets information from the `gp_toolkit.gp_db_files_current` view. 2. Saves the received information in the arenadata_toolkit.db_files_current table after clearing its contents first. Some table columns may remain NULL if it was impossible to find corresponding database objects for data files in the previous step (for example, `oid` and `table_name`). 3. Adds new rows to the arenadata_toolkit.db_files_history table (by copying data from arenadata_toolkit.db_files_current).	21:00
6	operation.py → vacuum_analyze_pg_catalog	- Runs the VACUUM and ANALYZE commands in the `pg_catalog` schema of all custom databases. - Writes operation results to the arenadata_toolkit.daily_operation table.	00:04 08:04 16:04
7	gzip_gpadmin_log.sh	Performs automatic compression of log files that are located in */home/gpadmin/gpAdminLogs/* and are older than 3 days	01:00

Example of a custom function for VACUUM

Below you can find an example of using a custom function that determines the order in which user tables should be processed when performing the `VACUUM` operation (see `--vacuum-order` in the operation.py arguments table):

1. Create a function that meets the following conditions:

   - The function should have one input parameter of the `text` type. When the *operation.py* script is run, `VACUUM` is always passed as the parameter value. There is no need to use the parameter in the function body.
   - The function should return a table with the following columns: `table_schema` (schema in DB) and `table_name` (table in DB). This is a sorted list of database tables, to which the `VACUUM` operation should be applied.

   Suppose that you want certain ETL data in the staging environment to be processed by the `VACUUM` operation only after all other data has been processed. The following function moves tables whose names start with `etl_staging` to the end of the list and sorts tables by schema names and then by table names in the alphabetical order.

```
CREATE OR REPLACE FUNCTION public.adb_vacuum_strategy_etl_staging(actionname text)
    RETURNS TABLE (table_schema name, table_name name)
    LANGUAGE sql
    STABLE
AS $$
    SELECT nspname, relname
    FROM pg_catalog.pg_class c
    JOIN pg_catalog.pg_namespace n ON relnamespace = n.oid
    WHERE relkind = 'r'
      AND nspname NOT IN (SELECT schema_name FROM arenadata_toolkit.operation_exclude)
    ORDER BY
        CASE WHEN relname LIKE 'etl_staging%' THEN 1 ELSE 0 END,
        nspname,
        relname;
$$
EXECUTE ON MASTER;
```

> **CAUTION**
>
> The function should be added to all custom databases to which the `VACUUM` operation will be applied, i.e. all databases except for system ones (`template0`, `template1`, `postgres`) and those that can be optionally specified in the --excludedb argument of the *operation.py* script.

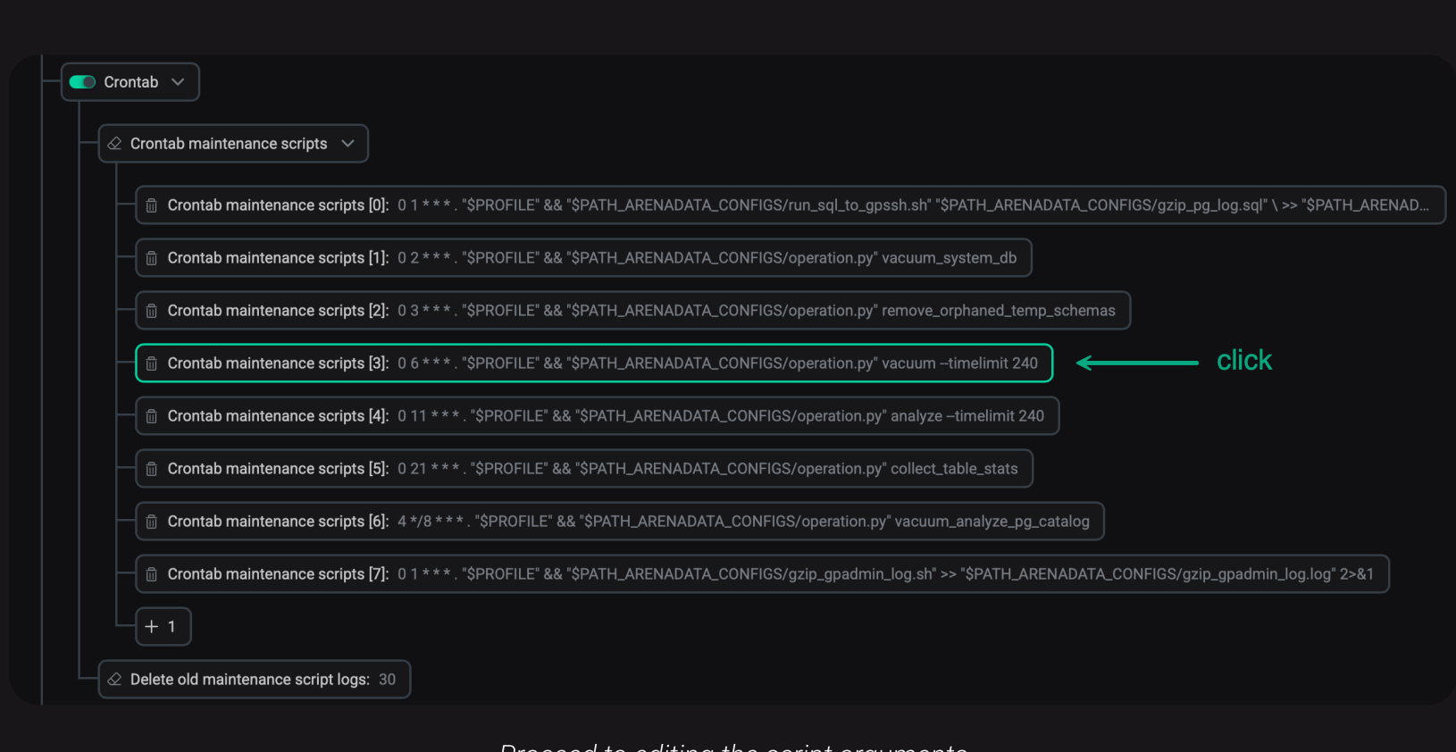
2. Ensure the function works:

```
SELECT * FROM public.adb_vacuum_strategy_etl_staging('VACUUM');
```

   In the resulting output, tables related to ETL staging (`etl_staging_*`) are placed at the very bottom:

```
  table_schema    |    table_name
------------------+-------------------------------
 arenadata_toolkit | daily_operation_1_prt_other
 arenadata_toolkit | daily_operation_1_prt_p202510
 arenadata_toolkit | daily_operation_1_prt_p202511
 arenadata_toolkit | db_files_current
 arenadata_toolkit | db_files_history_1_prt_other
 arenadata_toolkit | db_files_history_1_prt_p202510
 arenadata_toolkit | db_files_history_1_prt_p202511
 arenadata_toolkit | operation_exclude
 kadb              | offsets
 public            | spatial_ref_sys
 public            | etl_staging_main
 public            | etl_staging_orders
 public            | etl_staging_products
 public            | etl_staging_users
(14 rows)
```

3. In ADCM, open the ***Primary configuration*** tab on the configuration page of the ADB service.
4. Expand the ***Crontab → Crontab maintenance scripts*** node in the tree of configuration settings and click a description of the script that performs `vacuum`.

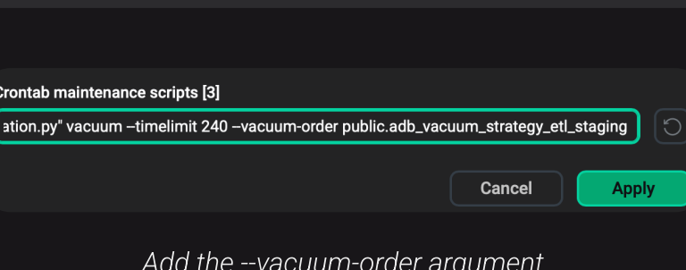


Crontab maintenance scripts [0]: 0 1 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/run_sql_to_gpssh.sh" "$PATH_ARENADATA_CONFIGS/gzip_pg_log.sql" \ >> "$PATH_ARENAD...
Crontab maintenance scripts [1]: 0 2 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum_system_db
Crontab maintenance scripts [2]: 0 3 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" remove_orphaned_temp_schemas
Crontab maintenance scripts [3]: 0 6 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum –timelimit 240
Crontab maintenance scripts [4]: 0 11 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" analyze –timelimit 240
Crontab maintenance scripts [5]: 0 21 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" collect_table_stats
Crontab maintenance scripts [6]: 4 */8 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum_analyze_pg_catalog
Crontab maintenance scripts [7]: 0 1 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.sh" >> "$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.log" 2>&1
Delete old maintenance script logs: 30

Proceed to editing the script arguments

5. In the window that opens, add the `--vacuum-order <schema_name>.<function_name>` argument to the end of the current script description. For example: `--vacuum-order public.adb_vacuum_strategy_custom`. Click ***Apply***.

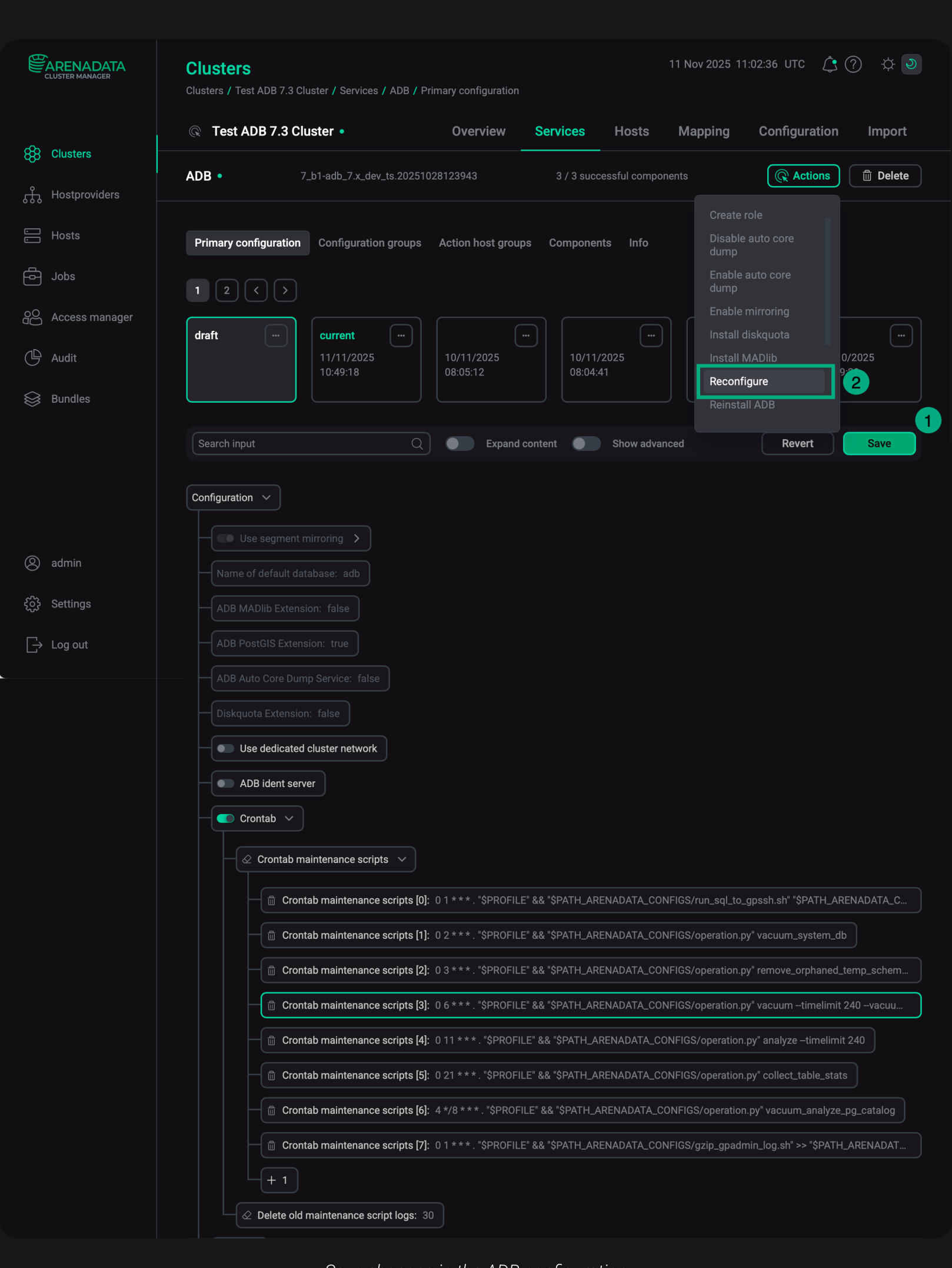
> **IMPORTANT**
>
> The custom function name should be defined in the following format: `<schema_name>.<function_name>`, where `<schema_name>` — a name of the schema where the function was created; `<function_name>` — the function name. Note that passing arguments to the function is not supported.



Crontab maintenance scripts [3]
ation.py" vacuum –timelimit 240 –vacuum-order public.adb_vacuum_strategy_etl_staging

Add the --vacuum-order argument

6. Click ***Save*** to save the configuration changes. Then, apply the ***Reconfigure*** action to the ADB service.



Use segment mirroring
Name of default database: adb
ADB MADlib Extension: false
ADB PostGIS Extension: true
ADB Auto Core Dump Service: false
Diskquota Extension: false
Use dedicated cluster network
ADB ident server
Crontab
Crontab maintenance scripts
Crontab maintenance scripts [0]: 0 1 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/run_sql_to_gpssh.sh" "$PATH_ARENADATA_C...
Crontab maintenance scripts [1]: 0 2 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum_system_db
Crontab maintenance scripts [2]: 0 3 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" remove_orphaned_temp_schem...
Crontab maintenance scripts [3]: 0 6 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum –timelimit 240 –vacuu...
Crontab maintenance scripts [4]: 0 11 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" analyze –timelimit 240
Crontab maintenance scripts [5]: 0 21 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" collect_table_stats
Crontab maintenance scripts [6]: 4 */8 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/operation.py" vacuum_analyze_pg_catalog
Crontab maintenance scripts [7]: 0 1 * * * . "$PROFILE" && "$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.sh" >> "$PATH_ARENADAT...
Delete old maintenance script logs: 30

Save changes in the ADB configuration

7. After the *operation.py* script is run according to the schedule, view the contents of the log created in the */home/gpadmin/<Arenadata_configs_directory_name>/operation_log/* directory (by default, */home/gpadmin/arenadata_configs/operation_log/*). Each log name contains date and time of the script launch — for example, *2025-11-10_10:05:01.172592.log*.
   Ensure that the table processing order corresponds to the one defined in the custom function. To do this, pay attention to the log entries marked as `DO`, which mean that the `VACUUM` command is started for the corresponding tables.

   Note that the order of events with the same timestamp in the log may differ from the actual one. In this case, compare identifiers of worker processes `ForkPoolWorker`. The smaller the identifier, the earlier the start of processing.

Log fragment for the VACUUM operation

```
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-3: pid=3437270: DO: 'vacuum "arenadata_toolkit"."daily_operation_1_prt_p202511";' for 'adb'
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-7: pid=3437279: DO: 'vacuum "arenadata_toolkit"."db_files_history_1_prt_p202511";' for 'adb'
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-2: pid=3437269: DO: 'vacuum "arenadata_toolkit"."daily_operation_1_prt_p202510";' for 'adb'
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-1: pid=3437268: DO: 'vacuum "arenadata_toolkit"."daily_operation_1_prt_other";' for 'adb'
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-4: pid=3437272: DO: 'vacuum "arenadata_toolkit"."db_files_current";' for 'adb'
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-5: pid=3437275: DO: 'vacuum "arenadata_toolkit"."db_files_history_1_prt_other";' for 'adb'
2025-11-10 08:25:02,667: INFO: ForkPoolWorker-8: pid=3437281: DO: 'vacuum "arenadata_toolkit"."operation_exclude";' for 'adb'
2025-11-10 08:25:02,669: INFO: ForkPoolWorker-9: pid=3437283: DO: 'vacuum "kadb"."offsets";' for 'adb'
2025-11-10 08:25:02,665: INFO: ForkPoolWorker-6: pid=3437277: DO: 'vacuum "arenadata_toolkit"."db_files_history_1_prt_p202510";' for 'adb'
2025-11-10 08:25:02,988: INFO: ForkPoolWorker-4: pid=3437272: DONE: 'vacuum "arenadata_toolkit"."db_files_current";' for 'adb'
2025-11-10 08:25:02,990: INFO: ForkPoolWorker-9: pid=3437283: DONE: 'vacuum "kadb"."offsets";' for 'adb'
2025-11-10 08:25:03,090: INFO: ForkPoolWorker-5: pid=3437275: DONE: 'vacuum "arenadata_toolkit"."db_files_history_1_prt_other";' for 'adb'
2025-11-10 08:25:03,092: INFO: ForkPoolWorker-8: pid=3437281: DONE: 'vacuum "arenadata_toolkit"."operation_exclude";' for 'adb'
2025-11-10 08:25:03,186: INFO: ForkPoolWorker-6: pid=3437277: DONE: 'vacuum "arenadata_toolkit"."db_files_history_1_prt_p202510";' for 'adb'
2025-11-10 08:25:03,186: INFO: ForkPoolWorker-2: pid=3437269: DONE: 'vacuum "arenadata_toolkit"."daily_operation_1_prt_p202510";' for 'adb'
2025-11-10 08:25:03,196: INFO: ForkPoolWorker-1: pid=3437268: DONE: 'vacuum "arenadata_toolkit"."daily_operation_1_prt_other";' for 'adb'
2025-11-10 08:25:03,398: INFO: ForkPoolWorker-3: pid=3437270: DONE: 'vacuum "arenadata_toolkit"."daily_operation_1_prt_p202511";' for 'adb'
2025-11-10 08:25:03,431: INFO: ForkPoolWorker-7: pid=3437279: DONE: 'vacuum "arenadata_toolkit"."db_files_history_1_prt_p202511";' for 'adb'
2025-11-10 08:25:04,020: INFO: ForkPoolWorker-4: pid=3437272: DO: 'vacuum "public"."spatial_ref_sys";' for 'adb'
2025-11-10 08:25:04,123: INFO: ForkPoolWorker-10: pid=3437285: DO: 'vacuum "public"."etl_staging_main";' for 'adb'
2025-11-10 08:25:04,123: INFO: ForkPoolWorker-8: pid=3437281: DO: 'vacuum "public"."etl_staging_orders";' for 'adb'
2025-11-10 08:25:04,124: INFO: ForkPoolWorker-2: pid=3437269: DO: 'vacuum "public"."etl_staging_products";' for 'adb'
2025-11-10 08:25:04,129: INFO: ForkPoolWorker-5: pid=3437275: DO: 'vacuum "public"."etl_staging_users";' for 'adb'
2025-11-10 08:25:04,326: INFO: ForkPoolWorker-4: pid=3437272: DONE: 'vacuum "public"."spatial_ref_sys";' for 'adb'
2025-11-10 08:25:04,424: INFO: ForkPoolWorker-9: pid=3437283: DONE: 'vacuum "public"."etl_temp_processing";' for 'adb'
2025-11-10 08:25:04,526: INFO: ForkPoolWorker-2: pid=3437269: DONE: 'vacuum "public"."etl_staging_products";' for 'adb'
2025-11-10 08:25:04,526: INFO: ForkPoolWorker-5: pid=3437275: DONE: 'vacuum "public"."etl_staging_users";' for 'adb'
2025-11-10 08:25:04,641: INFO: ForkPoolWorker-8: pid=3437281: DONE: 'vacuum "public"."etl_staging_orders";' for 'adb'
2025-11-10 08:25:04,727: INFO: ForkPoolWorker-10: pid=3437285: DONE: 'vacuum "public"."etl_staging_main";' for 'adb'
```

> **IMPORTANT**
>
> - If there is no selected function in some databases, the following message is written to the log: `This function <schema_name>.<function_name> does not exist in the database`.
> - If the `--vacuum-order` argument has an invalid value (other than `newest-first` / `newest-last` and not containing a point), the following message is written to the log: `--vacuum-order parameter is only valid for vacuum action or invalid function name`.
>
> In both cases, the `VACUUM` operation is not applied to the databases for which errors have been detected.

Manage spill files

Daria Barysheva

Contents

To Table of Contents

Overview
Analyze spill files usage
Manage spill files

Overview

Spill files (also known as **work files**) are created on disks if there is not enough memory to store temporary data when executing data queries. This temporary data may include `JOIN` hashes, sorting results, redistributed tables, etc. The directory for storing spill files can be defined via the server configuration parameter temp_tablespaces.

Since spill files are stored on the hard disk and the speed of reading/writing data from disks is several times slower than from RAM, the total execution time of queries grows. It can also increase the queues for resources from other users. Ideally, there should be no queries that generate spill files. However, a small number of spill files is allowed. In fact, spill files represent a flexible method of executing a query despite the limited amount of RAM. But you should avoid situations when there are too many spill files.

There are some typical cases when a large number of spill files can be generated:

- **Data skew is present in the queried data**. To see how to avoid it, refer to Recommendations on avoiding data skew.
- **The amount of memory allocated for the query is too low**. You can control the maximum memory size to be used by a query with the server configuration parameters `max_statement_mem` and `statement_mem`. You can also configure resource groups.
- **The query is not optimized**:
  - **DISTINCT is used on a large number of fields**. Every unique combination of columns specified in the `DISTINCT` clause is added to RAM. Accordingly, the more such combinations are detected, the faster the memory fills up and spill files are generated. It is recommended to limit the number of requested columns and avoid using huge size columns in the `DISTINCT` clause.
  - **ORDER BY is used**. Intermediate sorting results are initially stored in RAM. The larger the amount of data is to be sorted, the faster the generation of spill files begins. In Greenplum, it is not recommended to use sorting in queries, especially for large tables. Instead, it is better to sort the results directly in the data receiver (on the client side).
  - **A huge amount of data is processed without filters in the WHERE clause**. When performing `JOIN` queries, data from one of the tables is added to a hash table in RAM. Accordingly, if the requested data of this table is not filtered, the memory will overflow very quickly. It is recommended to use filters whenever possible, especially for dates and timestamps. Even if you need to get results for the entire period, it is desirable to form several queries in short time intervals, after which you can combine the results using `UNION ALL`. This approach is more effective than trying to process the data in its entirety.
  - **When executing JOIN queries, a hash table is built on a larger of several tables**. When executing `INNER JOIN`, the hash table mentioned above is built on the basis of the table with the smallest size (in order to minimize RAM usage). However, when you use `LEFT JOIN` — the table defined on the right side is always taken as the basis for the hash table, regardless of its volume. It can lead to problems if this table is the largest in the pair. Therefore, pay attention to the order of all tables in data queries. Also try to minimize the number of extracted columns.

TIP

- Always try to reduce the number of spill files by changing queries, distribution policy, or memory configuration.
- If these methods do not help, you can change some server configuration parameters to limit the number of spill files and the disk space they use.

Analyze spill files usage

The default schema `gp_toolkit` contains views that show information about all the queries that are currently using spill files. These views can be useful for analyzing queries and for troubleshooting:

- `gp_workfile_entries` — shows query operators that use disk space for spill files on a segment at the current time. A single row corresponds to a single query operator.

  Structure

Column	Description
datname	The database name
pid	Process ID of the server process
sess_id	Session ID
command_cnt	Command ID of the query
usename	The role name
query	The current query being running by the process
segid	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables → gp_segment_configuration*** section of the Tables overview article
slice	The query plan slice. Shows the portion of the query plan that is being run
optype	The type of the query operator that created the spill file
size	Spill files size (in bytes)
numfiles	The number of spill files created
prefix	The prefix used when naming a related spill files set

  Example:

```sql
SELECT * FROM gp_toolkit.gp_workfile_entries;
```

  Result:

```
 datname | pid  | sess_id | command_cnt | usename |                    query                    | segid | slice | optype |  size
| numfiles | prefix
---------+------+---------+-------------+---------+---------------------------------------------+-------+-------+--------+-------
--+----------+--------
 adb     | 5332 |      65 |          42 | gpadmin | SELECT DISTINCT(id1,id2,gen1,gen2) from t1; |    -1 |     0 | Sort   |
4292608 |        1 | Sort_1
(1 row)
```

- `gp_workfile_usage_per_query` — shows queries that use disk space for spill files on a segment at the current time. A single row corresponds to a single query.

  Structure

Column	Description
datname	The database name
pid	Process ID of the server process
sess_id	Session ID
command_cnt	Command ID of the query
usename	The role name
query	The current query being running by the process
segid	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables → gp_segment_configuration*** section of the Tables overview article
size	Spill files size (in bytes)
numfiles	The number of spill files created

  Example:

```sql
SELECT * FROM gp_toolkit.gp_workfile_usage_per_query;
```

  Result:

```
 datname | pid  | sess_id | command_cnt | usename |                    query                    | segid |   size   | numfiles
---------+------+---------+-------------+---------+---------------------------------------------+-------+----------+----------
 adb     | 5332 |      65 |          42 | gpadmin | SELECT DISTINCT(id1,id2,gen1,gen2) from t1; |    -1 | 20807680 |        1
(1 row)
```

- `gp_workfile_usage_per_segment` — shows the total size of disk space used for spill files on each segment at the current time. A single row corresponds to a single segment.

  Structure

Column	Description
segid	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables → gp_segment_configuration*** section of the Tables overview article
size	Spill files size (in bytes)
numfiles	The number of spill files created

  Example:

```sql
SELECT * FROM gp_toolkit.gp_workfile_usage_per_segment;
```

  Result:

```
 segid |   size   | numfiles
-------+----------+----------
     5 |        0 |
    13 |        0 |
    10 |        0 |
    11 |        0 |
     1 |        0 |
     6 |        0 |
    12 |        0 |
     0 |        0 |
     2 |        0 |
     7 |        0 |
    15 |        0 |
     4 |        0 |
     8 |        0 |
     3 |        0 |
    -1 | 31752192 |        1
    14 |        0 |
     9 |        0 |
(17 rows)
```

NOTE

The views listed above are accessible to all users. However, non-superusers can get information only on the databases that they have permission to access.

Manage spill files

To limit the number of spill files and the disk space consumed by them, you can use several configuration parameters. Their description is given below.

Name	Description	Default value	Type
gp_workfile_limit_files_per_query	The maximum number of spill files allowed per query at each segment. If the limit is reached, the following error occurs: `ERROR: number of workfiles per query limit exceeded.` The `0` value allows an unlimited number of spill files	100000	coordinator session reload
gp_workfile_limit_per_query	The maximum disk size each query is allowed to use for spill files at each segment (in KB). If the limit is reached, the following error occurs: `ERROR: workfile per query size limit exceeded.` The `0` value means no limit	25000000	coordinator session reload
gp_workfile_limit_per_segment	The maximum total disk size that all running queries are allowed to use for spill files at each segment (in KB). If the limit is reached, the following error occurs: `ERROR: workfile per segment size limit exceeded.` The `0` value means no limit	50000000	local system restart
gp_workfile_compression	Specifies whether to use compression for spill files when a hash aggregation or hash join operation spills to disk during query processing. The default value `off` means no compression is used. Setting the value to `on` enables compression	off	coordinator session reload

IMPORTANT

- The default value of the `gp_workfile_limit_files_per_query` parameter (`100000`) is sufficient for the majority of queries. Before raising it, try to optimize your queries and distribution/memory options.
- Limiting the number or size of spill files prevents queries from disrupting the DBMS. But it also leads to termination of your queries when the limits are reached.
- Enabling compression via the `gp_workfile_compression` parameter can improve performance of the queries that generate spill files. But remember that data compression reduces the number of I/O operations at the expense of increased CPU usage.

Delete orphaned files in ADB

Daria Barysheva

Contents

To Table of Contents

Overview

Configuration

Overview

Orphaned files are data files that are not related to any database table. They can remain in the file system after a crash of some backend process associated with a transaction that creates a new table. Possible crash causes include hardware failures, lack of free disk space, and so on. As a result of successful recovery after failure, uncommitted changes are missing in the database (the new table does not exist in the system catalog), but data files of that table may remain on segment hosts. Over time, such abandoned files may increase in number, wasting valuable storage space. The process of searching for and deleting such files is time-consuming, and there are always risks associated with removing valuable data.

Starting with 6.29.0.3 in ADB 6 and 7.4.0 in ADB 7, DBMS supports automatic tracking and deletion of orphaned files at the DBMS core level in case of the transaction interruption. In previous ADB versions, data files were automatically deleted for those transactions that were aborted normally (via `ABORT`), but orphaned files could be deleted only manually (for example, by searching them via functions and views of the gp_check_functions extension).

The new approach to tracking and deleting orphaned files that is implemented in the ADB versions mentioned above is based on the following key concepts:

- In case of a commit or rollback of a transaction that creates a table, the DBMS core, like before, independently determines how to process the table data files in normal mode. In case of `ABORT`, files are deleted; in case of `COMMIT PREPARED` they remain.
- Only transactions in the `TRANSACTION_STATUS_IN_PROGRESS` status are considered as candidates for deleting orphaned files. The DBMS core suggests that transactions in that status can abort abnormally and leave orphaned files. Transactions in all other statuses are not considered, as they are processed in the standard way (see the previous point).
- The decision to delete a certain orphaned file is automatically made at the recovery startup.

IMPORTANT

- For more information on the causes of orphaned files, approaches to solving that problem in other DBMS, and implementation details from the ADB/Greengage DB team, see Deleting orphaned files in the technical blog.
- Remember that installation of the ADB versions mentioned above does not allow you to delete already existing orphaned files, but it will prevent them from being produced in the future.

Configuration

You can manage tracking and deleting orphaned files for interrupted transactions using the `gp_track_pending_delete` configuration parameter (GUC). The default parameter value is `on`, which means using the new approach implemented in ADB 6.29.0.3 and 7.4.0. If for any reason you need to disable that approach, change the parameter value to `off`. Note that the `gp_track_pending_delete` parameter value can only be modified by editing the *postgresql.conf* file on all segments using the `gpconfig` utility; and you need to restart ADB to apply your changes (reload of the configuration file is not sufficient).

The following is an example of how to change the `gp_track_pending_delete` value:

1. Connect to the ADB coordinator host and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Get the current parameter value:

```
$ gpconfig -s gp_track_pending_delete
```

Result:

```
Values on all segments are consistent
GUC              : gp_track_pending_delete
Coordinator value: on
Segment     value: on
```

3. Change the parameter value on all segments:

```
$ gpconfig -c gp_track_pending_delete -v off
```

Result:

```
20250827:10:10:29:2622827 gpconfig:bds-mdw:gpadmin-[INFO]:-completed successfully with parameters '-c gp_track_pending_delete -v off'
```

4. Restart DBMS:

```
$ gpstop -r -a
```

5. Check the current parameter value again to ensure that your changes have been applied:

```
$ gpconfig -s gp_track_pending_delete
```

Command output:

```
Values on all segments are consistent
GUC              : gp_track_pending_delete
Coordinator value: off
Segment     value: off
```

Databases

Daria Barysheva

Contents

To Table of Contents

Overview

View the list of databases

- Way 1 — psql
- Way 2 — SQL query

Create a new database

Overview

Database is a collection of data physically stored together. Every ADB cluster can contain multiple databases. However, client programs can connect to only one database at a time — cross-queries between different databases are impossible since the databases are physically separated.

There are several template and default databases in every ADB cluster. These databases are added after the DBMS is initialized. See their descriptions in the table below.

Template and default databases

Name	Description
adb	This database can be used as default for client connections. All necessary database objects can be added to this database. It is created via ADCM — by applying the Create database ADB service action. However, you can also add other custom databases. Notice that you can change the name of the default database before applying the ***Create database*** action — via configuration parameters of the ADB service (see the ***Name of default database*** parameter in the ***ADB*** → ***Main*** section)
diskquota	This database is used by the `diskquota` ADB extension. The extension allows users to limit the disk space used by schemas and roles. The database is created automatically if you install ADB with the ***Diskquota Extension*** configuration parameter set to `true` or when you invoke the Install diskquota service action in ADB
postgres	This database is usually used as default for administrative connections. For example, it can be used by startup processes, the Global Deadlock Detector process, etc.
template0	This database template is used by the DBMS internally. You can use `template0` to create a completely clean database that contains only the standard objects predefined by DBMS at initialization. Do not drop or modify the `template0` database
template1	This database is used as a template for each new database you create (via the `CREATE DATABASE` command) — unless you specify another template. Creating objects in `template1` is not recommended

CAUTION

It is not recommended to drop or modify the databases listed above.

View the list of databases

To view the list of available databases in the current ADB cluster, you can use several methods. Two of them are listed below.

Way 1 — psql

If you work in the `psql` client program, you can use the `\l` meta-command to show the list of databases:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. List all ADB databases:

```
$ psql -l
```

The result can look like this:

```
                                  List of databases
   Name    |  Owner  | Encoding |  Collate   |   Ctype    |  Access privileges
-----------+---------+----------+------------+------------+---------------------
 adb       | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =Tc/gpadmin        +
           |         |          |            |            | gpadmin=CTc/gpadmin
 diskquota | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres  | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +
           |         |          |            |            | gpadmin=CTc/gpadmin
 template1 | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +
           |         |          |            |            | gpadmin=CTc/gpadmin
(5 rows)
```

Way 2 — SQL query

After connecting to the specified database via `psql` (or any other client program), you can also run the special SQL query that gets the list of databases from the `pg_catalog.pg_database` system catalog table. To do this, run the following commands:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

The output is:

```
psql (12.12)
Type "help" for help.
```

3. Run the following SQL query to list all ADB databases:

```
SELECT * FROM pg_catalog.pg_database ORDER BY datname;
```

The result can look like this:

```
  datname  | datdba | encoding | datcollate |  datctype  | datistemplate | datallowconn | datconnlimit | datlastsysoid |
datfrozenxid | datminmxid | dattablespace |               datacl
-----------+--------+----------+------------+------------+---------------+--------------+--------------+---------------+---------
-----+------------+---------------+-----------------------------------
 adb       |     10 |        6 | en_US.utf8 | en_US.utf8 | f             | t            |           -1 |         12809 |
702 |          1 |          1663 | {=Tc/gpadmin,gpadmin=CTc/gpadmin}
 diskquota |     10 |        6 | en_US.utf8 | en_US.utf8 | f             | t            |           -1 |         12809 |
948 |          1 |          1663 |
 postgres  |     10 |        6 | en_US.utf8 | en_US.utf8 | t             | t            |           -1 |         12809 |
1881 |          1 |          1663 |
 template0 |     10 |        6 | en_US.utf8 | en_US.utf8 | t             | f            |           -1 |         12809 |
702 |          1 |          1663 | {=c/gpadmin,gpadmin=CTc/gpadmin}
 template1 |     10 |        6 | en_US.utf8 | en_US.utf8 | t             | t            |           -1 |         12809 |
1799 |          1 |          1663 | {=c/gpadmin,gpadmin=CTc/gpadmin}
(5 rows)
```

Create a new database

To create a new database, it is necessary to use the `CREATE DATABASE` command followed by the database name.

The following example creates the new database `books_store`:

```
CREATE DATABASE books_store;
```

The output should be similar to:

```
CREATE DATABASE
```

To disconnect from the current database and switch to the new one, you can use the `psql` command `\c`:

```
\c books_store
```

Result:

```
You are now connected to database "books_store" as user "gpadmin".
books_store=#
```

NOTE

To create a database, you should have the correct privileges for this operation or be a superuser. If you do not have such privileges, please, contact your database administrator to either give you the necessary rights or to create a database for you.

Schemas

Daria Barysheva

Contents

To Table of Contents

Overview

View the list of schemas

Create a new schema

Schema search path

  Change the search_path parameter

Overview

Schema is a logical union of objects within a single database. In comparison with databases, schemas are not rigidly separated: users can access multiple schemas at a time (if they have the necessary privileges). You can use schemas for the following purposes:

- To allow many users to access one database without conflicts.
- To organize database objects into different logical groups to make them more manageable. It can be especially helpful while configuring access privileges.
- To put the third-party applications into separate schemas so they do not collide with the names of other schema objects.

TIP

Remember that schemas allow you to have more than one object (such as tables) with the same name in one database without conflicts — if these objects belong to different schemas.

The following table describes the system-level schemas that are added to the ADB database by default.

Default schemas

Name	Description
arenadata_toolkit	Stores ADB settings and information about the distribution of tables and operations performed
diskquota	It is used by the `diskquota` ADB extension. This extension allows users to limit the disk space used by schemas and roles. The schema is created automatically if you install ADB with the ***Diskquota Extension*** configuration parameter set to `true` or when you invoke the Install diskquota service action in ADB
gp_toolkit	It is an administrative schema that contains external tables, views, and functions that you can access with SQL commands — to view and query the system log files and other system metrics (including table sizes, bloat, indexes, resource queues, spill files, etc.)
information_schema	Consists of a standardized set of views that contain information about the database objects. These views get system information from the system catalog tables in a standardized way (according to the ANSI SQL 2008 standard)
kadb	It is used in Kafka to ADB Connector
madlib	Stores the Apache MADlib library functions. The schema is created if you install ADB with the ***ADB MADlib Extension*** configuration parameter set to `true` or when you invoke the Install MADlib service action in ADB
pg_aoseg	Stores append-optimized table objects. This schema is used internally by the DBMS
pg_bitmapindex	Stores bitmap index objects such as lists of values. This schema is used internally by the DBMS
pg_catalog	Contains the full information about DB objects: the system catalog tables, built-in data types, functions, and operators
pg_toast	Stores large objects such as records that exceed the page size (Oversized-Attribute Storage Technique, TOAST). This schema is used internally by the DBMS
public	It is a default schema that is used for each new database object you create — unless you explicitly specify another schema (or change the default schema search path). All database roles (users) have the `CREATE` and `USAGE` privileges in the `public` schema

CAUTION

It is not recommended to drop or modify the schemas listed above.

View the list of schemas

To view the list of available schemas in the current database, do the following:

1. Connect to the ADB coordinator host (for example, via SSH) and log in under the default user `gpadmin`:

```
$ sudo su - gpadmin
```

2. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

   The output is:

```
psql (12.12)
Type "help" for help.
```

3. Run one of the following commands:

   - Get information from the `pg_catalog.pg_namespace` table. It contains all database schemas.

```
SELECT * FROM pg_catalog.pg_namespace ORDER BY nspname;
```

     The output is:

```
      nspname       | nspowner |               nspacl
--------------------+----------+-----------------------------------
 arenadata_toolkit  |       10 | {gpadmin=UC/gpadmin}
 diskquota          |       10 |
 gp_toolkit         |       10 | {gpadmin=UC/gpadmin,=U/gpadmin}
 information_schema |       10 | {gpadmin=UC/gpadmin,=U/gpadmin}
 kadb               |       10 |
 madlib             |       10 |
 pg_aoseg           |       10 |
 pg_bitmapindex     |       10 |
 pg_catalog         |       10 | {gpadmin=UC/gpadmin,=U/gpadmin}
 pg_toast           |       10 |
 public             |       10 | {gpadmin=UC/gpadmin,=UC/gpadmin}
(11 rows)
```

   - Get data from the `information_schema.schemata` table. Compared to `pg_catalog.pg_namespace`, this table stores information about the database schemas according to the ANSI SQL 2008 standard.

```
SELECT * FROM information_schema.schemata ORDER BY schema_name;
```

     The output is:

```
 catalog_name |    schema_name     | schema_owner | default_character_set_catalog | default_character_set_schema |
default_character_set_name | sql_path
--------------+--------------------+--------------+-------------------------------+------------------------------+-----------
-----------------+----------
 adb          | arenadata_toolkit  | gpadmin      |                               |                              |
|
 adb          | diskquota          | gpadmin      |                               |                              |
|
 adb          | gp_toolkit         | gpadmin      |                               |                              |
|
 adb          | information_schema | gpadmin      |                               |                              |
|
 adb          | kadb               | gpadmin      |                               |                              |
|
 adb          | madlib             | gpadmin      |                               |                              |
|
 adb          | pg_aoseg           | gpadmin      |                               |                              |
|
 adb          | pg_bitmapindex     | gpadmin      |                               |                              |
|
 adb          | pg_catalog         | gpadmin      |                               |                              |
|
 adb          | pg_toast           | gpadmin      |                               |                              |
|
 adb          | public             | gpadmin      |                               |                              |
|
(11 rows)
```

   - If you work in the `psql` client program, you can use the `\dnS+` meta-command to show the list of schemas.

```
\dnS+
```

     The output is:

```
                                                 List of schemas
        Name        |  Owner  | Access privileges  |                          Description
--------------------+---------+--------------------+-------------------------------------------------------------
 arenadata_toolkit  | gpadmin | gpadmin=UC/gpadmin |
 diskquota          | gpadmin |                    |
 gp_toolkit         | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 information_schema | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 kadb               | gpadmin |                    |
 madlib             | gpadmin |                    |
 pg_aoseg           | gpadmin |                    | Reserved schema for Append Only segment list and eof tables
 pg_bitmapindex     | gpadmin |                    | Reserved schema for internal relations of bitmap indexes
 pg_catalog         | gpadmin | gpadmin=UC/gpadmin+| system catalog schema
                    |         | =U/gpadmin         |
 pg_toast           | gpadmin |                    | reserved schema for TOAST tables
 public             | gpadmin | gpadmin=UC/gpadmin+| standard public schema
                    |         | =UC/gpadmin        |
(11 rows)
```

Create a new schema

To create a new schema, it is necessary to use the `CREATE SCHEMA` command followed by the schema name.

The following example creates the new schema `test_schema` (in the current database `adb`).

```
CREATE SCHEMA test_schema;
```

The output should be similar to:

```
CREATE SCHEMA
```

Now, if you run any command to list the database schemas, you get the following:

```
\dnS+
```

```
                                                 List of schemas
        Name        |  Owner  | Access privileges  |                          Description
--------------------+---------+--------------------+-------------------------------------------------------------
 arenadata_toolkit  | gpadmin | gpadmin=UC/gpadmin |
 diskquota          | gpadmin |                    |
 gp_toolkit         | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 information_schema | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 kadb               | gpadmin |                    |
 madlib             | gpadmin |                    |
 pg_aoseg           | gpadmin |                    | Reserved schema for Append Only segment list and eof tables
 pg_bitmapindex     | gpadmin |                    | Reserved schema for internal relations of bitmap indexes
 pg_catalog         | gpadmin | gpadmin=UC/gpadmin+| system catalog schema
                    |         | =U/gpadmin         |
 pg_toast           | gpadmin |                    | reserved schema for TOAST tables
 public             | gpadmin | gpadmin=UC/gpadmin+| standard public schema
                    |         | =UC/gpadmin        |
 test_schema        | gpadmin |                    |
(12 rows)
```

Schema search path

To access database objects, you can use two main forms of object names: the ANSI standard `database.schema.object` or the shorter variant `schema.object`. The second form is preferable since you cannot access more than one database at a time anyway. But there is also the third option that allows you to omit even the schema names (and use only object names, e.g. table names). It is possible due to using the **schema search path**.

The schema search path is configured via the `search_path` parameter. This parameter specifies the order of searching the available schemas for objects. In fact, it contains the ordered comma-separated list of schemas. The requested object is searched in the first schema of the search path, in case of failure — in the second one, and so on. If there is no match in the search path, an error is reported, even if matching object names exist in other schemas in the database. The schema listed first in the search path becomes the **default** schema (it is also called the **current** schema). Aside from being the first schema searched, it is also the schema in which new objects are created if no schema name is explicitly defined.

To show the current schema search path, you can use the following command:

```
SHOW search_path;
```

In the default setup this command returns:

```
   search_path
-----------------
 "$user", public
(1 row)
```

The first element `"$user"` specifies that a schema with the same name as the current user is to be searched. If no such schema exists, the entry is ignored. The second element refers to the `public` schema. It explains why the `public` schema is used for new database objects by default. The next query returns the default (or current) schema:

```
SELECT current_schema();
```

In the default setup the result is:

```
 current_schema
----------------
 public
(1 row)
```

NOTE

The `pg_catalog` schema is always included in the schema search path, even if this schema is not explicitly named in the search path.

Change the search_path parameter

Changing the `search_path` parameter is possible at the database level (via the `ALTER DATABASE` command) and the role level (via the `ALTER ROLE` command). Let's consider the first method using a simple example:

1. Set the `search_path` parameter at the database level.

```
ALTER DATABASE adb SET search_path TO test_schema,public;
```

   The result should be similar to:

```
ALTER DATABASE
```

2. Log out from `psql` (or any other client program you use) to reopen the client session.

```
\q
```

3. Connect to the `adb` database again.

```
$ psql adb
```

   The output is:

```
psql (12.12)
Type "help" for help.
```

4. Create any table without assigning a schema name.

```
CREATE TABLE test(id1 INT) DISTRIBUTED BY(id1);
```

   Result:

```
CREATE TABLE
```

5. Check the schema that is used for the `test` table.

```
\dt test
```

   It should be equal to `test_schema`, not `public`.

```
              List of relations
   Schema    | Name | Type  |  Owner  | Storage
-------------+------+-------+---------+---------
 test_schema | test | table | gpadmin | heap
(1 row)
```

6. Return the previous value of the `search_path` parameter and restart the client session again.

```
ALTER DATABASE adb SET search_path TO "$user",public;
\q
```

NOTE

It is also possible to change the `search_path` parameter value at the session level — via the `SET search_path TO <new_path_value>;` command (where `<new_path_value>` is a new value of the parameter, e.g. `test_schema,public`). But it is strongly recommended to set the parameter at the database or role level, so that it does not reset on each session restart.

Tables overview

Daria Barysheva, Anton Vasilev

ADB

Contents

To Table of Contents

- Create a new table
  - Syntax
  - Recommendations on choosing column data types
  - Examples
- View the list of tables
- View the specified table
- System catalog tables
  - gp_configuration_history
  - gp_segment_configuration
  - pg_database
  - pg_class
  - pg_namespace
  - pg_tablespace
- Useful queries

Table is the main data object in ADB. ADB tables are similar to those in any relational database, except they involve distribution of table rows across different cluster segments. When creating a table, you specify the table distribution policy.

Also, ADB provides several table storage types and compression options that can be used to store data in the most efficient way.

Create a new table

Syntax

To create an initially empty table in the current database and define its structure, use the `CREATE TABLE` command. The user who issues the command owns the table. The basic syntax of this command is described below.

NOTE

For more details, see CREATE TABLE in the Greenplum documentation.

```
CREATE [ { TEMPORARY | TEMP } | UNLOGGED] TABLE [IF NOT EXISTS]
  <table_name> (
  [ { <column_name> <data_type> [ COLLATE <collation> ] [<column_constraint> [ ... ] ] [ ENCODING ( <storage_column_directive> [,
...] ) ]
    | <table_constraint>
    | LIKE <source_table> [ <like_option> ... ] }
    [, ... ]
  ])
[ INHERITS ( <parent_table> [, ... ] ) ]
[ WITH ( <storage_table_directive> [, ... ] ) ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE <tablespace_name> ]
[ USING <access_method> ]
[ DISTRIBUTED BY (<column_name> [<opclass>], [ ... ] )
      | DISTRIBUTED RANDOMLY | DISTRIBUTED REPLICATED ]
[ PARTITION BY <partition_type> (<column_name>) <partition_specification>]
```

Basic clauses

TEMPORARY | TEMP	Specifies a **temporary** table, which is automatically dropped at the end of a session, or optionally at the end of the current transaction (see `ON COMMIT` below)
UNLOGGED	Specifies an **unlogged** table, whose data is not written to the write-ahead log (WAL). This makes such tables faster than ordinary tables, yet not crash-safe. The contents of unlogged tables are not replicated to mirror segment instances
COLLATE	Defines the column collation. If not specified, the column data type's default collation is used
ENCODING	Defines the column storage options: type of compression, compression level, and block size. This clause is valid only for append-optimized column-oriented tables
LIKE	Specifies a table from which the new table automatically copies all column names, their data types, `NOT NULL`-constraints. Other column definition parts can be included or excluded via the `INCLUDING` / `EXCLUDING` statements (see the `<like_option>` argument below). Unlike `INHERITS`, the new table and original table are completely decoupled after creation is complete. It means that changing the original table structure (via `ALTER TABLE`) has no impact on the new table. Pay attention that storage properties specified in the `WITH` clause and partition structure are not copied
INHERITS	Specifies a list of tables from which the new table automatically inherits all columns and their constraints. Compared to the `LIKE` clause, use of `INHERITS` creates a persistent relationship between the new child table and its parent table. All modifications applied to the parent table structure normally propagate to the child one as well
WITH	Defines storage parameters for the table
ON COMMIT	Specifies the behavior of temporary tables at the end of a transaction block. Possible values: `PRESERVE ROWS` — no special action is taken. This is the default behavior. `DELETE ROWS` — all rows in the temporary table should be deleted at the end of the current transaction. Essentially, an automatic `TRUNCATE` is done. `DROP` — the temporary table should be dropped at the end of the current transaction.
TABLESPACE	Defines the name of the tablespace in which the new table is to be created. If not specified, the database's default tablespace is used, or the server configuration parameter `temp_tablespaces` (in case of a temporary table)
USING	Defines the table access method to use to store the contents. For tables, possible values are: `heap` — creates a heap table. `ao_row` — creates an append-optimized table with row-oriented storage. `ao_column` — creates an append-optimized table with column-oriented storage. The default value is `heap`, which can be changed using the `default_table_access_method` server configuration parameter. Prior to ADB 7, this setting was configured using the `WITH (appendoptimized=true|false, orientation=row|column)` clause, and this old syntax remains functional
DISTRIBUTED BY	Sets the table distribution policy. Possible values: `DISTRIBUTED BY (<column_name> [<opclass>], [ … ] )` — hash distribution. `DISTRIBUTED RANDOMLY` — random distribution that is based on the round-robin algorithm. `DISTRIBUTED REPLICATED` — full distribution (the table copy is stored on every cluster segment). For more details, see Distribution
PARTITION BY	Specifies a partitioned table. For more details, see Partitioning

Arguments

<table_name>	A table name
<column_name>	A column name
<data_type>	A column data type. See Recommendations on choosing column data types below
<collation>	A column collation
<column_constraint>	A constraint defined at the column level. The `<constraint_name>` name is given optionally. Possible constraints: `NOT NULL` — specifies that the column is not allowed to contain null values. `NULL` — specifies that the column is allowed to contain null values. It is the default behavior (unless `NOT NULL` is specified). `CHECK` — specifies the boolean expression `<expression>` which new or updated table rows should satisfy for an `INSERT` or `UPDATE` operation to succeed. A constraint marked as `NO INHERIT` will not propagate to child tables (created via `INHERITS`). `DEFAULT` — sets the default column value equal to `<default_expr>`. `GENERATED ALWAYS AS` — creates a generated column whose value is automatically computed and cannot be updated directly. `GENERATED AS IDENTITY` — creates an identity column that automatically generates unique, sequential values. `UNIQUE` — specifies that the column should contain only unique values. Null values are not considered equal. `PRIMARY KEY` — specifies that the column should contain unique (non-duplicate) non-null values. Only one primary key can be specified for a table (it can include several columns). Referential integrity constraints (`FOREIGN KEY`, `REFERENCES`) are also allowed, but not enforced. There are two groups of constraints: `DEFERRABLE | NOT DEFERRABLE`. A constraint that is **not deferrable** will be checked immediately after every command. Checking of **deferrable** constraints can be postponed until the end of the transaction (using the SET CONSTRAINTS command). `NOT DEFERRABLE` is the default. Currently, only `UNIQUE` and `PRIMARY KEY` constraints can be deferrable. `NOT NULL` and `CHECK` constraints are not deferrable. `INITIALLY DEFERRED | INITIALLY IMMEDIATE`. If a constraint is **deferrable**, this clause specifies the default time to check the constraint. If the constraint is `INITIALLY IMMEDIATE`, it is checked after each statement. This is the default behavior. If the constraint is `INITIALLY DEFERRED`, it is checked only at the end of the transaction. The constraint check time can be altered with the SET CONSTRAINTS command. For indexes associated with a `UNIQUE` or `PRIMARY KEY` constraint you can define a tablespace and storage options as well as for ordinary tables (see `<index_parameters>` below). Syntax:

```
[ CONSTRAINT <constraint_name>]
{ NOT NULL
  | NULL
  | CHECK  ( <expression> ) [ NO INHERIT ]
  | DEFAULT <default_expr>
  | UNIQUE <index_parameters>
  | PRIMARY KEY <index_parameters>
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

Where `<index_parameters>` is:

```
[ INCLUDE ( <column_name> [, ... ] ) ]
[ WITH ( <storage_parameter> [=<value>] [, ... ] ) ]
[ USING INDEX TABLESPACE <tablespace_name> ]
```

<storage_column_directive>	Specifies the column storage settings for append-optimized column-oriented tables: `compresstype` — compression type. `compresslevel` — compression level. `blocksize` — block size (in bytes). Syntax:

```
compresstype={ZLIB|ZSTD|RLE_TYPE|NONE}
[compresslevel={0-9}]
[blocksize={8192-2097152} ]
```

For more details on compression options, see Data compression

<table_constraint>	A constraint defined at the table level. The `<constraint_name>` name is given optionally. Possible constraints: `CHECK` — specifies the boolean expression `<expression>` which new or updated table rows should satisfy for an `INSERT` or `UPDATE` operation to succeed. A constraint marked as `NO INHERIT` will not propagate to child tables (created via `INHERITS`). `UNIQUE` — specifies that the column or a combination of columns should contain only unique values. Null values are not considered equal. `PRIMARY KEY` — specifies that the column or a combination of columns should contain unique (non-duplicate) non-null values. Only one primary key can be specified for a table (it can include several columns). `EXCLUDE` — ensures that if any two rows in a table are compared on a specified set of columns using specified operators, at least one of these operator comparisons will return `FALSE`. Syntax:

```
[ CONSTRAINT <constraint_name> ]
{ CHECK ( <expression> ) [ NO INHERIT ]
| UNIQUE ( <column_name> [, ... ] ) <index_parameters>
| PRIMARY KEY ( <column_name> [, ... ] ) <index_parameters>
| EXCLUDE [ USING <index_method> ] ( <exclude_element> WITH <operator> [, ...
] )
    <index_parameters> [ WHERE ( <predicate> ) ]
[ DEFERRABLE | NOT DEFERRABLE ] [ INITIALLY DEFERRED | INITIALLY IMMEDIATE ]
```

You can use this form of constraints to define primary keys based on several columns

<source_table>	An original table, which structure is used when a new table is being created via the `LIKE` statement
<like_option>	Defines whether to include or exclude the following parts of column definitions (when copying the original table via the `LIKE` statement): `AM` — the access method of the original table. `COMMENTS` — comments for the copied columns, constraints, and indexes. `CONSTRAINTS` — column constraints `CHECK`. `NOT NULL`-constraints are always copied to the new table. `DEFAULTS` — default expressions. By default, they are excluded, which results in the copied columns in the new table not having defaults. `ENCODING` — the character set encoding for columns. For append-optimized column-oriented tables only. `GENERATED` — the generated column definitions. `IDENTITY` — the identity column definitions. `INDEXES` — indexes, `PRIMARY KEY` and `UNIQUE` constraints. `STATISTICS` — extended statistics. `STORAGE` — column storage settings. By default, they are excluded, which results in the copied columns in the new table having type-specific default settings. `ALL` — all parts mentioned above. Syntax:

```
{INCLUDING|EXCLUDING} {DEFAULTS|CONSTRAINTS|INDEXES|STORAGE|COMMENTS|ALL}
```

<parent_table>	A parent table, which structure is used when a new table is being created via the `INHERITS` statement
<storage_table_directive>	Specifies the table storage settings: `appendoptimized` — whether or not the table should be created as append-optimized. If `FALSE` or not declared, the table will be created as a heap table. Starting with ADB 7, the `USING` clause can be used for this purpose. `blocksize` — block size (in bytes). Only for append-optimized tables. `orientation` — data orientation. If `ROW` or not declared, the table will be created as a row-oriented table, and as a column-oriented — otherwise. Only for append-optimized tables. `checksum` — if `TRUE` or not declared, CRC checksum validation is enabled. The checksum is calculated during block creation and is stored on disk. Checksum validation is performed during block reads. If the checksum calculated during the read does not match the stored checksum, the transaction is canceled. Only for append-optimized tables. `compresstype` — compression type. Only for append-optimized tables. `compresslevel` — compression level. Only for append-optimized tables. `fillfactor` — a percentage of filling of a standard fixed-size block (between `10` and `100`. `100` (complete packing) is the default value. When a smaller fillfactor is specified, `INSERT` operations pack table pages only to the indicated percentage; the remaining space on each page is reserved for updating rows on that page. For a table whose entries are never updated, complete packing is the best choice, but in heavily updated tables smaller fillfactors are appropriate. The `fillfactor` option is valid only for heap tables. `oids` — if `FALSE` or not declared, object identifiers (OID) are not assigned to the table rows. It is not recommended to use OIDs for user-created tables, as using OIDs for large tables can cause wrap-around of the 32-bit OID counter. Syntax:

```
appendoptimized={TRUE|FALSE}
blocksize={8192-2097152}
orientation={COLUMN|ROW}
checksum={TRUE|FALSE}
compresstype={ZLIB|ZSTD|RLE_TYPE|NONE}
compresslevel={0-9}
fillfactor={10-100}
[oids=FALSE]
```

For more details on compression options, see Data compression.

The default values for some storage options listed above can be changed via the server configuration parameter gp_default_storage_options

<tablespace_name>	A name of the tablespace in which the table should be stored
<opclass>	An operator class name. Can be specified optionally to use a non-default hash function for distribution
<partition_type>	A partitioning type. Possible values: `LIST` — list of values. `RANGE` — numeric or date range.
<partition_specification>	A partitioning specification including: partitions, subpartitions, default partition. For more details, see Partitioning

Recommendations on choosing column data types

There are some recommendations on how to choose data types for columns in the most efficient way:

- Choose data types according to the stored data. For example, use numeric data types for numbers, character data types for strings, date or timestamp data types for dates, and so on.
- The data types you select should use the least possible space but can still accommodate your data. For example, use `INT` instead of `BIGINT`, `TEXT` instead of `CHAR(<n>)`, etc.
- When using the smallest numeric data types, try to forecast their possible expansion in the future. Pay attention that changing from a smaller datatype to a larger one after loading large amounts of data can be costly. Therefore, if your current data values fit, for example, in the `SMALLINT` type properly but it is likely that the values will expand later, `INT` is the better long-term choice.
- If possible, use specialized data types: `INET`, `CIDR`, `JSON`, `JSONB`, `MACADDR`, etc.
- Use the same data types in the columns for which `JOIN` operations will be performed. When data types are different, the database should convert one of them in order to compare data values correctly, which adds unnecessary overhead.
- Though Greenplum has a rich set of native data types, you can create your own ones. Usage of `ENUM` can give a huge performance boost.

Examples

Create a heap table with REPLICATED distribution type and column constraints

```
CREATE TABLE book_type(id SERIAL PRIMARY KEY, name TEXT UNIQUE NOT NULL)
DISTRIBUTED REPLICATED;
```

Create a heap table with REPLICATED distribution type and table constraints

```
CREATE TABLE book_cover(id SERIAL, name TEXT NOT NULL, PRIMARY KEY(id), UNIQUE(name))
DISTRIBUTED REPLICATED;
```

Create an append-optimized column-oriented table (distributed by the column value)

```
CREATE TABLE author(id SERIAL, name TEXT NOT NULL)
USING ao_column
DISTRIBUTED BY(id);
```

Create an append-optimized row-oriented table with ZLIB compression (at the table level)

```
CREATE TABLE book(id SERIAL, title TEXT, author_id INT NOT NULL, public_year SMALLINT NULL, type_id INT NOT NULL, cover_id INT
NOT NULL)
USING ao_row
WITH (compresstype=ZLIB, compresslevel=5)
DISTRIBUTED BY(id);
```

Create an append-optimized column-oriented table with ZSTD compression (at the column level)

```
CREATE TABLE book_free_fragment(id SERIAL, book_id INT, shop_id INT, fragment TEXT ENCODING(compresstype=ZSTD,
compresslevel=5))
USING ao_column
DISTRIBUTED BY(id);
```

Create a table via the LIKE statement

1. Create the `book_type` table if it does not exist yet:

```
CREATE TABLE IF NOT EXISTS book_type(id SERIAL PRIMARY KEY, name TEXT UNIQUE NOT NULL)
DISTRIBUTED REPLICATED;
```

The possible result is:

```
NOTICE:  relation "book_type" already exists, skipping
CREATE TABLE
```

2. Check the structure of the created table via the `psql` command `\d+`:

```
\d+ book_type
```

Result:

```
                                   Table "public.book_type"
 Column |  Type   | Collation | Nullable |                Default                | Storage  | Stats target | Description
--------+---------+-----------+----------+---------------------------------------+----------+--------------+-------------
 id     | integer |           | not null | nextval('book_type_id_seq'::regclass) | plain    |              |
 name   | text    |           | not null |                                       | extended |              |
Indexes:
    "book_type_pkey" PRIMARY KEY, btree (id)
    "book_type_name_key" UNIQUE CONSTRAINT, btree (name)
Distributed Replicated
Access method: heap
```

3. Add another table `book_type_arch` with the same structure and constraints via the `LIKE` statement:

```
CREATE TABLE book_type_arch(LIKE book_type INCLUDING ALL)
DISTRIBUTED REPLICATED;
```

Result:

```
CREATE TABLE
```

4. Check the structure of the created table via the `psql` command `\d+`:

```
\d+ book_type_arch
```

You can see that it is similar to the `book_type` table:

```
                                 Table "public.book_type_arch"
 Column |  Type   | Collation | Nullable |                Default                | Storage  | Stats target | Description
--------+---------+-----------+----------+---------------------------------------+----------+--------------+-------------
 id     | integer |           | not null | nextval('book_type_id_seq'::regclass) | plain    |              |
 name   | text    |           | not null |                                       | extended |              |
Indexes:
    "book_type_arch_pkey" PRIMARY KEY, btree (id)
    "book_type_arch_name_key" UNIQUE CONSTRAINT, btree (name)
Distributed Replicated
Access method: heap
```

CAUTION

Pay attention that both tables now use the same sequence `book_type_id_seq` for the first column `id`. It can lead to unexpected consequences.

5. In order to check that two tables are not related anymore, alter the first table `book_type`:

```
ALTER TABLE book_type
DROP COLUMN name;
```

Result:

```
ALTER TABLE
```

6. Check the structure of the `book_type_arch` table again:

```
\d+ book_type_arch
```

The table has not changed:

```
                                 Table "public.book_type_arch"
 Column |  Type   | Collation | Nullable |                Default                | Storage  | Stats target | Description
--------+---------+-----------+----------+---------------------------------------+----------+--------------+-------------
 id     | integer |           | not null | nextval('book_type_id_seq'::regclass) | plain    |              |
 name   | text    |           | not null |                                       | extended |              |
Indexes:
    "book_type_arch_pkey" PRIMARY KEY, btree (id)
    "book_type_arch_name_key" UNIQUE CONSTRAINT, btree (name)
Distributed Replicated
Access method: heap
```

Create a table via the INHERITS statement

1. Create the `book` table if it does not exist yet:

```
CREATE TABLE IF NOT EXISTS book(id SERIAL, title TEXT, author_id INT NOT NULL, public_year SMALLINT NULL, type_id INT NOT NULL,
cover_id INT NOT NULL)
USING ao_row
WITH (compresstype=ZLIB, compresslevel=5)
DISTRIBUTED BY(id);
```

The possible result is:

```
NOTICE:  relation "book" already exists, skipping
CREATE TABLE
```

2. Check the structure of the created table via the `psql` command `\d+`:

```
\d+ book
```

Result:

```
                                       Table "public.book"
   Column    |   Type   | Collation | Nullable |              Default              | Storage  | Stats target | Description
-------------+----------+-----------+----------+-----------------------------------+----------+--------------+-------------
 id          | integer  |           | not null | nextval('book_id_seq'::regclass) | plain    |              |
 title       | text     |           |          |                                   | extended |              |
 author_id   | integer  |           | not null |                                   | plain    |              |
 public_year | smallint |           |          |                                   | plain    |              |
 type_id     | integer  |           | not null |                                   | plain    |              |
 cover_id    | integer  |           | not null |                                   | plain    |              |
Distributed by: (id)
Access method: ao_row
Options: compresstype=zlib, compresslevel=5, blocksize=32768, checksum=true
```

3. Add another table `book_arch` with the same structure and constraints via the `INHERITS` statement:

```
CREATE TABLE book_arch()
INHERITS(book)
USING ao_row
WITH (compresstype=ZLIB, compresslevel=5)
DISTRIBUTED BY(id);
```

Result:

```
CREATE TABLE
```

4. Check the structure of the created table via the `psql` command `\d+`:

```
\d+ book_arch
```

You can see that it is similar to the `book` table:

```
                                     Table "public.book_arch"
   Column    |   Type   | Collation | Nullable |              Default              | Storage  | Stats target | Descriptio
n
-------------+----------+-----------+----------+-----------------------------------+----------+--------------+-----------
--
 id          | integer  |           | not null | nextval('book_id_seq'::regclass) | plain    |              |
 title       | text     |           |          |                                   | extended |              |
 author_id   | integer  |           | not null |                                   | plain    |              |
 public_year | smallint |           |          |                                   | plain    |              |
 type_id     | integer  |           | not null |                                   | plain    |              |
 cover_id    | integer  |           | not null |                                   | plain    |              |
Inherits: book
Distributed by: (id)
Access method: ao_row
Options: compresstype=zlib, compresslevel=5, blocksize=32768, checksum=true
```

5. In order to check that two tables stay related after creation is complete, alter the first table `book`:

```
ALTER TABLE book
DROP COLUMN type_id;
```

Result:

```
ALTER TABLE
```

6. Check the structure of the `book_arch` table again:

```
\d+ book_arch
```

The `type_id` column does not exist anymore, just like in the parent `book` table:

```
                                     Table "public.book_arch"
   Column    |   Type   | Collation | Nullable |              Default              | Storage  | Stats target | Descriptio
n
-------------+----------+-----------+----------+-----------------------------------+----------+--------------+-----------
--
 id          | integer  |           | not null | nextval('book_id_seq'::regclass) | plain    |              |
 title       | text     |           |          |                                   | extended |              |
 author_id   | integer  |           | not null |                                   | plain    |              |
 public_year | smallint |           |          |                                   | plain    |              |
 cover_id    | integer  |           | not null |                                   | plain    |              |
Inherits: book
Distributed by: (id)
Access method: ao_row
Options: compresstype=zlib, compresslevel=5, blocksize=32768, checksum=true
```

Create a temporary table

1. Start a new transaction:

```
START TRANSACTION;
```

Result:

```
START TRANSACTION
```

2. Create a temporary table. In the `ON COMMIT` clause define that the table should be dropped after the current transaction is committed:

```
CREATE TEMP TABLE book_temp(id INT, name TEXT)
ON COMMIT DROP
DISTRIBUTED BY(id);
```

Result:

```
CREATE TABLE
```

3. Check the table existence via the `psql` command `\dt+`:

```
\dt+
```

Result:

```
                                  List of relations
    Schema    |        Name        | Type  |  Owner  |       Storage        |  Size   | Description
--------------+--------------------+-------+---------+----------------------+---------+-------------
 pg_temp_1229 | book_temp          | table | gpadmin | heap                 | 544 kB  |
 public       | author             | table | gpadmin | append only columnar | 1088 kB |
 public       | book               | table | gpadmin | append only          | 1088 kB |
 public       | book_arch          | table | gpadmin | append only          | 1088 kB |
 public       | book_cover         | table | gpadmin | heap                 | 544 kB  |
 public       | book_free_fragment | table | gpadmin | append only columnar | 1088 kB |
 public       | book_type          | table | gpadmin | heap                 | 544 kB  |
 public       | book_type_arch     | table | gpadmin | heap                 | 544 kB  |
 public       | spatial_ref_sys    | table | gpadmin | heap                 | 7424 kB |
(9 rows)
```

4. Commit the current transaction:

```
COMMIT;
```

Result:

```
COMMIT
```

5. Check the temporary table existence again:

```
\dt+
```

You can see that the `book_temp` table is not available anymore:

```
                              List of relations
 Schema |        Name        | Type  |  Owner  |       Storage        |  Size   | Description
--------+--------------------+-------+---------+----------------------+---------+-------------
 public | author             | table | gpadmin | append only columnar | 1088 kB |
 public | book               | table | gpadmin | append only          | 1088 kB |
 public | book_arch          | table | gpadmin | append only          | 1088 kB |
 public | book_cover         | table | gpadmin | heap                 | 544 kB  |
 public | book_free_fragment | table | gpadmin | append only columnar | 1088 kB |
 public | book_type          | table | gpadmin | heap                 | 544 kB  |
 public | book_type_arch     | table | gpadmin | heap                 | 544 kB  |
 public | spatial_ref_sys    | table | gpadmin | heap                 | 7424 kB |
(8 rows)
```

Create a table with the CHECK constraint

1. Create a table with the following restriction: there should be no rows with the value less than `1` in the `book_count` column:

```
CREATE TABLE book_order(id INT, book_id INT, client_id INT, book_count SMALLINT, order_date TIMESTAMP, CHECK(book_count >= 1))
USING ao_row
WITH (compresstype=ZLIB, compresslevel=5)
DISTRIBUTED BY(id);
```

The following query uses the same `CHECK` constraint, but at the column level:

```
CREATE TABLE book_order(id INT, book_id INT, client_id INT, book_count SMALLINT CHECK(book_count >= 1), order_date TIMESTAMP)
USING ao_row
WITH (compresstype=ZLIB, compresslevel=5)
DISTRIBUTED BY(id);
```

2. Regardless of the selected syntax form, you can check the created constraint. Try to insert rows into the `book_order` table with `0` in the `book_count` column:

```
INSERT INTO book_order VALUES(1, 1, 1, 0, now());
```

The query returns the error:

```
ERROR:  new row for relation "book_order" violates check constraint "book_order_book_count_check"  (seg4 10.92.16.97:10000
pid=45029)
DETAIL:  Failing row contains (1, 1, 1, 0, 2025-10-01 13:07:01.633072).
```

3. Change the `INSERT` query as described below:

```
INSERT INTO book_order VALUES(1, 1, 1, 1, now());
```

Now the result is successful:

```
INSERT 0 1
```

Create a partitioned table

```
CREATE TABLE client (id INT, name TEXT, gender CHAR(1))
DISTRIBUTED BY (id)
PARTITION BY LIST (gender)
( PARTITION girls VALUES ('F'),
  PARTITION boys VALUES ('M'));
```

The result is:

```
NOTICE:  CREATE TABLE will create partition "client_1_prt_girls" for table "client"
NOTICE:  CREATE TABLE will create partition "client_1_prt_boys" for table "client"
CREATE TABLE
```

NOTE

For more examples, see Partitioning.

View the list of tables

To view the list of available tables in the current database, use one of the following ways:

- Run the SQL query against the `information_schema.tables` catalog table. Use `table_type = 'BASE TABLE'` in the `WHERE` predicate if you want to list only tables. Otherwise, the result will also contain views (having `table_type = 'VIEW'`).

```
SELECT table_catalog, table_schema, table_name, table_type
FROM information_schema.tables
WHERE table_type = 'BASE TABLE'
ORDER BY table_schema, table_name;
```

This command returns the following columns:

- `table_catalog` — database name;
- `table_schema` — schema name;
- `table_name` — table name;
- `table_type` — table type (`BASE TABLE` for tables, `VIEW` for views).

Query result

```
 table_catalog |    table_schema    |        table_name        | table_type
---------------+--------------------+--------------------------+------------
 adb           | arenadata_toolkit  | daily_operation          | BASE TABLE
 adb           | arenadata_toolkit  | db_files_current         | BASE TABLE
 adb           | arenadata_toolkit  | db_files_history         | BASE TABLE
 adb           | arenadata_toolkit  | operation_exclude        | BASE TABLE
 adb           | diskquota          | quota_config             | BASE TABLE
 adb           | diskquota          | state                    | BASE TABLE
 adb           | diskquota          | table_size               | BASE TABLE
 adb           | gp_toolkit         | gp_disk_free             | BASE TABLE
 adb           | gp_toolkit         | __gp_log_master_ext      | BASE TABLE
 adb           | gp_toolkit         | __gp_log_segment_ext     | BASE TABLE
 adb           | information_schema | sql_features             | BASE TABLE
 adb           | information_schema | sql_implementation_info  | BASE TABLE
 adb           | information_schema | sql_languages            | BASE TABLE
 adb           | information_schema | sql_packages             | BASE TABLE
 adb           | information_schema | sql_parts                | BASE TABLE
 adb           | information_schema | sql_sizing               | BASE TABLE
 adb           | information_schema | sql_sizing_profiles      | BASE TABLE
 adb           | kadb               | offsets                  | BASE TABLE
 adb           | madlib             | migrationhistory         | BASE TABLE
 adb           | pg_catalog         | gp_configuration_history | BASE TABLE
 adb           | pg_catalog         | gp_distribution_policy   | BASE TABLE
 adb           | pg_catalog         | gp_fastsequence          | BASE TABLE
 adb           | pg_catalog         | gp_id                    | BASE TABLE
 adb           | pg_catalog         | gp_segment_configuration | BASE TABLE
 adb           | pg_catalog         | gp_version_at_initdb     | BASE TABLE
 adb           | pg_catalog         | pg_aggregate             | BASE TABLE
 adb           | pg_catalog         | pg_am                    | BASE TABLE
 adb           | pg_catalog         | pg_amop                  | BASE TABLE
 adb           | pg_catalog         | pg_amproc                | BASE TABLE
 adb           | pg_catalog         | pg_appendonly            | BASE TABLE
 adb           | pg_catalog         | pg_attrdef               | BASE TABLE
 adb           | pg_catalog         | pg_attribute             | BASE TABLE
 adb           | pg_catalog         | pg_attribute_encoding    | BASE TABLE
 adb           | pg_catalog         | pg_authid                | BASE TABLE
 adb           | pg_catalog         | pg_auth_members          | BASE TABLE
 adb           | pg_catalog         | pg_auth_time_constraint  | BASE TABLE
 adb           | pg_catalog         | pg_cast                  | BASE TABLE
 adb           | pg_catalog         | pg_class                 | BASE TABLE
 adb           | pg_catalog         | pg_collation             | BASE TABLE
 adb           | pg_catalog         | pg_compression           | BASE TABLE
 adb           | pg_catalog         | pg_constraint            | BASE TABLE
 adb           | pg_catalog         | pg_conversion            | BASE TABLE
 adb           | pg_catalog         | pg_database              | BASE TABLE
 adb           | pg_catalog         | pg_db_role_setting       | BASE TABLE
 adb           | pg_catalog         | pg_default_acl           | BASE TABLE
 adb           | pg_catalog         | pg_depend                | BASE TABLE
 adb           | pg_catalog         | pg_description           | BASE TABLE
 adb           | pg_catalog         | pg_enum                  | BASE TABLE
 adb           | pg_catalog         | pg_event_trigger         | BASE TABLE
 adb           | pg_catalog         | pg_extension             | BASE TABLE
 adb           | pg_catalog         | pg_extprotocol           | BASE TABLE
 adb           | pg_catalog         | pg_exttable              | BASE TABLE
 adb           | pg_catalog         | pg_foreign_data_wrapper  | BASE TABLE
 adb           | pg_catalog         | pg_foreign_server        | BASE TABLE
 adb           | pg_catalog         | pg_foreign_table         | BASE TABLE
 adb           | pg_catalog         | pg_index                 | BASE TABLE
 adb           | pg_catalog         | pg_inherits              | BASE TABLE
 adb           | pg_catalog         | pg_language              | BASE TABLE
 adb           | pg_catalog         | pg_largeobject           | BASE TABLE
 adb           | pg_catalog         | pg_largeobject_metadata  | BASE TABLE
 adb           | pg_catalog         | pg_namespace             | BASE TABLE
 adb           | pg_catalog         | pg_opclass               | BASE TABLE
 adb           | pg_catalog         | pg_operator              | BASE TABLE
 adb           | pg_catalog         | pg_opfamily              | BASE TABLE
 adb           | pg_catalog         | pg_partition             | BASE TABLE
 adb           | pg_catalog         | pg_partition_encoding    | BASE TABLE
 adb           | pg_catalog         | pg_partition_rule        | BASE TABLE
 adb           | pg_catalog         | pg_pltemplate            | BASE TABLE
 adb           | pg_catalog         | pg_proc                  | BASE TABLE
 adb           | pg_catalog         | pg_proc_callback         | BASE TABLE
 adb           | pg_catalog         | pg_range                 | BASE TABLE
 adb           | pg_catalog         | pg_resgroup              | BASE TABLE
 adb           | pg_catalog         | pg_resgroupcapability    | BASE TABLE
 adb           | pg_catalog         | pg_resourcetype          | BASE TABLE
 adb           | pg_catalog         | pg_resqueue              | BASE TABLE
 adb           | pg_catalog         | pg_resqueuecapability    | BASE TABLE
 adb           | pg_catalog         | pg_rewrite               | BASE TABLE
 adb           | pg_catalog         | pg_seclabel              | BASE TABLE
 adb           | pg_catalog         | pg_shdepend              | BASE TABLE
 adb           | pg_catalog         | pg_shdescription         | BASE TABLE
 adb           | pg_catalog         | pg_shseclabel            | BASE TABLE
 adb           | pg_catalog         | pg_statistic             | BASE TABLE
 adb           | pg_catalog         | pg_stat_last_operation   | BASE TABLE
 adb           | pg_catalog         | pg_stat_last_shoperation | BASE TABLE
 adb           | pg_catalog         | pg_tablespace            | BASE TABLE
 adb           | pg_catalog         | pg_trigger               | BASE TABLE
 adb           | pg_catalog         | pg_ts_config             | BASE TABLE
 adb           | pg_catalog         | pg_ts_config_map         | BASE TABLE
 adb           | pg_catalog         | pg_ts_dict               | BASE TABLE
 adb           | pg_catalog         | pg_ts_parser             | BASE TABLE
 adb           | pg_catalog         | pg_ts_template           | BASE TABLE
 adb           | pg_catalog         | pg_type                  | BASE TABLE
 adb           | pg_catalog         | pg_type_encoding         | BASE TABLE
 adb           | pg_catalog         | pg_user_mapping          | BASE TABLE
 adb           | public             | author                   | BASE TABLE
 adb           | public             | book                     | BASE TABLE
 adb           | public             | book_arch                | BASE TABLE
 adb           | public             | book_cover               | BASE TABLE
 adb           | public             | book_free_fragment       | BASE TABLE
 adb           | public             | book_order               | BASE TABLE
 adb           | public             | book_type                | BASE TABLE
 adb           | public             | book_type_arch           | BASE TABLE
 adb           | public             | client                   | BASE TABLE
 adb           | public             | client_1_prt_boys        | BASE TABLE
 adb           | public             | client_1_prt_girls       | BASE TABLE
 adb           | public             | spatial_ref_sys          | BASE TABLE
(106 rows)
```

- Run the SQL query against the `pg_catalog.pg_tables` catalog view.

```
SELECT * FROM pg_catalog.pg_tables
ORDER BY schemaname, tablename;
```

This command returns the following columns:

- `schemaname` — schema name;
- `tablename` — table name;
- `tableowner` — table owner;
- `tablespace` — tablespace name;
- `hasindexes` — boolean value that defines whether the table has indexes;
- `hasrules` — boolean value that defines whether the table has rules;
- `hastriggers` — boolean value that defines whether the table has triggers.

Query result

```
     schemaname     |        tablename         | tableowner | tablespace | hasindexes | hasrules | hastriggers
--------------------+--------------------------+------------+------------+------------+----------+-------------
 arenadata_toolkit  | daily_operation          | gpadmin    |            | f          | f        | f
 arenadata_toolkit  | db_files_current         | gpadmin    |            | f          | f        | f
 arenadata_toolkit  | db_files_history         | gpadmin    |            | f          | f        | f
 arenadata_toolkit  | operation_exclude        | gpadmin    |            | f          | f        | f
 diskquota          | quota_config             | gpadmin    |            | t          | f        | f
 diskquota          | state                    | gpadmin    |            | t          | f        | f
 diskquota          | table_size               | gpadmin    |            | t          | f        | f
 gp_toolkit         | __gp_log_master_ext      | gpadmin    |            | f          | f        | f
 gp_toolkit         | __gp_log_segment_ext     | gpadmin    |            | f          | f        | f
 gp_toolkit         | gp_disk_free             | gpadmin    |            | f          | f        | f
 information_schema | sql_features             | gpadmin    |            | f          | f        | f
 information_schema | sql_implementation_info  | gpadmin    |            | f          | f        | f
 information_schema | sql_languages            | gpadmin    |            | f          | f        | f
 information_schema | sql_packages             | gpadmin    |            | f          | f        | f
 information_schema | sql_parts                | gpadmin    |            | f          | f        | f
 information_schema | sql_sizing               | gpadmin    |            | f          | f        | f
 information_schema | sql_sizing_profiles      | gpadmin    |            | f          | f        | f
 kadb               | offsets                  | gpadmin    |            | t          | f        | f
 madlib             | migrationhistory         | gpadmin    |            | t          | f        | f
 pg_catalog         | gp_configuration_history | gpadmin    | pg_global  | f          | f        | f
 pg_catalog         | gp_distribution_policy   | gpadmin    |            | t          | f        | f
 pg_catalog         | gp_fastsequence          | gpadmin    |            | t          | f        | f
 pg_catalog         | gp_id                    | gpadmin    | pg_global  | f          | f        | f
 pg_catalog         | gp_segment_configuration | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | gp_version_at_initdb     | gpadmin    | pg_global  | f          | f        | f
 pg_catalog         | pg_aggregate             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_am                    | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_amop                  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_amproc                | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_appendonly            | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_attrdef               | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_attribute             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_attribute_encoding    | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_auth_members          | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_auth_time_constraint  | gpadmin    | pg_global  | f          | f        | f
 pg_catalog         | pg_authid                | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_cast                  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_class                 | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_collation             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_compression           | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_constraint            | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_conversion            | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_database              | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_db_role_setting       | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_default_acl           | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_depend                | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_description           | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_enum                  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_event_trigger         | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_extension             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_extprotocol           | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_exttable              | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_foreign_data_wrapper  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_foreign_server        | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_foreign_table         | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_index                 | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_inherits              | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_language              | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_largeobject           | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_largeobject_metadata  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_namespace             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_opclass               | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_operator              | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_opfamily              | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_partition             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_partition_encoding    | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_partition_rule        | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_pltemplate            | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_proc                  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_proc_callback         | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_range                 | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_resgroup              | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_resgroupcapability    | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_resourcetype          | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_resqueue              | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_resqueuecapability    | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_rewrite               | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_seclabel              | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_shdepend              | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_shdescription         | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_shseclabel            | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_stat_last_operation   | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_stat_last_shoperation | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_statistic             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_tablespace            | gpadmin    | pg_global  | t          | f        | f
 pg_catalog         | pg_trigger               | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_ts_config             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_ts_config_map         | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_ts_dict               | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_ts_parser             | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_ts_template           | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_type                  | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_type_encoding         | gpadmin    |            | t          | f        | f
 pg_catalog         | pg_user_mapping          | gpadmin    |            | t          | f        | f
 public             | author                   | gpadmin    |            | f          | f        | f
 public             | book                     | gpadmin    |            | f          | f        | f
 public             | book_arch                | gpadmin    |            | f          | f        | f
 public             | book_cover               | gpadmin    |            | f          | f        | f
 public             | book_free_fragment       | gpadmin    |            | f          | f        | f
 public             | book_order               | gpadmin    |            | f          | f        | f
 public             | book_type                | gpadmin    |            | t          | f        | f
 public             | book_type_arch           | gpadmin    |            | t          | f        | f
 public             | client                   | gpadmin    |            | f          | f        | f
 public             | client_1_prt_boys        | gpadmin    |            | f          | f        | f
 public             | client_1_prt_girls       | gpadmin    |            | f          | f        | f
 public             | spatial_ref_sys          | gpadmin    |            | t          | f        | f
(106 rows)
```

- If you work in the `psql` client program, you can use meta-commands to show the list of tables:
  - `\dt` — this command returns the following information:
    - `Schema` — schema name;
    - `Name` — table name;
    - `Type` — relation type (`table` for tables, `view` for views);
    - `Owner` — table owner;
    - `Storage` — table type.

```
                          List of relations
 Schema |        Name        | Type  |  Owner  |       Storage
--------+--------------------+-------+---------+----------------------
 public | author             | table | gpadmin | append only columnar
 public | book               | table | gpadmin | append only
 public | book_arch          | table | gpadmin | append only
 public | book_cover         | table | gpadmin | heap
 public | book_free_fragment | table | gpadmin | append only columnar
 public | book_order         | table | gpadmin | append only
 public | book_type          | table | gpadmin | heap
 public | book_type_arch     | table | gpadmin | heap
 public | client             | table | gpadmin | heap
 public | client_1_prt_boys  | table | gpadmin | heap
 public | client_1_prt_girls | table | gpadmin | heap
 public | spatial_ref_sys    | table | gpadmin | heap
(12 rows)
```

  - `\dt+` — in comparison with `\dt`, this command returns additional fields:
    - `Size` — table size;
    - `Description` — table description.

```
                                List of relations
 Schema |        Name        | Type  |  Owner  |       Storage        |  Size   | Description
--------+--------------------+-------+---------+----------------------+---------+-------------
 public | author             | table | gpadmin | append only columnar | 1088 kB |
 public | book               | table | gpadmin | append only          | 1088 kB |
 public | book_arch          | table | gpadmin | append only          | 1088 kB |
 public | book_cover         | table | gpadmin | heap                 | 544 kB  |
 public | book_free_fragment | table | gpadmin | append only columnar | 1088 kB |
 public | book_order         | table | gpadmin | append only          | 608 kB  |
 public | book_type          | table | gpadmin | heap                 | 544 kB  |
 public | book_type_arch     | table | gpadmin | heap                 | 544 kB  |
 public | client             | table | gpadmin | heap                 | 544 kB  |
 public | client_1_prt_boys  | table | gpadmin | heap                 | 544 kB  |
 public | client_1_prt_girls | table | gpadmin | heap                 | 544 kB  |
 public | spatial_ref_sys    | table | gpadmin | heap                 | 7424 kB |
(12 rows)
```

TIP

- In order to add system tables to the output of `psql` commands listed above, use the `S` postfix after their names: `\dtS`, `\dtS+`.
- To list tables with the specified schema name, you can use templates with regular expressions. For example, the `\dt kadb.*` command returns tables with the schema name `kadb`. By default, `\dt` and `\dt+` commands show tables for the current schema. For more information, see Schema search path.

View the specified table

To view the structure of the specified table, you can use one of the following `psql` commands:

- `\d` — this command returns the following information:
  - A list of table columns with their names, data types, and modifiers (i.e. constraints, sequences, etc.).
  - Table storage options including: compression type, compression level, block size. Specified only for append-optimized tables.
  - Whether or not the checksum is used.
  - A number of child tables (created via the `INHERITS` clause).
  - Distribution policy.

```
\d author
```

```
                          Table "public.author"
 Column |  Type   | Collation | Nullable |               Default
--------+---------+-----------+----------+------------------------------------
 id     | integer |           | not null | nextval('author_id_seq'::regclass)
 name   | text    |           | not null |
Distributed by: (id)
```

```
\d book
```

```
                             Table "public.book"
   Column    |   Type   | Collation | Nullable |              Default
-------------+----------+-----------+----------+----------------------------------
 id          | integer  |           | not null | nextval('book_id_seq'::regclass)
 title       | text     |           |          |
 author_id   | integer  |           | not null |
 public_year | smallint |           |          |
 cover_id    | integer  |           | not null |
Number of child tables: 1 (Use \d+ to list them.)
Distributed by: (id)
```

- `\d+` — in comparison with `\d`, this command returns some additional information:
  - Access method: heap, `ao_row`, or `ao_column`.
  - Column storage options: compression type, compression level, and block size.
  - Some additional column data including: statistic target, description.
  - A list of child tables (created via the `INHERITS` clause).
  - Table storage options (defined via the `WITH` clause).

```
\d+ author
```

```
                                            Table "public.author"
 Column |  Type   | Collation | Nullable |               Default               | Storage  | Stats target | Compress
ion Type | Compression Level | Block Size | Description
--------+---------+-----------+----------+------------------------------------+----------+--------------+---------
---------+-------------------+------------+-------------
 id     | integer |           | not null | nextval('author_id_seq'::regclass) | plain    |              | none
         | 0                 | 32768      |
 name   | text    |           | not null |                                    | extended |              | none
         | 0                 | 32768      |
Distributed by: (id)
Access method: ao_column
Options: blocksize=32768, compresslevel=0, compresstype=none, checksum=true
```

```
\d+ book
```

```
                                          Table "public.book"
   Column    |   Type   | Collation | Nullable |              Default              | Storage  | Stats target | Desc
ription
-------------+----------+-----------+----------+-----------------------------------+----------+--------------+-----
--------
 id          | integer  |           | not null | nextval('book_id_seq'::regclass) | plain    |              |
 title       | text     |           |          |                                   | extended |              |
 author_id   | integer  |           | not null |                                   | plain    |              |
 public_year | smallint |           |          |                                   | plain    |              |
 cover_id    | integer  |           | not null |                                   | plain    |              |
Child tables: book_arch
Distributed by: (id)
Access method: ao_row
Options: compresstype=zlib, compresslevel=5, blocksize=32768, checksum=true
```

System catalog tables

ADB (as well as Greenplum) supports several system catalog tables within the `pg_catalog` schema. Some of them can be useful for monitoring of your cluster and other DBA activity. Examples of such tables are listed below.

NOTE

For more information on all system catalog tables, see System Tables in Greenplum documentation.

gp_configuration_history

Contains information about system changes related to fault detection and recovery operations.

Basic columns

Column name	Purpose
time	A timestamp for the event occurred
dbid	A unique identifier of a segment (or coordinator) instance. Refers to `gp_segment_configuration.dbid`
desc	The event description

gp_segment_configuration

Contains information about mirroring and segment instance configuration.

Basic columns

Column name	Purpose
dbid	A unique identifier of a segment instance
content	A content identifier for a segment instance. A primary segment instance and its corresponding mirror always have the same `content` value. Possible values: For a **segment** the value is from `0` to `N-1`, where `N` is the number of primary segments in the system. For a **coordinator** the value is always `-1`.
role	The role that a segment is currently running as. Possible values are `p` (primary) or `m` (mirror)
preferred_role	The role that a segment was originally assigned at initialization time. Possible values are `p` (primary) or `m` (mirror)
mode	The synchronization status of a segment instance with its mirror copy. Possible values are `s` (Synchronized) or `n` (Not In Sync). This column always shows `n` for coordinator and `s` for standby coordinator. Use the `gp_stat_replication` table to determine the synchronization state between coordinator and standby coordinator
status	The fault status of a segment instance. Possible values are `u` (up) or `d` (down)
port	The TCP port the database server listener process is using
hostname	A hostname of a segment host
address	The hostname used to access a particular segment instance on a segment host. This value may be the same as `hostname` on systems that do not have per-interface hostnames configured
datadir	A segment instance data directory

pg_database

Stores information about databases.

Basic columns

Column name	Purpose
datname	A database name
datdba	A database owner, usually the user who created it. Refers to pg_authid.oid
dattablespace	A database tablespace. This tablespace is the default one for all system catalogs of that database and for all tables and indexes created within the database (unless the `TABLESPACE` clause is given for them when creating)

pg_class

Stores information about all database **relations**. In addition to tables, it includes indexes, sequences, views, composite types, and TOAST tables (which store large "out of line" attributes of the source table in the secondary table). Not all columns are meaningful for all relation types.

Basic columns

Column name	Purpose
relname	A relation name
relnamespace	OID of the schema that contains this relation. Refers to `pg_namespace.oid`
relowner	A relation owner, usually the user who created it. Refers to pg_authid.oid
relam	An access method: `btree`, `bitmap`, `brin`, `hash`, `gist`, `gin`, and `spgist` for indexes and `heap`, `ao_row`, and `ao_column` for tables. Refers to pg_am.oid
reltablespace	The tablespace in which the relation is stored. If zero, the database's default tablespace is used. Refers to `pg_tablespace.oid`
relpages	Size of the on-disk relation representation in pages (of 32 KB each). This is only an estimate used by the planner. It is updated by `VACUUM`, `ANALYZE`, and a few DDL commands
reltuples	A number of rows in the relation. This is only an estimate used by the planner. It is updated by `VACUUM`, `ANALYZE`, and a few DDL commands
reltoastrelid	OID of the TOAST table associated with the current relation, `0` if none. Refers to `pg_class.oid`
relpersistence	A relation persistence type: `p` — heap or append-optimized table; `u` — unlogged temporary table; `t` — temporary table.
relkind	A relation type: `r` — heap or append-optimized table; `i` — index; `S` — sequence; `t` — TOAST table; `v` — view; `m` — materialized view; `c` — composite type; `f` — foreign table; `u` — uncatalogued temporary heap table; `o` — internal append-optimized segment files and EOFs; `p` — partitioned table; `b` — append-only block directory; `M` — append-only visibility map.

pg_namespace

Stores information about schemas (namespaces).

Basic columns

Column name	Purpose
nspname	A schema name
nspowner	A schema owner, usually the user who created it. Refers to pg_authid.oid

pg_tablespace

Stores information about tablespaces.

Basic columns

Column name	Purpose
spcname	A tablespace name
spcowner	A tablespace owner, usually the user who created it. Refers to pg_authid.oid

Useful queries

Examples of queries to the system catalog tables are listed below.

List all databases sorted by their size

```
SELECT oid, datname, pg_size_pretty(pg_database_size(datname))
FROM pg_database
ORDER BY pg_database_size(datname) DESC;
```

The result is:

```
  oid  |  datname  | pg_size_pretty
-------+-----------+----------------
 16384 | adb       | 438 MB
 19895 | diskquota | 283 MB
 12812 | postgres  | 280 MB
     1 | template1 | 280 MB
 12809 | template0 | 276 MB
(5 rows)
```

List all schemas

```
SELECT oid, nspname FROM pg_catalog.pg_namespace ORDER BY nspname;
```

The result is:

```
  oid  |      nspname
-------+--------------------
 16385 | arenadata_toolkit
 19895 | diskquota
 12350 | gp_toolkit
 12890 | information_schema
 19968 | kadb
 16472 | madlib
  6104 | pg_aoseg
  7012 | pg_bitmapindex
    11 | pg_catalog
    99 | pg_toast
  2200 | public
(11 rows)
```

List all tablespaces

```
SELECT oid, spcname FROM pg_catalog.pg_tablespace ORDER BY spcname;
```

The result is:

```
 oid  |  spcname
------+------------
 1663 | pg_default
 1664 | pg_global
(2 rows)
```

List external/foreign tables

```
SELECT oid, relname FROM pg_catalog.pg_class WHERE relkind = 'f';
```

The result is:

```
  oid  |          relname
-------+----------------------------
 16410 | __gp_log_segment_ext
 16413 | __gp_log_coordinator_ext
 16479 | gp_disk_free
(3 rows)
```

Find a number of database relations grouped by their type

```
SELECT relkind, COUNT(*) FROM pg_class GROUP BY(relkind) ORDER BY COUNT(*);
```

The result is:

```
 relkind | count
---------+-------
 m       |     1
 p       |     2
 f       |     3
 M       |     8
 o       |     8
 S       |    11
 c       |    11
 t       |    59
 r       |   126
 i       |   227
 v       |   293
(8 rows)
```

List all TOAST tables with the main tables they are associated with

```
SELECT t1.oid AS "TOAST OID", t1.relname AS "TOAST name", t2.oid AS "Main OID", t2.relname AS "Main name"
FROM pg_catalog.pg_class t1
LEFT JOIN pg_catalog.pg_class t2 ON t1.oid = t2.reltoastrelid
WHERE t1.relkind = 't'
ORDER BY t2.relname;
```

The result is:

```
 TOAST OID |   TOAST name    | Main OID |        Main name
-----------+-----------------+----------+--------------------------
     53318 | pg_toast_53315  |    53315 | author
     53331 | pg_toast_53328  |    53328 | book
     53366 | pg_toast_53363  |    53363 | book_arch
     53306 | pg_toast_53302  |    53302 | book_cover
     53344 | pg_toast_53341  |    53341 | book_free_fragment
     53293 | pg_toast_53289  |    53289 | book_type
     53356 | pg_toast_53352  |    53352 | book_type_arch
     53409 | pg_toast_53406  |    53406 | client
     53423 | pg_toast_53420  |    53420 | client_1_prt_boys
     53415 | pg_toast_53412  |    53412 | client_1_prt_girls
     16398 | pg_toast_16396  |    16396 | daily_operation
     45090 | pg_toast_45087  |    45087 | db_files_current
     16388 | pg_toast_16386  |    16386 | db_files_history
      6092 | pg_toast_5036   |     5036 | gp_segment_configuration
     16408 | pg_toast_16406  |    16406 | operation_exclude
      2830 | pg_toast_2604   |     2604 | pg_attrdef
      6233 | pg_toast_6231   |     6231 | pg_attribute_encoding
      2832 | pg_toast_2606   |     2606 | pg_constraint
      2966 | pg_toast_2964   |     2964 | pg_db_role_setting
      2834 | pg_toast_2609   |     2609 | pg_description
      5510 | pg_toast_3079   |     3079 | pg_extension
      9905 | pg_toast_9903   |     9903 | pg_partition_encoding
      2836 | pg_toast_1255   |     1255 | pg_proc
      2838 | pg_toast_2618   |     2618 | pg_rewrite
      3598 | pg_toast_3596   |     3596 | pg_seclabel
      2846 | pg_toast_2396   |     2396 | pg_shdescription
      2840 | pg_toast_2619   |     2619 | pg_statistic
      2336 | pg_toast_2620   |     2620 | pg_trigger
      6222 | pg_toast_6220   |     6220 | pg_type_encoding
     18759 | pg_toast_18755  |    18755 | spatial_ref_sys
     12202 | pg_toast_12200  |    12200 | sql_features
     12207 | pg_toast_12205  |    12205 | sql_implementation_info
     12212 | pg_toast_12210  |    12210 | sql_languages
     12217 | pg_toast_12215  |    12215 | sql_packages
     12222 | pg_toast_12220  |    12220 | sql_parts
     12227 | pg_toast_12225  |    12225 | sql_sizing
     12232 | pg_toast_12230  |    12230 | sql_sizing_profiles
(37 rows)
```

Get information about a relation by its name

```
SELECT rel.oid,
       rel.relname,
       schema.nspname,
       ts.spcname,
       rel.relpersistence,
       rel.relkind,
       rel.relpages,
       rel.reltuples
FROM pg_catalog.pg_class rel
LEFT JOIN pg_catalog.pg_namespace schema ON schema.oid = rel.relnamespace
LEFT JOIN pg_catalog.pg_tablespace ts ON ts.oid = rel.reltablespace
WHERE rel.relname = 'pg_class';
```

Where `'pg_class'` can be replaced with any other relation name.

The result is:

```
 oid  | relname  |  nspname   | spcname | relpersistence | relkind | relpages | reltuples
------+----------+------------+---------+----------------+---------+----------+-----------
 1259 | pg_class | pg_catalog |         | p              | r       |        7 |       720
(1 row)
```

Get information on all segments in the current ADB cluster

```
SELECT * FROM pg_segment_configuration ORDER BY content, role;
```

The result is:

```
 dbid | content | role | preferred_role | mode | status | port  | hostname | address  |        datadir
------+---------+------+----------------+------+--------+-------+----------+----------+------------------------
   34 |      -1 | m    | m              | s    | u      |  5432 | bds-smdw | bds-smdw | /data1/master/gpseg-1
    1 |      -1 | p    | p              | n    | u      |  5432 | bds-mdw  | bds-mdw  | /data1/master/gpseg-1
   18 |       0 | m    | m              | s    | u      | 10500 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg0
    2 |       0 | p    | p              | s    | u      | 10000 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg0
   19 |       1 | m    | m              | s    | u      | 10501 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg1
    3 |       1 | p    | p              | s    | u      | 10001 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg1
   20 |       2 | m    | m              | s    | u      | 10502 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg2
    4 |       2 | p    | p              | s    | u      | 10002 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg2
   21 |       3 | m    | m              | s    | u      | 10503 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg3
    5 |       3 | p    | p              | s    | u      | 10003 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg3
   22 |       4 | m    | m              | s    | u      | 10504 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg4
    6 |       4 | p    | p              | s    | u      | 10004 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg4
   23 |       5 | m    | m              | s    | u      | 10505 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg5
    7 |       5 | p    | p              | s    | u      | 10005 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg5
   24 |       6 | m    | m              | s    | u      | 10506 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg6
    8 |       6 | p    | p              | s    | u      | 10006 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg6
   25 |       7 | m    | m              | s    | u      | 10507 | bds-sdw2 | bds-sdw2 | /data1/mirror/gpseg7
    9 |       7 | p    | p              | s    | u      | 10007 | bds-sdw1 | bds-sdw1 | /data1/primary/gpseg7
   26 |       8 | m    | m              | s    | u      | 10500 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg8
   10 |       8 | p    | p              | s    | u      | 10000 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg8
   27 |       9 | m    | m              | s    | u      | 10501 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg9
   11 |       9 | p    | p              | s    | u      | 10001 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg9
   28 |      10 | m    | m              | s    | u      | 10502 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg10
   12 |      10 | p    | p              | s    | u      | 10002 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg10
   29 |      11 | m    | m              | s    | u      | 10503 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg11
   13 |      11 | p    | p              | s    | u      | 10003 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg11
   30 |      12 | m    | m              | s    | u      | 10504 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg12
   14 |      12 | p    | p              | s    | u      | 10004 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg12
   31 |      13 | m    | m              | s    | u      | 10505 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg13
   15 |      13 | p    | p              | s    | u      | 10005 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg13
   32 |      14 | m    | m              | s    | u      | 10506 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg14
   16 |      14 | p    | p              | s    | u      | 10006 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg14
   33 |      15 | m    | m              | s    | u      | 10507 | bds-sdw1 | bds-sdw1 | /data1/mirror/gpseg15
   17 |      15 | p    | p              | s    | u      | 10007 | bds-sdw2 | bds-sdw2 | /data1/primary/gpseg15
(34 rows)
```

Find which host a mirror of the specified segment is located on

```
SELECT t2.hostname
FROM gp_segment_configuration t1
LEFT JOIN gp_segment_configuration t2 ON t1.content = t2.content
WHERE t1.dbid = 2
AND t2.preferred_role = 'm';
```

Where `2` can be replaced with the specified segment `dbid`.

The result is:

```
 hostname
----------
 bds-sdw2
(1 row)
```

List last 10 events related to segment configuration changes

```
SELECT * FROM gp_configuration_history ORDER BY time desc LIMIT 10;
```

The result is:

```
             time              | dbid |                                     description
-------------------------------+------+--------------------------------------------------------------------------------------
 2025-09-22 10:11:30.121835+00 |   16 | FTS: update role, status, and mode for dbid 16 with contentid 6 to m, u, and s
 2025-09-22 10:11:30.121817+00 |    8 | FTS: update role, status, and mode for dbid 8 with contentid 6 to p, u, and s
 2025-09-22 10:11:30.111939+00 |   17 | FTS: update role, status, and mode for dbid 17 with contentid 7 to m, u, and s
 2025-09-22 10:11:30.111917+00 |    9 | FTS: update role, status, and mode for dbid 9 with contentid 7 to p, u, and s
 2025-09-22 10:11:30.103092+00 |   15 | FTS: update role, status, and mode for dbid 15 with contentid 5 to m, u, and s
 2025-09-22 10:11:30.103023+00 |    7 | FTS: update role, status, and mode for dbid 7 with contentid 5 to p, u, and s
 2025-09-22 10:11:28.953528+00 |   13 | FTS: update role, status, and mode for dbid 13 with contentid 3 to m, u, and s
 2025-09-22 10:11:28.953505+00 |    5 | FTS: update role, status, and mode for dbid 5 with contentid 3 to p, u, and s
 2025-09-22 10:11:28.941994+00 |   12 | FTS: update role, status, and mode for dbid 12 with contentid 2 to m, u, and s
 2025-09-22 10:11:28.941886+00 |    4 | FTS: update role, status, and mode for dbid 4 with contentid 2 to p, u, and s
(10 rows)
```

Table types

Daria Barysheva

Contents

To Table of Contents

Overview
Heap vs Append-optimized
Row-oriented vs Column-oriented

Overview

ADB (as well as Greengage DB) supports several types of tables with different data storage models:

- **Heap**. This is the default table type. Heap tables are recommended for OLTP-type workloads. You can use them if the data is frequently modified after being initially loaded. It is also the most suitable kind of tables for single-row queries `INSERT`, `UPDATE`, and `DELETE`.
- **Append-optimized (AO)**. These tables are preferable for OLAP-type workloads. They are also optimized for bulk data loading. You can use them if the data is rarely modified after being initially loaded, i.e. the read-only queries prevail. Unlike heap tables (that can be only row-oriented), append-optimized tables support two forms of data orientation:
  - **Row-oriented**. This storage model is recommended for retrieving all or most of the table columns at once.
  - **Column-oriented**. This model is good for data computing over a small number of columns. You can also use it for tables that need regular updates of a small part of the columns — without modifying the rest of them.

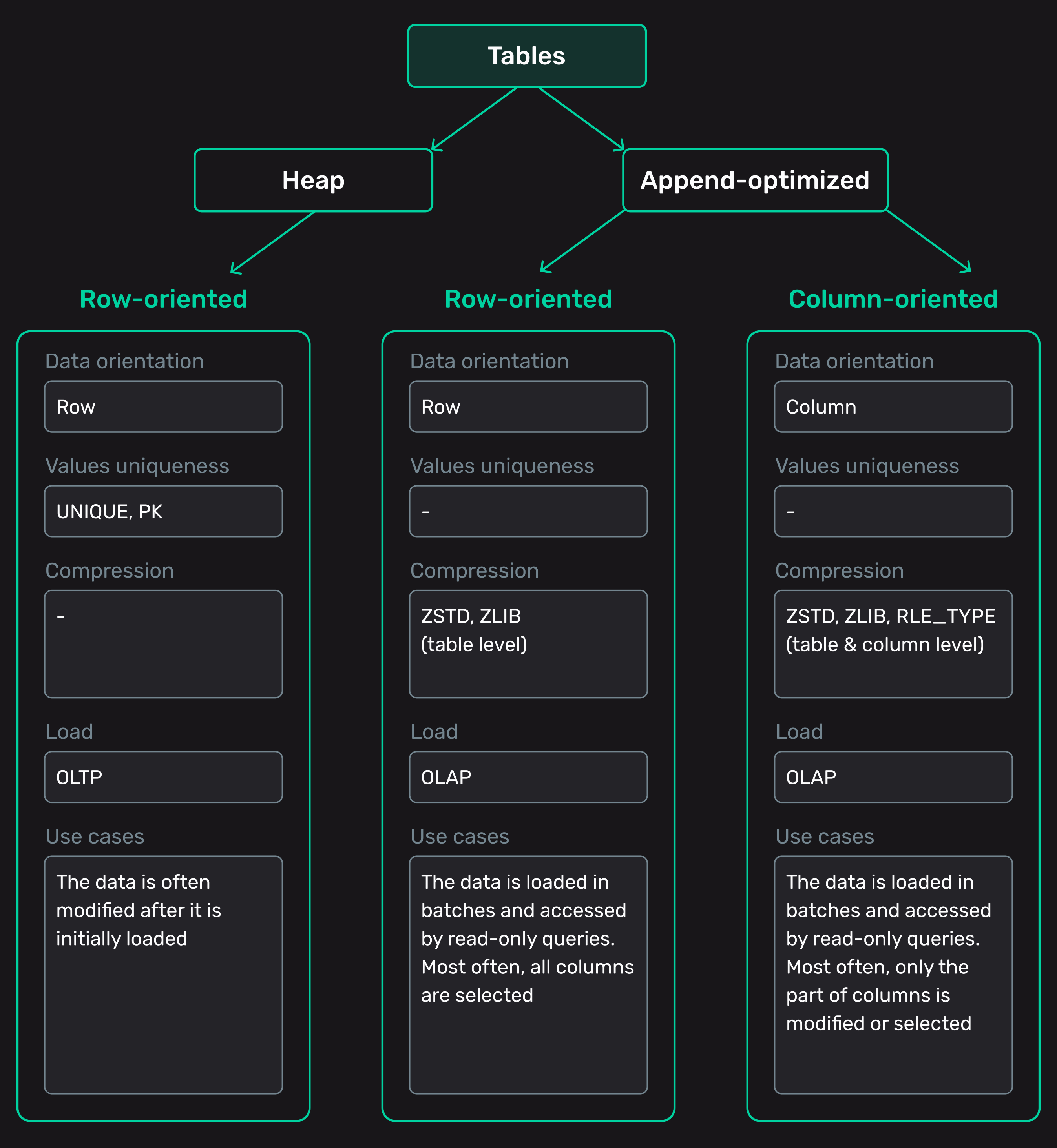

Tables
Heap
Append-optimized
Row-oriented
Row-oriented
Column-oriented
Data orientation
Row
Values uniqueness
UNIQUE, PK
Compression
-
Load
OLTP
Use cases
The data is often modified after it is initially loaded
Data orientation
Row
Values uniqueness
-
Compression
ZSTD, ZLIB (table level)
Load
OLAP
Use cases
The data is loaded in batches and accessed by read-only queries. Most often, all columns are selected
Data orientation
Column
Values uniqueness
-
Compression
ZSTD, ZLIB, RLE_TYPE (table & column level)
Load
OLAP
Use cases
The data is loaded in batches and accessed by read-only queries. Most often, only the part of columns is modified or selected

Table types

When you create a table, you should choose the most suitable table type. Starting with ADB 7, this type is defined by the table **access method** — an interface introduced in PostgreSQL 12.

The following subsections show the difference between the table types that are possible in ADB. This information can be useful for choosing the best storage model for your workloads.

Heap vs Append-optimized

Criteria	Heap	Append-optimized
Load	OLTP	OLAP
Size of tables	Small tables, e.g. dimension tables	Denormalized fact tables that are typically the largest ones in the system
Frequency of data updates after the initial loading	Often	Rarely
Prevailing queries	Many iterative transactions including single-row updates and parallel batch operations	- Loading by batches. - Access by read-only queries.
Ability to define unique values	Supported — by using the `UNIQUE` and `PRIMARY KEY` column constraints	—
Compression	—	- Row-oriented: `ZSTD`, `ZLIB` (table level). - Column-oriented: `ZSTD`, `ZLIB`, `RLE_TYPE` (table and column level).

Syntax

Heap:

Heap tables are created by default:

```
CREATE TABLE test_heap (a int, b text)
    DISTRIBUTED BY (a);
```

Append-optimized:

To create an append-optimized row-oriented table, use the `USING ao_row` clause.

```
CREATE TABLE test_ao_row (a int, b text)
  USING ao_row
  DISTRIBUTED BY (a);
```

To create an append-optimized column-oriented table, use the `USING ao_column` clause:

```
CREATE TABLE test_ao_column (a int, b
text)
  USING ao_column
  DISTRIBUTED BY (a);
```

The legacy `WITH (appendoptimized=true|false, orientation=row|column)` way is also supported

Recommendations on usage	OLTP-type workloads — with many iterative transactions over small tables	Data warehouse environment (OLAP-type workloads) — with bulk loading and subsequent read-only requests over large denormalized tables

CAUTION

It is not recommended to perform the following commands over append-optimized tables due to their inefficiency:

- The single-row `INSERT`, `UPDATE`, or `DELETE` operations.
- The parallel batch `UPDATE` or `DELETE` operations (this does not apply to the batch `INSERT` operations).

Row-oriented vs Column-oriented

Criteria	Row-oriented	Column-oriented
Table type	Heap and append-optimized	Append-optimized
Frequency of data updates after the initial loading	Often	Rarely
Frequency of data insertion	Often	Rarely (column-oriented tables are not optimized for the `INSERT` operations, as column values for a row are written to different places on disk)
Number of columns in the table	Relatively small	Many columns (dozens or more)
Number of requested columns	All or the majority of columns	A small subset of columns

Prevailing queries

Row-oriented: Any

Column-oriented:

- The `SELECT` queries that:
  - Aggregate many values of a single column (the `WHERE` and `HAVING` predicates also use this column):

```
SELECT SUM(price) FROM book;
SELECT SUM(price) FROM book WHERE
price > 1000;
```

  - Return a relatively small number of columns (the `WHERE` predicate uses a single column):

```
SELECT title FROM book WHERE
public_year=2022;
```

- Rare `UPDATE` queries that modify a small subset of columns without modifying other column data.

Compression	Possible at the table level in append-optimized tables. Algorithms: `ZSTD`, `ZLIB`	Possible at the table and the column levels in append-optimized tables. Algorithms: `ZSTD`, `ZLIB`, `RLE_TYPE`

Syntax

Row-oriented:

- Heap:

```
CREATE TABLE test_heap (a int, b text)
    DISTRIBUTED BY (a);
```

- Append-optimized:

```
CREATE TABLE test_ao_row (a int, b text)
  USING ao_row
  DISTRIBUTED BY (a);
```

Column-oriented:

Append-optimized only:

```
CREATE TABLE test_ao_column (a int, b
text)
  USING ao_column
  DISTRIBUTED BY (a);
```

Recommendations on usage	- OLTP-type workloads — with many iterative transactions including frequent inserts and updates. - OLAP-type workloads or mixed workloads — with queries where many columns of a single row are retrieved all at once.	Data warehouse environment (OLAP-type workloads) — with aggregations of many values of a single column or regular selects/updates of a small number of columns

Data compression

Daria Barysheva

Contents

To Table of Contents

Overview

Recommendations on data compression usage

Overview

To improve I/O operations and decrease data size in ADB clusters, you can use data compression. It is available only for append-optimized tables. There are two types of data compression:

- **Table-level compression** — can be applied to an entire table. It is possible both for row-oriented and column-oriented append-optimized tables.
- **Column-level compression** — can be applied to a specific column. You can choose different column-level compression algorithms for different table columns. This data compression type is possible only for column-oriented append-optimized tables.

Compression types

Data orientation	Compression level	Algorithms	Example
Row-oriented	Table	`ZLIB` — deflate algorithm. `ZSTD` — Zstandard algorithm.	Table level only:
Column-oriented	Table and column	`ZLIB` — deflate algorithm. `ZSTD` — Zstandard algorithm. `RLE_TYPE` — run-length encoding.	Table level: Column level:

Row-oriented — Table level only:

```
CREATE TABLE
ao_row_compressed (a int, b
text)
USING ao_row
WITH (compresstype=zlib,
   compresslevel=5)
DISTRIBUTED BY (a);
```

Column-oriented — Table level:

```
CREATE TABLE
ao_col_compressed_table (a
int, b text)
USING ao_column
WITH (compresstype=zlib,
    compresslevel=5)
DISTRIBUTED BY (a);
```

Column-oriented — Column level:

```
CREATE TABLE
ao_col_compressed_column
          (a int,
           b text ENCODING
(compresstype=RLE_TYPE,

compresslevel=1))
USING ao_column
DISTRIBUTED BY (a);
```

Regardless of the level where you define data compression (table or column), you can use the following storage directives. They should be specified in the `WITH` or `ENCODING` clause, accordingly:

- `compresstype` — type of data compression. Possible values: `ZLIB`, `ZSTD`, and `RLE_TYPE`. Values are not case-sensitive. The default value is `none` — no compression.
- `compresslevel` — compression level. Possible values for different compression algorithms are summarized in the table below. Levels with small numbers are the fastest, but they perform the least compression.

IMPORTANT

When an `ENCODING` clause conflicts with a `WITH` clause, the `ENCODING` clause has a higher precedence.

Storage directives for compression

Directive	Deflate	Zstandard	Run-length encoding
compresstype	ZLIB	ZSTD	RLE_TYPE
compresslevel	Possible values: `1` — `9`. `1` is the fastest method with the least compression. It is used by default. `9` is the slowest method with the most compression.	Possible values: `1` — `19`. `1` is the fastest method with the least compression. It is used by default. `19` is the slowest method with the most compression.	Possible values: `1` — apply `RLE_TYPE` only. It is the fastest method with the least compression. It is used by default. `2` — apply `RLE_TYPE`, then apply the `ZLIB` compression with level `1`. `3` — apply `RLE_TYPE`, then apply the `ZLIB` compression with level `5`. `4` — apply `RLE_TYPE`, then apply the `ZLIB` compression with level `9`. It is the slowest method with the most compression. `5` — apply `RLE_TYPE` then apply the `ZSTD` compression with level `1`. `6` — apply `RLE_TYPE` then apply the `ZSTD` compression with level `3`.

Recommendations on data compression usage

There are some recommendations on how to get the maximum performance when using data compression:

- Choose the most suitable compression algorithm:
  - `ZSTD` — provides both speed and a good compression ratio that can be tuned with the `compresslevel` option.
  - `ZLIB` — is usually used for backward compatibility. `ZSTD` outperforms this compression type on usual workloads.
  - `RLE_TYPE` — compresses data better than other algorithms when the same data values occur in many consecutive rows. But this compression type is not useful with tables that do not have large sets of repeated data.
- When specifying the compression level, remember that the bigger that value is — the more (yet slower) the compression is. In fact, data compression allows you to "buy" I/O and disk space for CPU. So you should always balance between the available CPU resources and the required speed of compression. Try to find the optimal settings — to avoid long compression times or slow scan rates.
- The final performance of compressed append-optimized tables depends on many factors, such as hardware, query tuning parameters, etc. You should perform comparison testing to determine whether data compression is useful in your environment.
- It is not recommended to create compressed append-optimized tables on file systems that use compression.

TIP

- Use the `RLE_TYPE` compression type for large sets of repeated data.
- For most other tasks, the `ZSTD` compression with level `1` is the best option (comparable with the `ZLIB` compression with level `5`).

Distribution

Daria Barysheva

Contents

To Table of Contents

Overview of distribution policies

Recommendations on avoiding data skew

Examples

- Correct distribution
- Wrong distribution

Overview of distribution policies

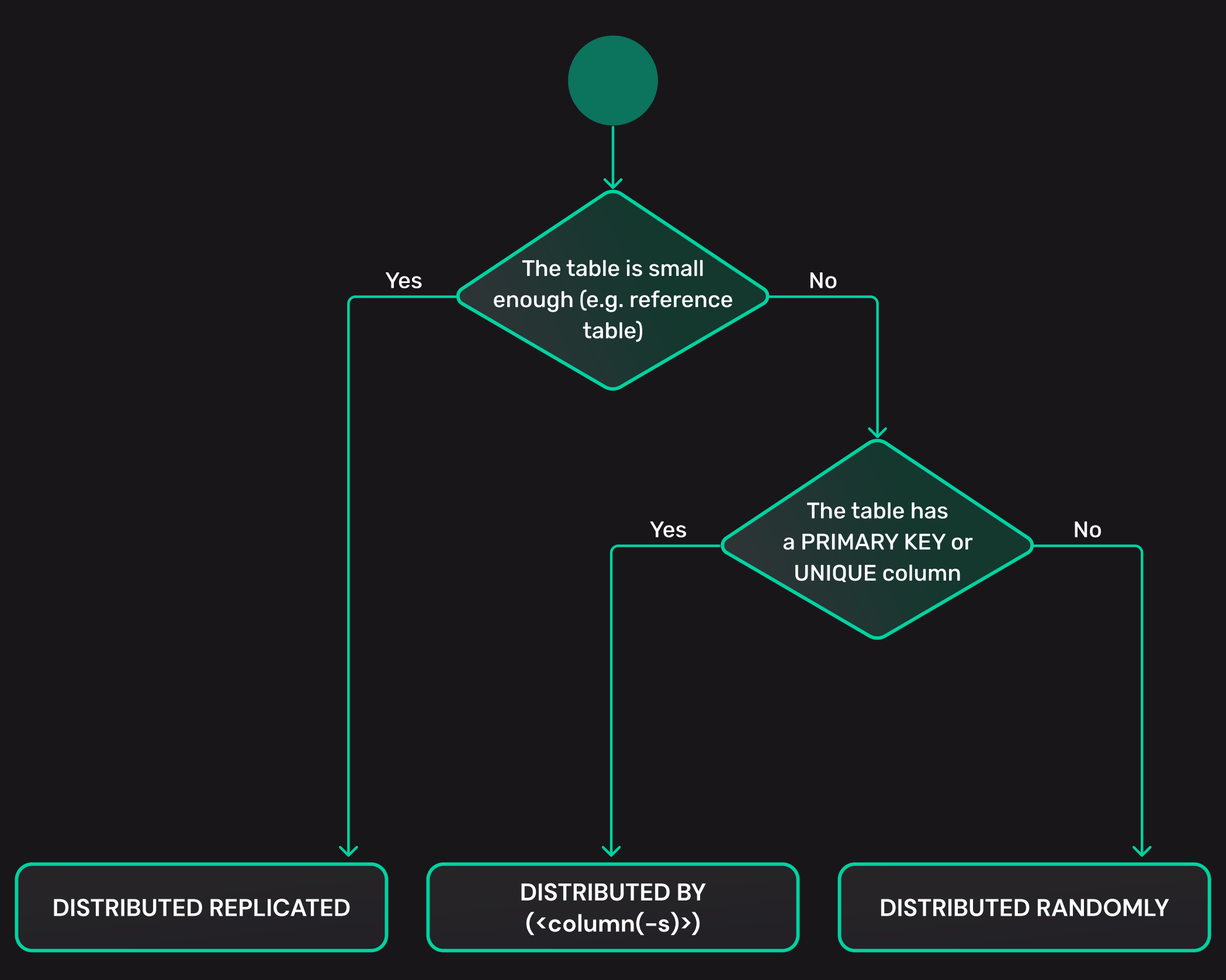
Distribution is one of the most important concepts underlying ADB. It means storing table rows on different segments of the cluster. The more uniform the distribution of the data between segments is — the higher the whole cluster performance is. All segments should contain approximately equal portions of data.

All ADB tables are distributed. When you create or alter a table, you optionally specify one of the available policies — to determine the table row distribution. Distribution policies are listed below.

Distribution policies

Type	Description	Example
DISTRIBUTED BY (<column(-s)>)	A hash distribution. The particular segment is chosen on the basis of hashes. These hashes are calculated for the fields specified in brackets. Recommended for tables that have primary keys or unique columns — these columns can be used as a **distribution key** (the field that is used to choose the segment)	`CREATE TABLE test_distr_key (a INT PRIMARY KEY, b TEXT)` `DISTRIBUTED BY (a);`
DISTRIBUTED REPLICATED	A full distribution. A copy of the table is stored on every cluster segment. Recommended only for small tables (e.g. reference tables). Can be useful to avoid any motion when joining tables	`CREATE TABLE test_distr_replicated (a TEXT, b TEXT)` `DISTRIBUTED REPLICATED;`
DISTRIBUTED RANDOMLY	A random distribution that is based on the round-robin algorithm. Since the system chooses the particular segment randomly, uniformity is not guaranteed. Recommended for other cases — when tables do not have unique columns and the table size is not small	`CREATE TABLE test_distr_random (a TEXT, b INT)` `DISTRIBUTED RANDOMLY;`

The image below shows (in a simplified way) how to choose the appropriate distribution policy.



The table is small enough (e.g. reference table)
Yes
No
The table has a PRIMARY KEY or UNIQUE column
Yes
No
DISTRIBUTED REPLICATED
DISTRIBUTED BY (<column(-s)>)
DISTRIBUTED RANDOMLY

Choosing the distribution policy

Recommendations on avoiding data skew

For the best possible performance, all cluster segments should contain equal portions of data. Otherwise, the **skew** can occur. The skew means the data becomes unbalanced and the segments with more data start to work harder to perform their part of the query processing. To avoid the skew and make the data distribution more efficient, follow the recommendations listed below:

1. Always explicitly specify the distribution policy when creating a table. If the `DISTRIBUTED` clause is missed, the table distribution policy is controlled by the `gp_create_table_random_default_distribution` configuration parameter. That can lead to choosing the first suitable column as the distribution key, unless the parameter is explicitly set to `ON`. See Greenplum documentation for more information.
2. Avoid selecting the following non-uniformly distributed data fields as distribution keys:
   - Dates and timestamps.
   - The fields that contain a lot of the `NULL` values.
   - Custom data type fields with the unknown subsequent distribution.

   Instead, try to choose a distribution key that is unique for each record, such as the primary key or any unique column without `NULL` values.

   You can also specify custom hash functions for a hash distribution. To do this, fill in the `<opclass>` argument in the `DISTRIBUTED BY` clause. It defines an operator class name. For more details on syntax, see Tables.
3. Make sure that the data is uniformly distributed after the initial and incremental loads. To obtain the segment where the current table row is stored, you can run the following query:

```
SELECT gp_segment_id, * FROM test_distr_key;
```

   The result can be similar to:

```
 gp_segment_id | a |   b
---------------+---+-------
             4 | 1 | first
(1 row)
```

   The `gp_segment_id` system field corresponds to the `content` column of the `pg_catalog.gp_segment_configuration` table that contains information about the cluster segments. This field is hidden by default. It can be obtained for all tables except the tables with the `DISTRIBUTED REPLICATED` distribution policy (as in this case the data is duplicated on all segments).

> **CAUTION**
>
> It is not recommended to request the `gp_segment_id` field for large tables on a regular basis.

4. If necessary, you can change the distribution type using one of the following ways.
   - Changing the distribution policy:

```
ALTER TABLE test_distr_replicated SET DISTRIBUTED RANDOMLY;
```

   - Redistribution without changing the distribution key:

```
ALTER TABLE test_distr_key SET WITH (REORGANIZE=TRUE);
```

5. The correct data distribution also affects the speed of queries execution, especially the `JOIN` queries. The main recommendations on data querying are:
   - Do not choose the fields used in the `WHERE` clauses as distribution keys. For the best performance, all segments should handle an equal share of the query workload. But if the `WHERE` predicate in a query references a distribution key, the query processing work is concentrated on only one segment that stores all the rows with the requested distribution key value.
   - Remember that joining tables by their distribution keys is the fastest way to join tables due to identical values being always stored on a single segment. When tables share a common distribution key, the work of joining or sorting on this key is done locally — without distributing data tuples across the segments. Local operations are always faster than distributed ones.
   - Use the `DISTRIBUTED REPLICATED` policy for small, frequently used tables. That improves query performance by making it unnecessary to broadcast a table to all segments (so-called broadcast motion).
   - Choose the same data types for the fields that are used to join tables.

Examples

Correct distribution

1. Create two tables as shown below:

```
CREATE TABLE t1 (id INT, msg TEXT) DISTRIBUTED BY(id);
CREATE TABLE t2 (id INT, msg TEXT) DISTRIBUTED BY(id);
```

2. Insert some data into the tables:

```
INSERT INTO t1 (id, msg) VALUES (1, 'text1'), (2, 'text2');
INSERT INTO t2 (id, msg) VALUES (1, 'text3'), (2, 'text4');
```

3. Check that in both tables the rows with the same distribution key value are stored on one segment:

```
SELECT gp_segment_id, * FROM t1;
```

   Result:

```
 gp_segment_id | id |  msg
---------------+----+-------
             4 |  1 | text1
             9 |  2 | text2
(2 rows)
```

```
SELECT gp_segment_id, * FROM t2;
```

   The result is shown below. You can see that segment identifiers are similar to the listed above:

```
 gp_segment_id | id |  msg
---------------+----+-------
             9 |  2 | text4
             4 |  1 | text3
(2 rows)
```

4. Use the `EXPLAIN` statement to see the plan of joining two tables.

```
EXPLAIN SELECT * FROM t1 JOIN t2 ON t1.id = t2.id;
```

   Result:

```
                                  QUERY PLAN
---------------------------------------------------------------------------------
 Gather Motion 16:1  (slice1; segments: 16)  (cost=0.00..862.00 rows=2 width=20)
   ->  Hash Join  (cost=0.00..862.00 rows=1 width=20)
         Hash Cond: (t1.id = t2.id)
         ->  Seq Scan on t1  (cost=0.00..431.00 rows=1 width=10)
         ->  Hash  (cost=431.00..431.00 rows=1 width=10)
               ->  Seq Scan on t2  (cost=0.00..431.00 rows=1 width=10)
 Optimizer: GPORCA
(7 rows)
```

   The command output shows that no data motions are needed for this query, except `Gather Motion` — which means a standard process of sending the data from segments to coordinator.

Wrong distribution

Mismatched distribution keys on table join

1. Create one additional table as shown below:

```
CREATE TABLE t3 (id INT, msg TEXT) DISTRIBUTED BY(msg);
```

2. Insert some data into the table:

```
INSERT INTO t3 (id, msg) VALUES (1, 'text5'), (2, 'text6');
```

3. List the segments that store the table data:

```
SELECT gp_segment_id, * FROM t3;
```

   Result:

```
 gp_segment_id | id |  msg
---------------+----+-------
             2 |  1 | text5
            10 |  2 | text6
(2 rows)
```

   You can see that the segments are not similar to those that were shown for the `t1` table in the previous example.

4. Use the `EXPLAIN` statement to see the plan of joining the `t3` and `t1` tables.

```
EXPLAIN SELECT * FROM t1 JOIN t3 ON t1.id = t3.id;
```

   Result:

```
                                              QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Gather Motion 16:1  (slice2; segments: 16)  (cost=0.00..862.00 rows=2 width=20)
   ->  Hash Join  (cost=0.00..862.00 rows=1 width=20)
         Hash Cond: (t1.id = t3.id)
         ->  Seq Scan on t1  (cost=0.00..431.00 rows=1 width=10)
         ->  Hash  (cost=431.00..431.00 rows=2 width=10)
               ->  Broadcast Motion 16:16  (slice1; segments: 16)  (cost=0.00..431.00 rows=2 width=10)
                     ->  Seq Scan on t3  (cost=0.00..431.00 rows=1 width=10)
 Optimizer: GPORCA
(8 rows)
```

   In comparison with the previous example, `Broadcast Motion` is added. This operation means that the table copy is sent to all segments — as they have no necessary data for joining. This is not critical for small tables, but in case of joining large tables this can lead to significant performance problems. To fix it in the given example, change the distribution key in the `t3` table from `msg` to `id`.

Not unique field as a key

1. Create an additional table whose distribution key is different from the distribution key of the `t1` table created earlier, as shown below:

```
CREATE TABLE t4 (id INT, msg TEXT) DISTRIBUTED BY(id);
```

2. Insert some data into the table. Notice that the first column values are not unique:

```
INSERT INTO t4 (id, msg) VALUES (1, 'text7'), (1, 'text8');
```

3. List the segments that store the table data:

```
SELECT gp_segment_id, * FROM t4;
```

   Since the table uses hash distribution, the same input values (of the `id` column) cause the hash function to output the same segment identifier:

```
 gp_segment_id | id |  msg
---------------+----+-------
             4 |  1 | text7
             4 |  1 | text8
(2 rows)
```

   You can see that the same segment is used for two rows as they have the same value of the distribution key. You should avoid such situations as they cause the workload to increase on single segments. To fix it in the given example, choose the unique field as the distribution key. If this option is impossible, change the distribution policy to the `DISTRIBUTED RANDOMLY` value.

Tablespaces

Daria Barysheva

Contents

To Table of Contents

- Overview
- View existing tablespaces
- Create a new tablespace
- Use a tablespace to work with database objects
  - Create objects
  - Check tablespaces
  - Move objects
  - Move temporary objects
- Drop a tablespace

Overview

Tablespace is a system object that allows administrators to locate database entities (such as tables, indexes, spill files, etc.) in different directories or even on different disks. In other words, tablespaces tell where to place the physical files for logical objects. Due to using tablespaces, it is possible to have multiple file systems per machine and choose the optimal physical storage for different objects. For example, tablespaces can be used to place frequently-used tables on file systems based on the high-performance solid-state drives (SSD), and place other tables (e.g. storing archived data) on the standard hard drives (HDD).

A tablespace is a global object, so you can use it from any database — if you have the appropriate privileges.

There are several default tablespaces in every ADB cluster. These tablespaces are added after the DBMS is initialized. See their description in the table below.

Default tablespaces

Name	Description
pg_global	Used by shared system catalogs
pg_default	Used by other database objects. For example, the `template0` and `template1` databases use it. Therefore, this is also the default tablespace for other databases, unless overridden by the `TABLESPACE` clause in the `CREATE DATABASE` command

CAUTION

It is not recommended to drop or modify the tablespaces listed above.

View existing tablespaces

To view the list of available tablespaces in the current ADB cluster, you can run the following commands:

1. Connect to the ADB coordinator host (via SSH, etc.) and log in under the default user `gpadmin`.

```
$ sudo su - gpadmin
```

2. Connect to the database via `psql` (or any other client program).

```
$ psql adb
```

The output is:

```
psql (12.12)
Type "help" for help.
```

3. Perform one of the following commands to list all existing tablespaces:

   - Run the SQL query against the `pg_catalog.pg_tablespace` catalog table.

```
SELECT oid, * FROM pg_catalog.pg_tablespace;
```

The result is listed below. The `oid` field corresponds to the unique tablespace identifier.

```
 oid  |  spcname   | spcowner | spcacl | spcoptions
------+------------+----------+--------+------------
 1663 | pg_default |       10 |        |
 1664 | pg_global  |       10 |        |
(2 rows)
```

   - If you work in the `psql` client program, you can use the `\db+` meta-command to show the list of tablespaces.

```
\db+
```

Result:

```
                                  List of tablespaces
    Name    |  Owner  | Location | Access privileges | Options |  Size   | Description
------------+---------+----------+-------------------+---------+---------+-------------
 pg_default | gpadmin |          |                   |         | 1086 MB |
 pg_global  | gpadmin |          |                   |         | 41 MB   |
(2 rows)
```

Pay attention to the `Location` field in the output of the last command:

- For **user-defined tablespaces**, this field shows the tablespace physical location.
- For **default tablespaces**, this field is empty. The location of the default tablespaces is the same as the data directory configured when the database cluster is initialized. To get it, you can show the value of the server configuration parameter `data_directory`.

```
SHOW data_directory;
```

Result:

```
    data_directory
----------------------
 /data1/master/gpseg-1
(1 row)
```

Create a new tablespace

To create a new tablespace, it is necessary to use the `CREATE TABLESPACE` command, write the tablespace name, and then define the file system location where this tablespace should be stored — after the `LOCATION` keyword.

IMPORTANT

There are some requirements to the location that is chosen for a new tablespace:

- It should exist on all cluster hosts — including coordinator, standby coordinator, and every segment host.
- It should be empty.
- It should be owned by the system user `gpadmin`.
- It should not be on removable or transient storage. If you lose a tablespace (due to the directory removing, disk failure, etc.), the whole cluster can become unreadable.

The following example creates the new tablespace `newspace` in the */newspace* directory.

```
CREATE TABLESPACE newspace LOCATION '/newspace';
```

The output should be similar to:

```
CREATE TABLESPACE
```

Now, if you run any command to list all tablespaces, you get the following:

- SQL query:

```
SELECT * FROM pg_tablespace;
```

```
  oid  |  spcname   | spcowner | spcacl | spcoptions
-------+------------+----------+--------+------------
  1663 | pg_default |       10 |        |
  1664 | pg_global  |       10 |        |
 28508 | newspace   |       10 |        |
(3 rows)
```

- `psql` command:

```
\db+
```

```
                                  List of tablespaces
    Name    |  Owner  |   Location    | Access privileges | Options |  Size   | Description
------------+---------+---------------+-------------------+---------+---------+------------
 newspace   | gpadmin | /newspace     |                   |         | 0 bytes |
 pg_default | gpadmin |               |                   |         | 1086 MB |
 pg_global  | gpadmin |               |                   |         | 41 MB   |
(3 rows)
```

Internally, within the *pg_tblspc* subfolder of the main data directory, ADB stores all user-defined tablespaces as symlinks. To check it, log out from `psql` and run the following command. Use your own data directory instead of */data1/master/gpseg-1/*:

```
$ ls -l /data1/master/gpseg-1/pg_tblspc
```

The result is listed below. `28508` is the tablespace OID.

```
total 0
lrwxrwxrwx 1 gpadmin gpadmin 11 Sep  7 08:43 28508 -> /newspace/1
```

NOTE

To create a new tablespace, you should be a superuser. After that you can allow ordinary users to use the added tablespace. To do that, grant them the `CREATE` privilege.

Use a tablespace to work with database objects

Create objects

You can define tablespaces for databases, tables, and indexes. To do this, use the `TABLESPACE` option when running the `CREATE` command for these types of objects. Some examples are listed below:

- **Databases**. The following command creates a new database `test_space_db` in the `newspace` tablespace. This tablespace becomes the default one for all system catalogs of that database and for all tables and indexes created within the database (unless the `TABLESPACE` clause is given for them). If the `TABLESPACE` clause is not defined in the `CREATE DATABASE` command, a new database uses the same tablespace as the template database it is copied from (`pg_default`, by default).

```
CREATE DATABASE test_space_db TABLESPACE newspace;
```

Result:

```
CREATE DATABASE
```

You can connect to the created database via `psql`:

```
\c test_space_db
```

Result:

```
You are now connected to database "test_space_db" as user "gpadmin".
```

- **Tables**:
  - If you do not explicitly specify a tablespace for a table, the database's default tablespace is used. For example, the following command creates the `test_space_t1` table in the `newspace` tablespace.

```
CREATE TABLE test_space_t1(a INT, b TEXT) DISTRIBUTED BY(a);
```

Result:

```
CREATE TABLE
```

  - You can also specify another tablespace for a table — other than the selected one for the database. The command below creates the `test_space_t2` table in the `pg_default` tablespace:

```
CREATE TABLE test_space_t2(a INT, b TEXT) TABLESPACE pg_default DISTRIBUTED BY(a);
```

Result:

```
CREATE TABLE
```

- **Indexes**. As with tables, you can create indexes either in the default database's tablespace or in the specified one.
  - Using the default database's tablespace:

```
CREATE INDEX test_space_idx2 ON test_space_t1 (b);
```

Result:

```
CREATE INDEX
```

  - Using the explicitly specified tablespace:

```
CREATE INDEX test_space_idx ON test_space_t2 (b) TABLESPACE pg_default;
```

The result is shown below:

```
CREATE INDEX
```

You can also set the server configuration parameter `default_tablespace` — to specify the default tablespace for all subsequent `CREATE TABLE` and `CREATE INDEX` commands that do not explicitly specify a tablespace. It does not apply to the `CREATE DATABASE` command, as for databases the default tablespace is always copied from the template database (if the `TABLESPACE` option is not defined). The following example shows how to set the parameter on the fly within a database session, but as for all session parameters, you can define its value at the role, the database, and the system levels too.

```
SET default_tablespace = newspace;
```

Check tablespaces

In order to check whether the tablespace is defined for database objects correctly, you can use the following commands:

- **Databases**. To see the list of databases with their tablespaces, you can run the `psql` command `\l+`:

```
\l+
```

The command result is listed below. Pay attention to the `tablespace` column.

```
                                                         List of databases
     Name      |  Owner  | Encoding |  Collate   |   Ctype    |  Access privileges  |  Size  | Tablespace |
Description
---------------+---------+----------+------------+------------+---------------------+--------+------------+---------------------
---------------------
 adb           | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =Tc/gpadmin        +| 422 MB | pg_default |
               |         |          |            |            | gpadmin=CTc/gpadmin |        |            |
 diskquota     | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |                     | 283 MB | pg_default |
 postgres      | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |                     | 280 MB | pg_default | default
administrative connection database
 template0     | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +| 276 MB | pg_default | unmodifiable empty
database
               |         |          |            |            | gpadmin=CTc/gpadmin |        |            |
 template1     | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 | =c/gpadmin         +| 278 MB | pg_default | default template for
new databases
               |         |          |            |            | gpadmin=CTc/gpadmin |        |            |
 test_space_db | gpadmin | UTF8     | en_US.utf8 | en_US.utf8 |                     | 284 MB | newspace   |
(6 rows)
```

- **Tables**. To see the tablespace defined for the specified table, you can run the following query against the `pg_catalog.pg_tables` table:

```
SELECT * FROM pg_catalog.pg_tables WHERE tablename='test_space_t2';
```

Result:

```
 schemaname |   tablename   | tableowner | tablespace | hasindexes | hasrules | hastriggers | rowsecurity
------------+---------------+------------+------------+------------+----------+-------------+-------------
 public     | test_space_t2 | gpadmin    | pg_default | t          | f        | f           | f
(1 row)
```

- **Indexes**. To see the tablespace defined for the specified index, you can run the following query against the `pg_catalog.pg_indexes` table:

```
SELECT * FROM pg_catalog.pg_indexes WHERE tablename = 'test_space_t2';
```

Result:

```
 schemaname |   tablename   |   indexname    | tablespace |                               indexdef
------------+---------------+----------------+------------+-----------------------------------------------------------------------
 public     | test_space_t2 | test_space_idx | pg_default | CREATE INDEX test_space_idx ON public.test_space_t2 USING btree (b)
(1 row)
```

Specifics for the default database's tablespace

The `pg_catalog.pg_tables` and `pg_catalog.pg_indexes` tables do not contain any data in the `tablespace` column for those tables and indexes that use the default database's tablespace. The `tablespace` column in such cases is empty. This ensures that if the database is cloned with a different default tablespace, the `pg_tables` and `pg_indexes` entries will still match the location where the `CREATE DATABASE` will put the physically copied relation.

- `pg_catalog.pg_tables`:

```
SELECT * FROM pg_catalog.pg_tables WHERE tablename='test_space_t1';
```

Result:

```
 schemaname |   tablename   | tableowner | tablespace | hasindexes | hasrules | hastriggers | rowsecurity
------------+---------------+------------+------------+------------+----------+-------------+-------------
 public     | test_space_t1 | gpadmin    |            | f          | f        | f           | f
(1 row)
```

- `pg_catalog.pg_indexes`:

```
SELECT * FROM pg_catalog.pg_indexes WHERE tablename = 'test_space_t1';
```

Result:

```
 schemaname |   tablename   |    indexname    | tablespace |                                indexdef
------------+---------------+-----------------+------------+---------------------------------------------------------------------
--------
 public     | test_space_t1 | test_space_idx2 |            | CREATE INDEX test_space_idx2 ON public.test_space_t1 USING
btree (b)
(1 row)
```

Move objects

After a tablespace is defined for a table or an index, it can be changed into a different tablespace. Examples are listed below:

- **Tables:**
  - To move the specified table into another tablespace, use the following query:

```
ALTER TABLE test_space_t1 SET TABLESPACE pg_default;
```

Result:

```
ALTER TABLE
```

  - To move all tables in the current database from one tablespace into another, run the command:

```
ALTER TABLE ALL IN TABLESPACE newspace SET TABLESPACE pg_default;
```

The result is shown below. It can also contain the `No matching relations in tablespace` message if there are no any tables for the current database in the specified tablespace.

```
ALTER TABLE
```

CAUTION

Be careful when using `ALTER TABLE ALL`. This command moves not only user-defined tables but also system ones (such as `pg_toast.*`). If in the future you need to move the current database tables to another tablespace — you should again use the `ALL` statement.

- **Indexes:**
  - To move the specified index into another tablespace, use the following query:

```
ALTER INDEX test_space_idx2 SET TABLESPACE pg_default;
```

Result:

```
ALTER INDEX
```

  - To move all indexes in the current database from one tablespace into another, run the command:

```
ALTER INDEX ALL IN TABLESPACE newspace SET TABLESPACE pg_default;
```

The result is shown below. It can also contain the `No matching relations in tablespace` message if there are no any indexes for the current database in the specified tablespace.

```
ALTER INDEX
```

CAUTION

Note that system catalogs are not moved by the `ALTER TABLE ALL` and `ALTER INDEX ALL` commands. To move system catalogs, use `ALTER DATABASE` or explicit `ALTER TABLE` / `ALTER INDEX` commands instead. The `information_schema` relations are not considered as a part of system catalogs and will be moved.

- **Databases.** To change the default tablespace at the database level, you can use the `ALTER DATABASE` command with the `SET TABLESPACE` clause. This command physically moves all tables and indexes stored in the database's old default tablespace to the new tablespace. Pay attention that tables and indexes stored in non-default tablespaces (which have been explicitly defined) are not affected. The following example creates a new tablespace and then assigns it to the database:
  - Create the new tablespace `newspace2` in the */newspace2* directory:

```
CREATE TABLESPACE newspace2 LOCATION '/newspace2';
```

  - Connect to any other database (for example, `adb`):

```
\c adb
```

  - Change the default tablespace for the `test_space_db` database via the following command:

```
ALTER DATABASE test_space_db SET TABLESPACE newspace2;
```

Result:

```
ALTER DATABASE
```

CAUTION

- To perform the `ALTER DATABASE` command, you should be the database owner or a superuser.
- You should have the `CREATE` privilege for the new tablespace.
- Before changing a tablespace for the currently open database, connect to any other database.
- There should be no objects of the current database (including system tables) in the destination tablespace.

Move temporary objects

Tablespaces are also used to store temporary objects:

- Temporary tables and indexes that are added explicitly via the `CREATE TEMP TABLE` command.
- Temporary tables and spill files that are created automatically for query execution (including hash aggregates, hash joins, and sorting large datasets).
- Temporary tables that are created automatically when refreshing the materialized view data.

The server configuration parameter `temp_tablespaces` allows you to control the location of all these temporary objects. This parameter can contain a comma-separated list of tablespace names, rather than only one — in such case a random list member is picked each time a temporary object is to be created. You can use this feature to spread the load associated with temporary objects over multiple tablespaces.

The default parameter value is an empty string, which corresponds to the default tablespace of the current database. As for all session parameters, you can define the `temp_tablespaces` value for the current session, role, database, or all system. The following command shows how to set the parameter within a database session:

```
SET temp_tablespaces = newspace;
```

The output looks like this:

```
SET
```

Drop a tablespace

To remove a tablespace, it is necessary to use the `DROP TABLESPACE` command and write the tablespace name:

```
DROP TABLESPACE newspace;
```

The output can be similar to:

```
DROP TABLESPACE
```

CAUTION

- To drop a tablespace, you should be the tablespace owner or a superuser.
- You can drop only an empty tablespace. Make sure that none of the database objects use the tablespace. Also, the tablespace selected for removing should not store any temporary or transaction files.

Arenadata DB Control overview

Daria Barysheva

Contents

To Table of Contents

Features
Architecture

Features

Arenadata DB Control (ADB Control) is a real-time Arenadata DB query monitoring system.

ADB Control allows you to perform the following functions:

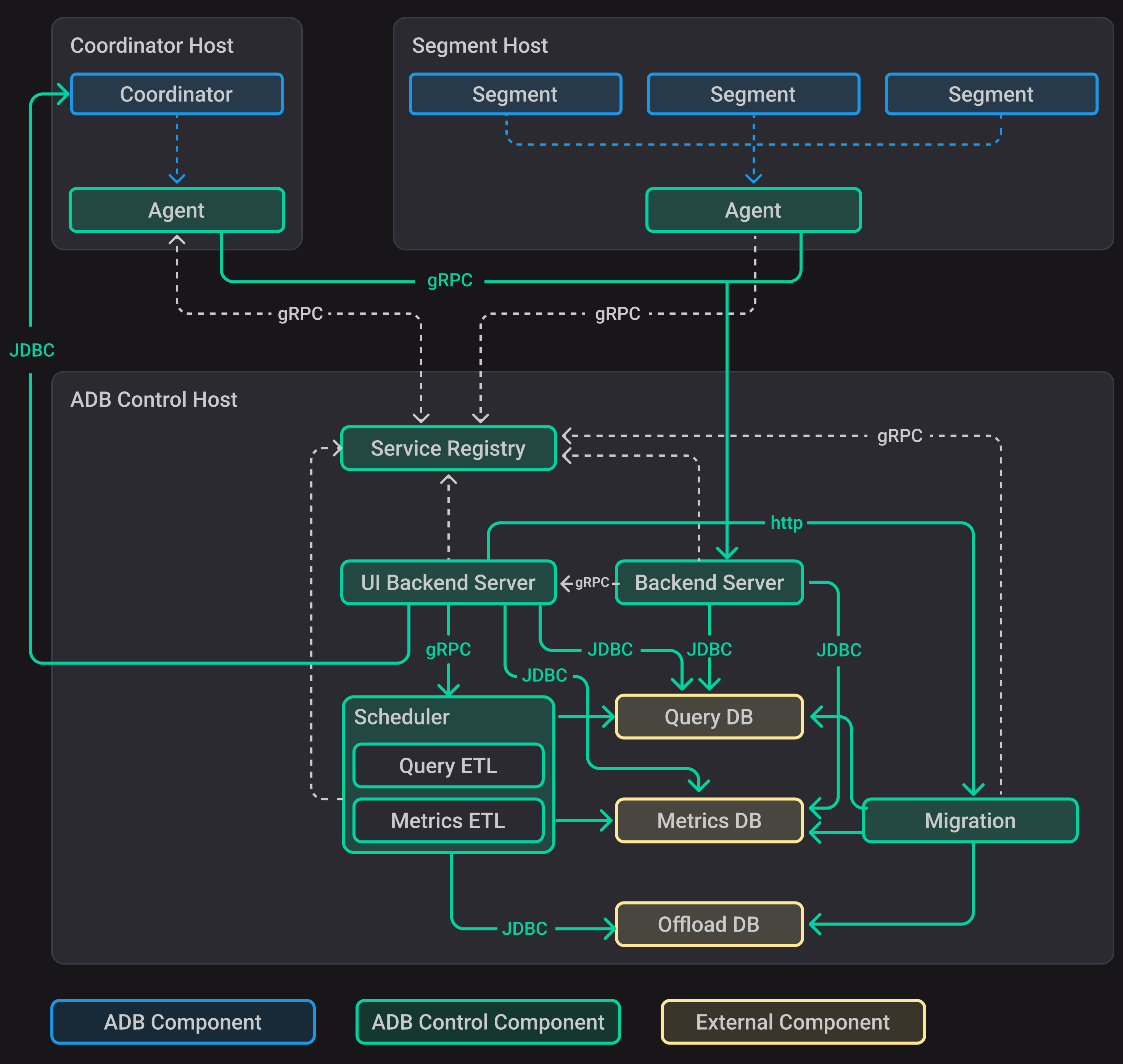
- Work with multiple ADB clusters.
- Monitor SQL commands and transactions, including inner queries.
- Monitor active and idle sessions with the ability to terminate them.
- View information on system resource consumption at the query and transaction levels.
- Build different charts for tracking the dynamic changes of system metrics over the selected time period.
- Cancel running transactions.
- View a query plan with the ability to track the execution progress.
- Offload ADB Control metrics and events to an external database (PostgreSQL/ADB).
- Audit ADB relation accesses (in clusters used for monitoring), as well as user authorizations and other operations in ADB Control.

NOTE

- ADB Control is available in the ADB Enterprise Edition.
- Currently, ADB Control can be installed only offline.
- To use ADB Control, you need to install Chrony. Time synchronization is required for proper work of Scheduler job triggers and accuracy of system metric calculations.

Architecture

The high-level architecture view of ADB Control is shown below.



ADB Control architecture

ADB Control architecture is based on the following components:

- **Agent**. Application for receiving and filtering metrics from the ADB coordinator and segments. Agents are installed one per ADB cluster host, i.e. one agent collects metrics from all segments within a single host.
- **Service Registry**. Registry of ADB Control services. Responsible for service discovery in ADB Control. Due to Service Registry, ADB Control discovers available agents (there is no pre-configured agent map in ADB Control). And agents, in turn, find necessary ADB Control services to send metrics. Since ADB Control can work with multiple ADB clusters at the same time — when you add a cluster, new agents are also registered via Service Registry.
- **UI Backend Server**. Backend server of the ADB Control web interface. Handles requests from Backend Server.
- **Backend Server**. Horizontally scalable service for processing and aggregating metrics from agents. It also writes query/metric information to the Query DB/Metrics DB databases and sends updates to the UI Backend Server.
- **Scheduler**. Runs system jobs on a schedule:
    - Cleans up old metrics and other data in ADB Control.
    - Offloads ADB Control metrics and audit events to an external database (Offload DB).
    - Keeps command and transaction statuses up to date.
    - Aggregates information on system resource consumption.
- **Migration**. Provides automatic migration of schemas and data in databases with which ADB Control interacts (Query DB, Metrics DB, Offload DB).
- **Query DB**. PostgreSQL database that performs the following functions:
    - Query logging.
    - Transaction logging.
    - Storage of system metric aggregates by transactions and queries.
    - Audit.
    - Statistics on database relations.
    - Storage of ADB Control system tables with information on users, clusters, system settings, and so on.
- **Metrics DB**. ClickHouse database that is used to store the following information:
    - System metrics obtained from agents.
    - History of resource consumption within resource groups (based on `gp_resgroup_*`).
- **Offload DB**. External PostgreSQL/ADB database that is used to offload metrics and audit data from ADB Control.

External storage usage

- By default, Query DB and Metrics DB are deployed via the Docker containers. You can connect both of them to ADB Control as external storage systems. To do this, fill in the ***External database parameters*** (for Query DB) and/or ***External ADQM parameters*** (for Metrics DB) section parameters when configuring the ADB Control service during ADB installation (for more details, see Configuration parameters → ADB Control).
- Starting with ADB Control 4.7.5, you need to create users in external databases and grant them permissions manually. For more details, see Configure external databases for ADB Control.

Configure external databases for ADB Control

Daria Barysheva

Contents

To Table of Contents

PostgreSQL
ClickHouse

For production environments, it is recommended to configure external PostgreSQL and ClickHouse databases to store ADB Control data (see Query DB and Metrics DB in Arenadata DB Control overview) for a number of reasons:

- You will be able to set up external storages on hosts of required size for desired performance and configure high availability for them.
- The ADB Control recovery process will be easier and consist mainly of the service configuration steps.

Starting with ADB Control 4.7.5, to configure external databases, you need to follow the steps that are described below.

PostgreSQL

1. Perform the following queries in PostgreSQL. The queries should be run under a superuser:

```
CREATE ROLE <user_name> LOGIN PASSWORD '<user_password>';
ALTER ROLE <user_name> SET search_path TO public, adcc;
CREATE DATABASE <database_name> WITH owner <user_name>;
```

2. Connect to the created database `<database_name>` under a superuser and run:

```
CREATE SCHEMA adcc;
ALTER SCHEMA adcc OWNER TO <user_name>;

CREATE EXTENSION IF NOT EXISTS pgcrypto WITH schema public;
CREATE EXTENSION IF NOT EXISTS btree_gin WITH schema public;
CREATE EXTENSION IF NOT EXISTS pg_trgm WITH schema public;

GRANT SELECT ON ALL TABLES IN SCHEMA pg_catalog TO <user_name>;

CREATE OPERATOR adcc.^| (PROCEDURE = jsonb_exists_any, LEFTARG = jsonb, RIGHTARG = _text, RESTRICT = contsel, JOIN = contjoinsel);
ALTER OPERATOR adcc.^|(jsonb, text[]) OWNER TO <user_name>;
CREATE OPERATOR CLASS adcc.jsonb_ops_custom
FOR TYPE jsonb USING gin AS
OPERATOR 7 pg_catalog.@>(jsonb, jsonb),
OPERATOR 10 adcc.^| (jsonb, _text),
FUNCTION 1  gin_compare_jsonb(text, text),
FUNCTION 2  gin_extract_jsonb(jsonb, internal, internal),
FUNCTION 3  gin_extract_jsonb_query(jsonb, internal, smallint, internal, internal, internal, internal),
FUNCTION 4  gin_consistent_jsonb(internal, smallint, jsonb, integer, internal, internal, internal, internal),
FUNCTION 6  gin_triconsistent_jsonb(internal, smallint, jsonb, integer, internal, internal, internal);
ALTER OPERATOR CLASS adcc.jsonb_ops_custom USING GIN OWNER TO <user_name>;
```

where `<user_name>` is a name of the previously created user.

> If you can assign superuser rights to a PostgreSQL user (which will be used for connecting to an external database from ADB Control), it is enough to create a role and a database in PostgreSQL before installation or reconfiguration of ADB Control:
>
> ```
> CREATE ROLE <user_name> LOGIN PASSWORD '<user_password>' SUPERUSER;
> ALTER ROLE <user_name> SET search_path TO public, adcc;
> CREATE DATABASE <database_name> WITH owner <user_name>;
> ```

3. Optionally, you can run the additional query to increase performance by disabling synchronous writes to WAL files:

```
ALTER DATABASE <database_name> SET synchronous_commit = OFF;
```

where `<database_name>` is a name of the created database.

4. Provide the ADB Control access to the external PostgreSQL database by adding the following record to the *pg_hba.conf* file on the PostgreSQL server.

Without SSL

```
host    <database_name>  <user_name>       <adbc_address>      trust
```

where:

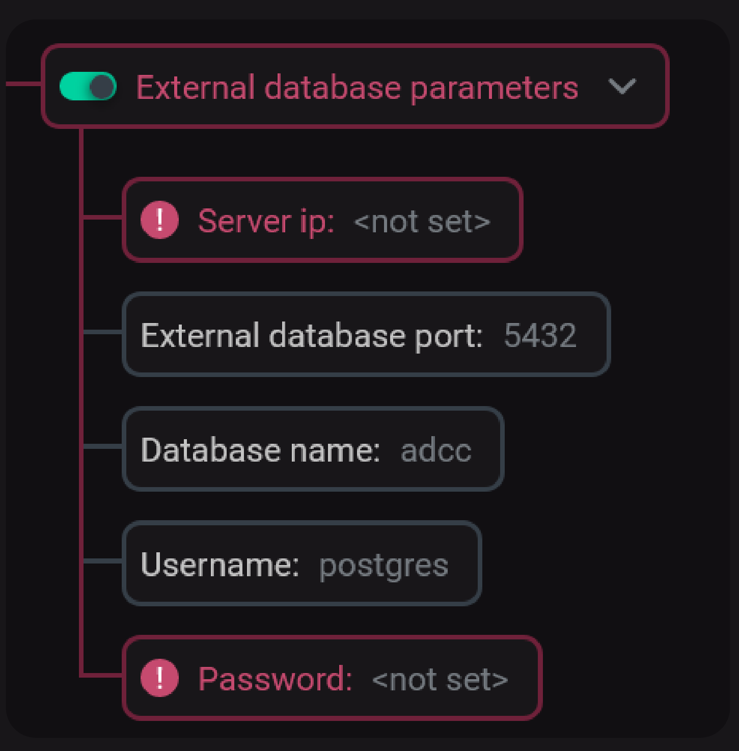
- `<database_name>` — a name of the created database.
- `<user_name>` — a name of the created user.
- `<adbc_address>` — an IP address of ADB Control with a subnet number.

With SSL

```
hostssl    all  all       0.0.0.0/0      md5
```

For more information on how to configure PostgreSQL for using SSL, see SSL encryption.

5. On the ADB Control service configuration page, activate the ***External database parameters*** toggle and fill in all parameters in the list that opens. The assignment of parameters is described in the Configuration parameters → ADB Control → External database parameters section.



External database parameters
Server ip: <not set>
External database port: 5432
Database name: adcc
Username: postgres
Password: <not set>

Parameters for connecting to the external PostgreSQL DB

6. Click ***Save*** to save the ADB Control configuration.
7. Install the ADB Control service using the Install service action or apply the ***Reconfigure & Restart*** action to the ADB Control service if it is already installed.

ClickHouse

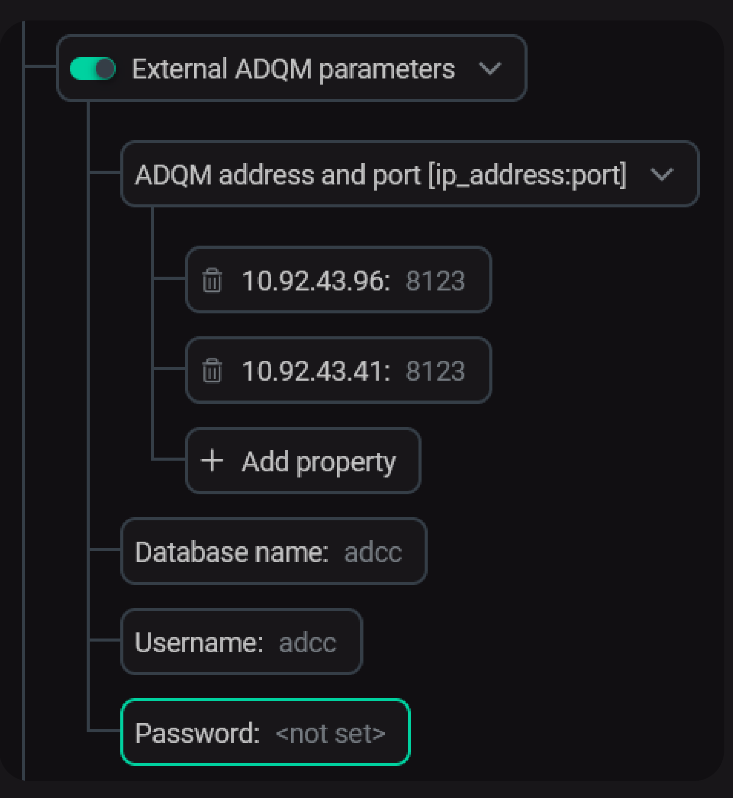
IMPORTANT

- To use clustered external ClickHouse as an external database (instead of a standalone node), you should first create a logical cluster called `adcc_cluster` in ClickHouse.
- If you use SSL encryption in ADB Control, you need to configure SSL on the ClickHouse side according to the SSL encryption article and define the `8443` port in the ClickHouse connection parameters (see ***External ADQM parameters*** below).

1. Run the following queries to create a user with appropriate permissions in ClickHouse:

```
CREATE USER <user_name> NOT IDENTIFIED;
GRANT SELECT, INSERT, ALTER, CREATE, DROP, TRUNCATE ON *.* TO <user_name>
```

2. On the ADB Control service configuration page, activate the ***External ADQM parameters*** toggle and fill in all parameters in the list that opens. The assignment of parameters is described in the Configuration parameters → ADB Control → External ADQM parameters section.



External ADQM parameters
ADQM address and port [ip_address:port]
10.92.43.96: 8123
10.92.43.41: 8123
Database name: adcc
Username: adcc
Password: <not set>

Parameters for connecting to the external ClickHouse DB

3. Click ***Save*** to save the ADB Control configuration.
4. Install the ADB Control service using the Install service action or apply the ***Reconfigure & Restart*** action to the ADB Control service if it is already installed.

Connect to ADB Control

Daria Barysheva

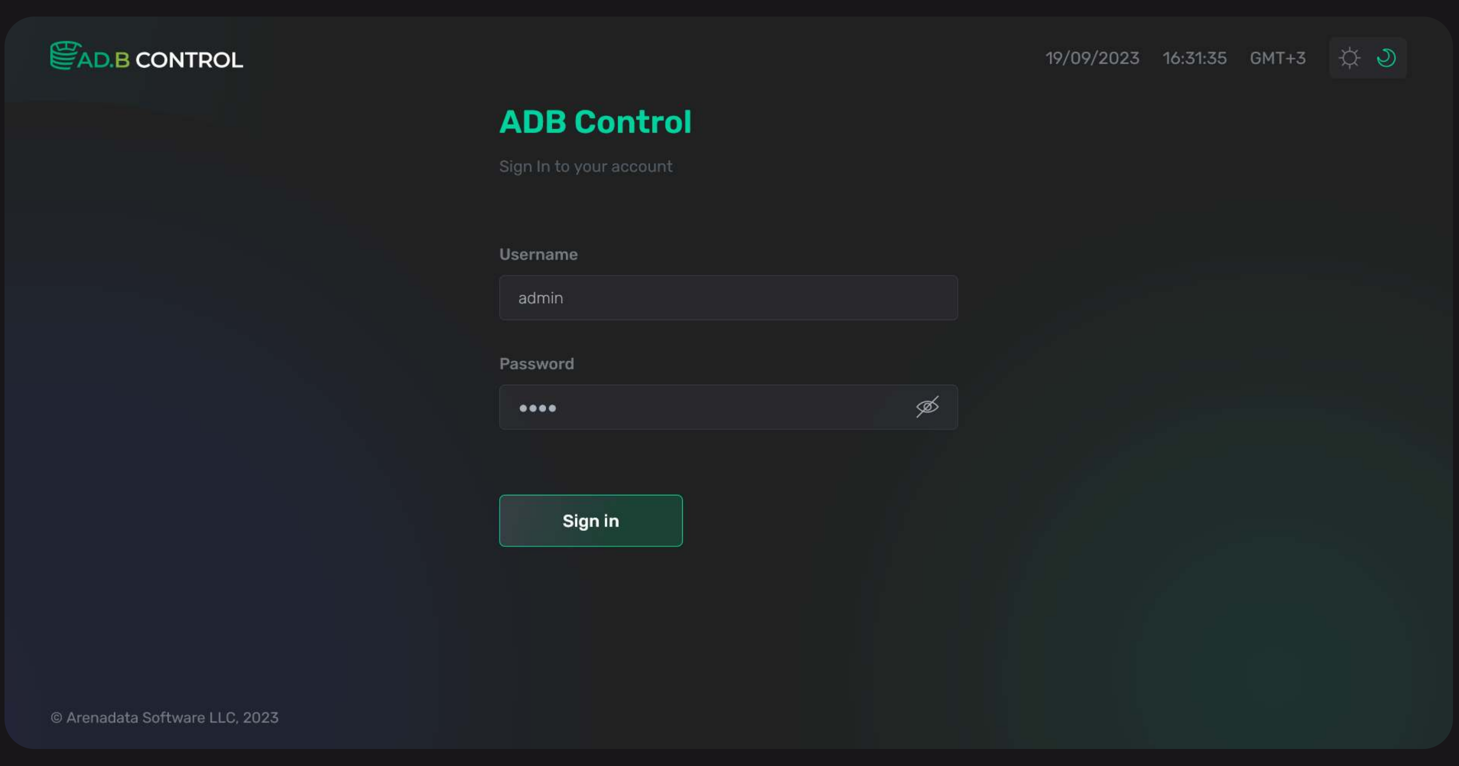
To Table of Contents

To connect to the ADB Control web interface, follow the steps:

1. In the browser bar, enter the IP address of the host where ADB Control is deployed. To specify the port number, use the value of the Listening port parameter from the ***UI component parameters*** section defined when configuring the ADB Control service during the ADB cluster installation (`81` by default).

2. In the window that opens, enter the default user credentials and click ***Sign in***:

   - ***Username:*** admin
   - ***Password:*** 1234



ADB Control

Sign In to your account

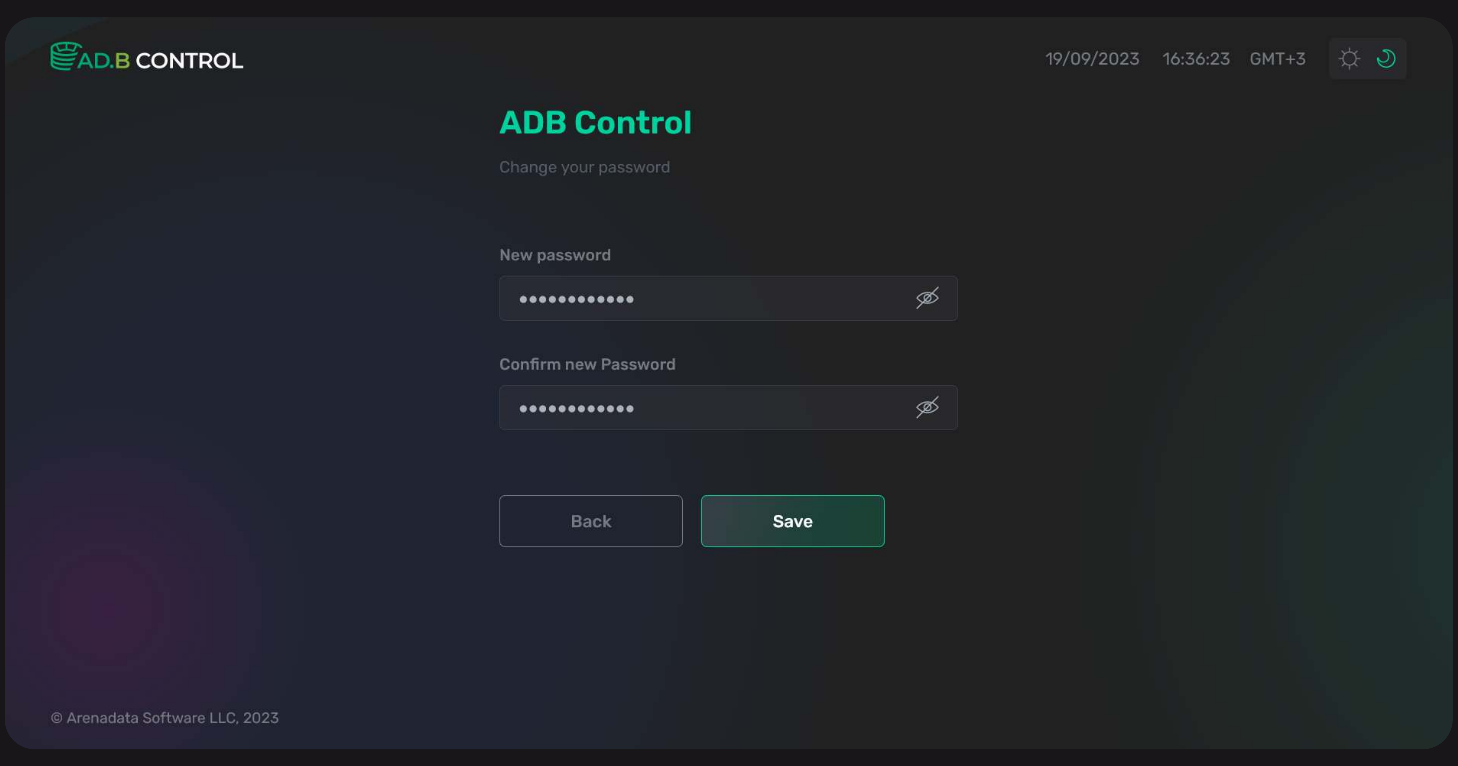
Username: admin

Password: ••••

19/09/2023 16:31:35 GMT+3
Sign in
© Arenadata Software LLC, 2023

Log in to ADB Control

3. When you first log in, you should change a password for the `admin` user. Enter a new password in the ***New password*** field and confirm it in the ***Confirm new Password*** field. Click ***Save***. Note that your password must be greater than or equal to 12 characters by default. Any characters other than spaces are allowed to enter.



ADB Control

Change your password

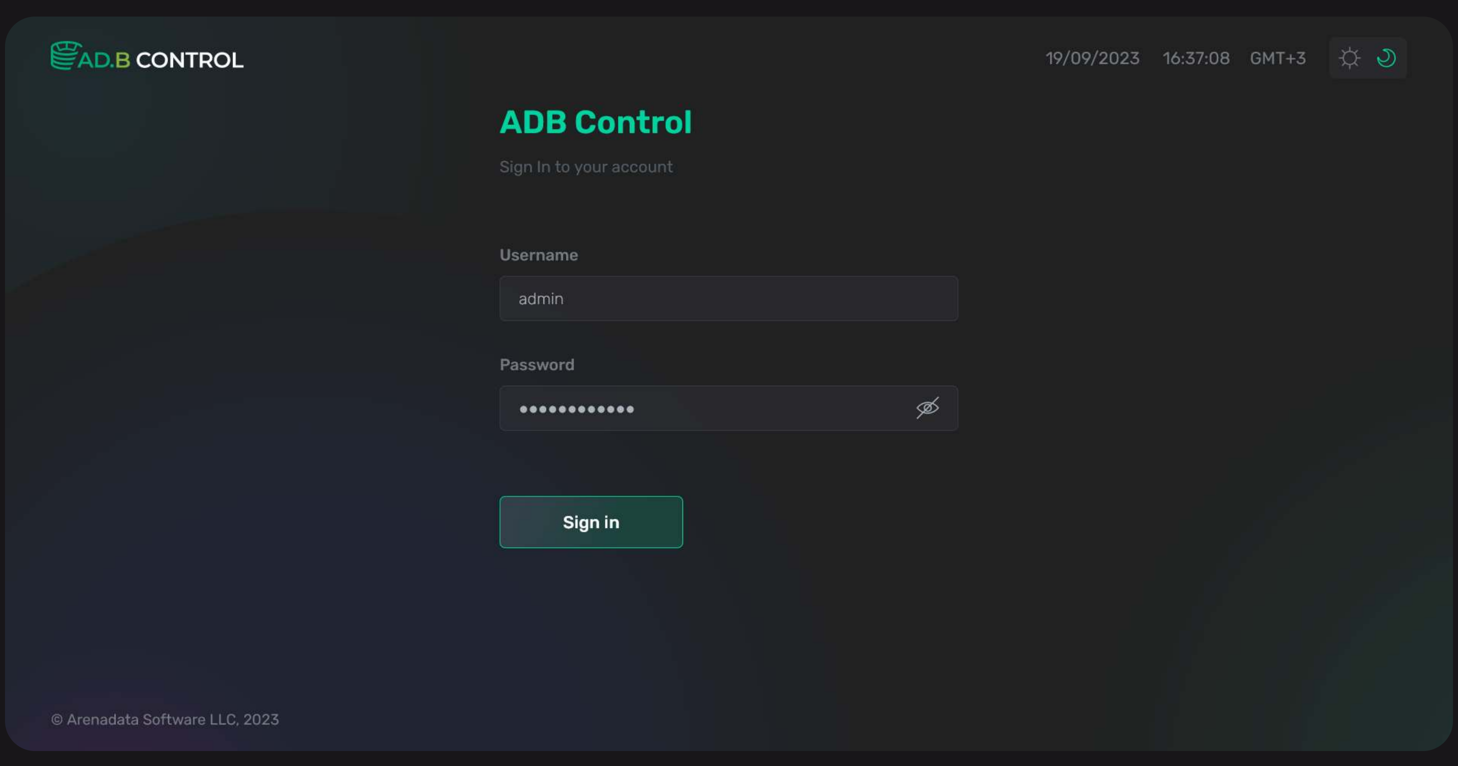
New password: ••••••••••••

Confirm new Password: ••••••••••••

19/09/2023 16:36:23 GMT+3
Back
Save
© Arenadata Software LLC, 2023

Change a password

4. After saving a password, re-enter your login and a new password. Note that when a certain number of unsuccessful login attempts is reached, the user will be blocked — permanently or temporarily, depending on the settings.



ADB Control

Sign In to your account

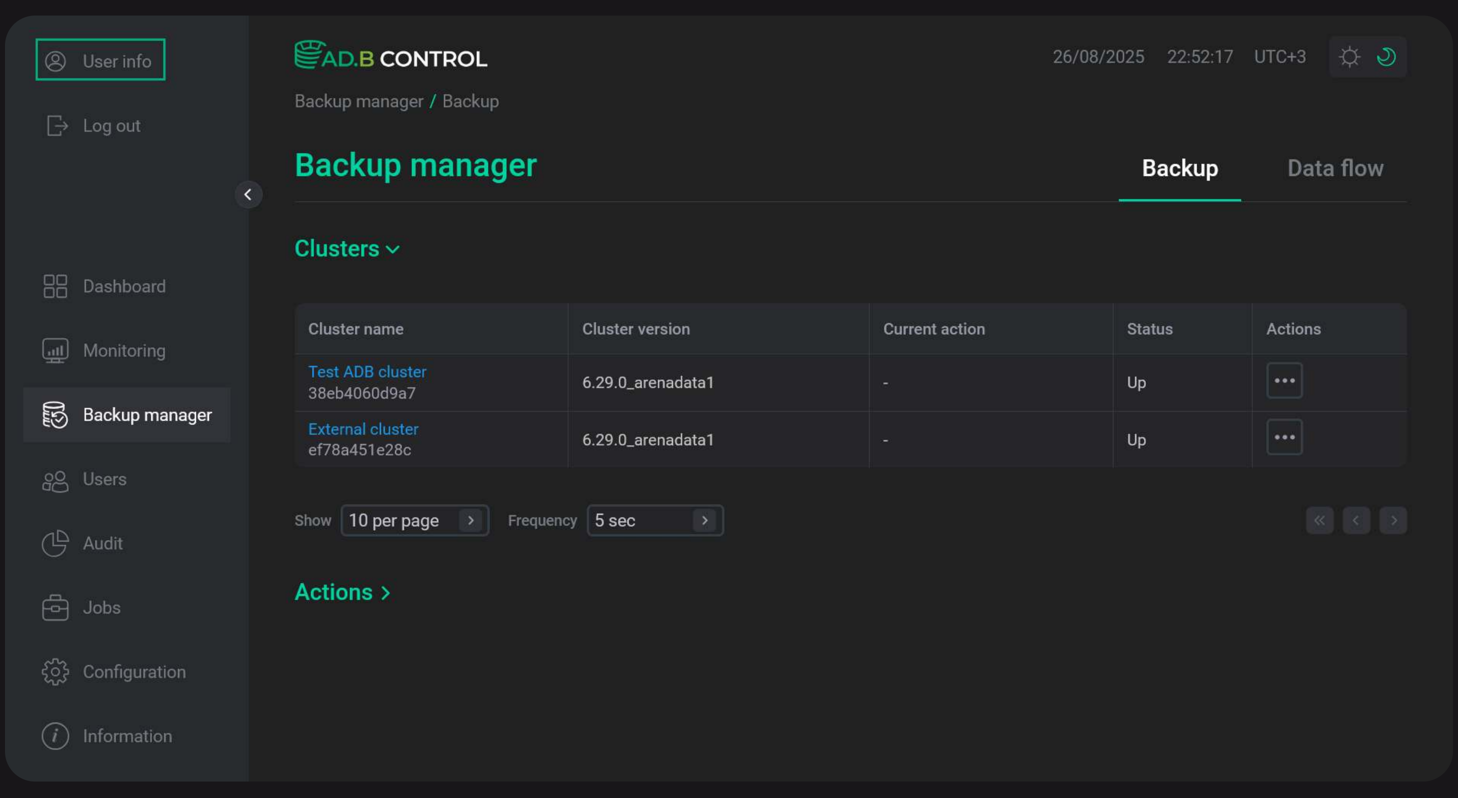
Username: admin

Password: ••••••••••••

19/09/2023 16:37:08 GMT+3
Sign in
© Arenadata Software LLC, 2023

Log in to ADB Control with a new password

5. If you later need to get information about the current user, select the ***User info*** tab in the left navigation menu.



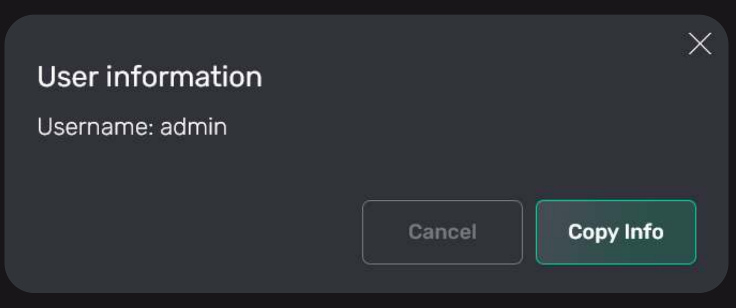
User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
26/08/2025 22:52:17 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show: 10 per page Frequency: 5 sec

View user information

The modal window that opens displays the current user name, which can be copied by clicking ***Copy info***.



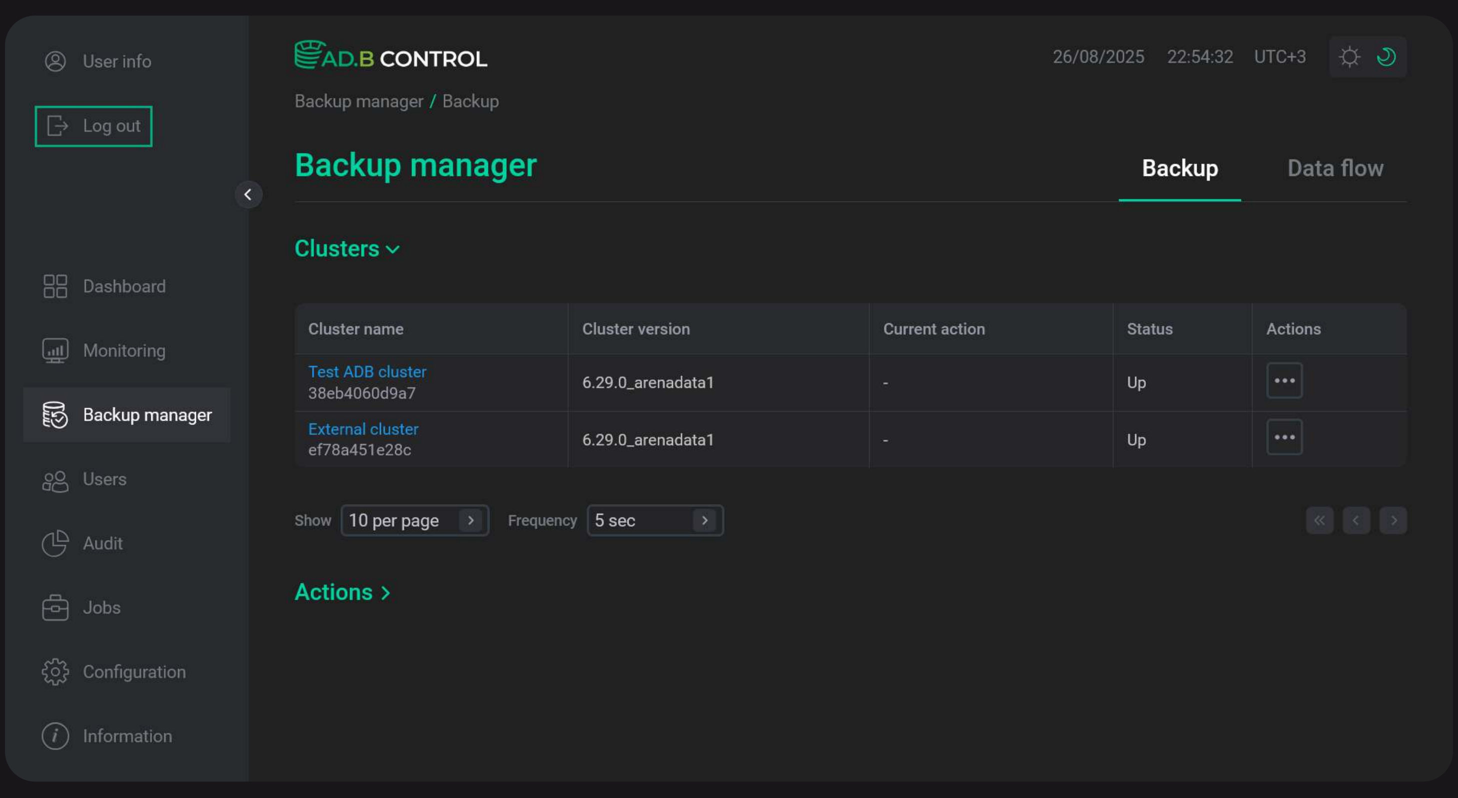
User information

Username: admin

Cancel
Copy info

Copy user information

6. To log out from ADB Control, click ***Log out*** in the left navigation menu.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
26/08/2025 22:54:32 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show: 10 per page Frequency: 5 sec

Log out from ADB Control

NOTE

- To get information on how to add new users and configure access rights for them, refer to the following articles:
  - Basic authentication
  - LDAP authentication
  - Authorization
- You can change security policies (password length limits, authorization attempt limit, blocking rules) on the ***Configuration*** → ***Security*** page. For more information, see Configure security policies.

Basic authentication

Daria Barysheva

Contents

To Table of Contents

View a list of users
Add a user
Edit a user
Block a user
Unblock a user
Delete a user

When basic authentication is used, users log in to ADB Control under the accounts previously created for them in the system. On the creation step, every user is assigned a role, which defines a set of permissions in ADB Control.

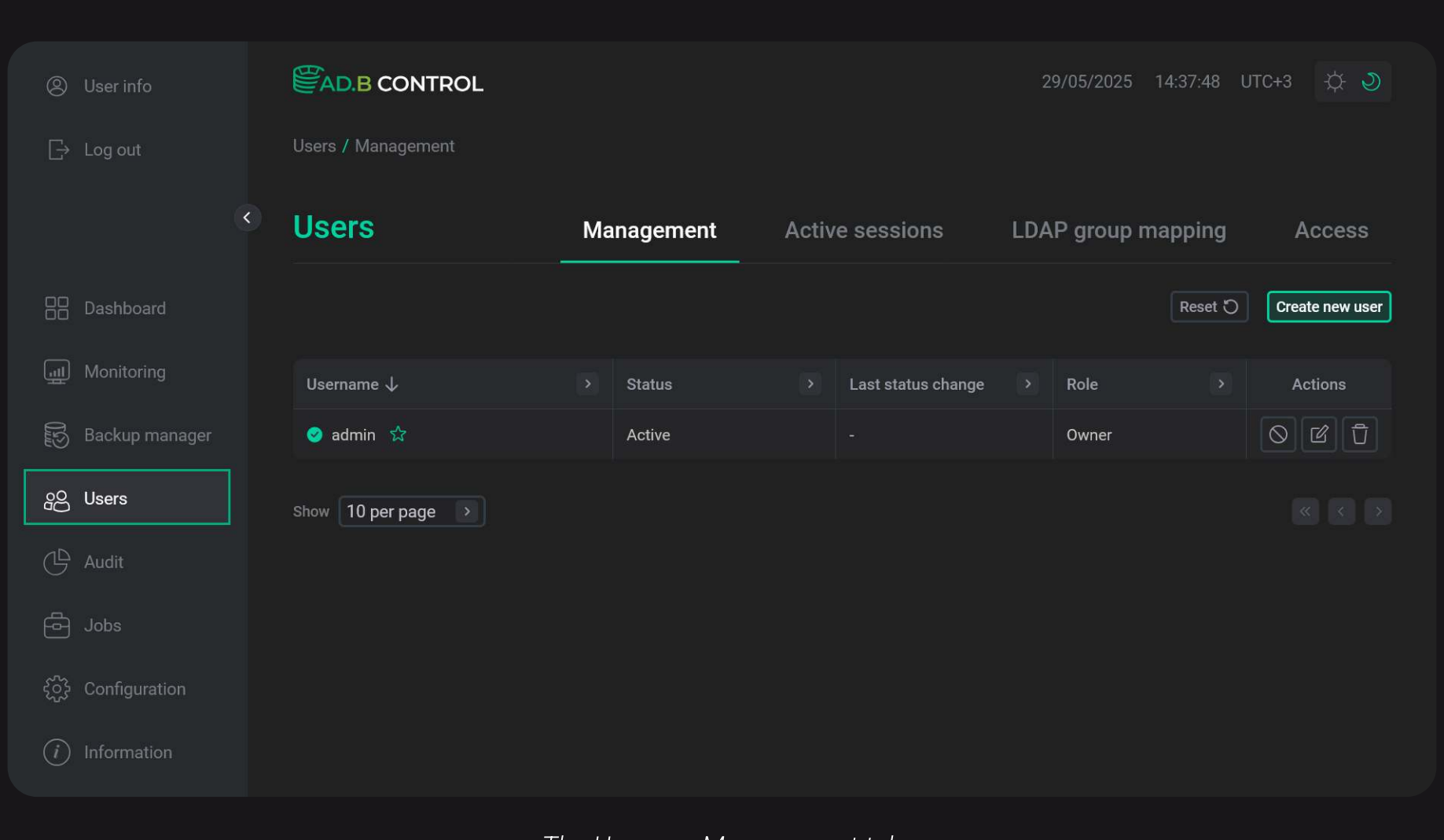
To manage users, you can use the ***Management*** tab on the ***Users*** page in the ADB Control web interface.

IMPORTANT

The operations listed below are available only to users with the `Owner` role. For more information on roles and permissions, see Authorization.

View a list of users

The ***Users*** → ***Management*** tab displays the following information on users.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
29/05/2025 14:37:48 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	

The Users → Management tab

The "Users → Management" tab fields

Field	Description
Username	A user name. Depending on the user role and status (see below), the column can contain the following icons: ✅ — user is active; ⛔ — user is blocked; 🟡 — user is temporarily blocked; ☆ — user has the `Owner` role.
Status	A user status. Possible values: ■ `Active` — user is active. ■ `Blocked` — user is blocked. Users get this status after being blocked manually or automatically according to the security policies. ■ `Temporary blocked` — user is temporarily blocked according to the security policies.
Last status change	The last status change timestamp (see ***Status*** above) in the `DD/MM/YYYY HH:mm:ss` format
Role	A user role. Possible values: ■ `ADBM Administrator` — ADBM administrator. ■ `Administrator` — ADB Control administrator. ■ `Advanced User` — user with advanced permissions. ■ `Owner` — system owner. ■ `Regular User` — ordinary user.

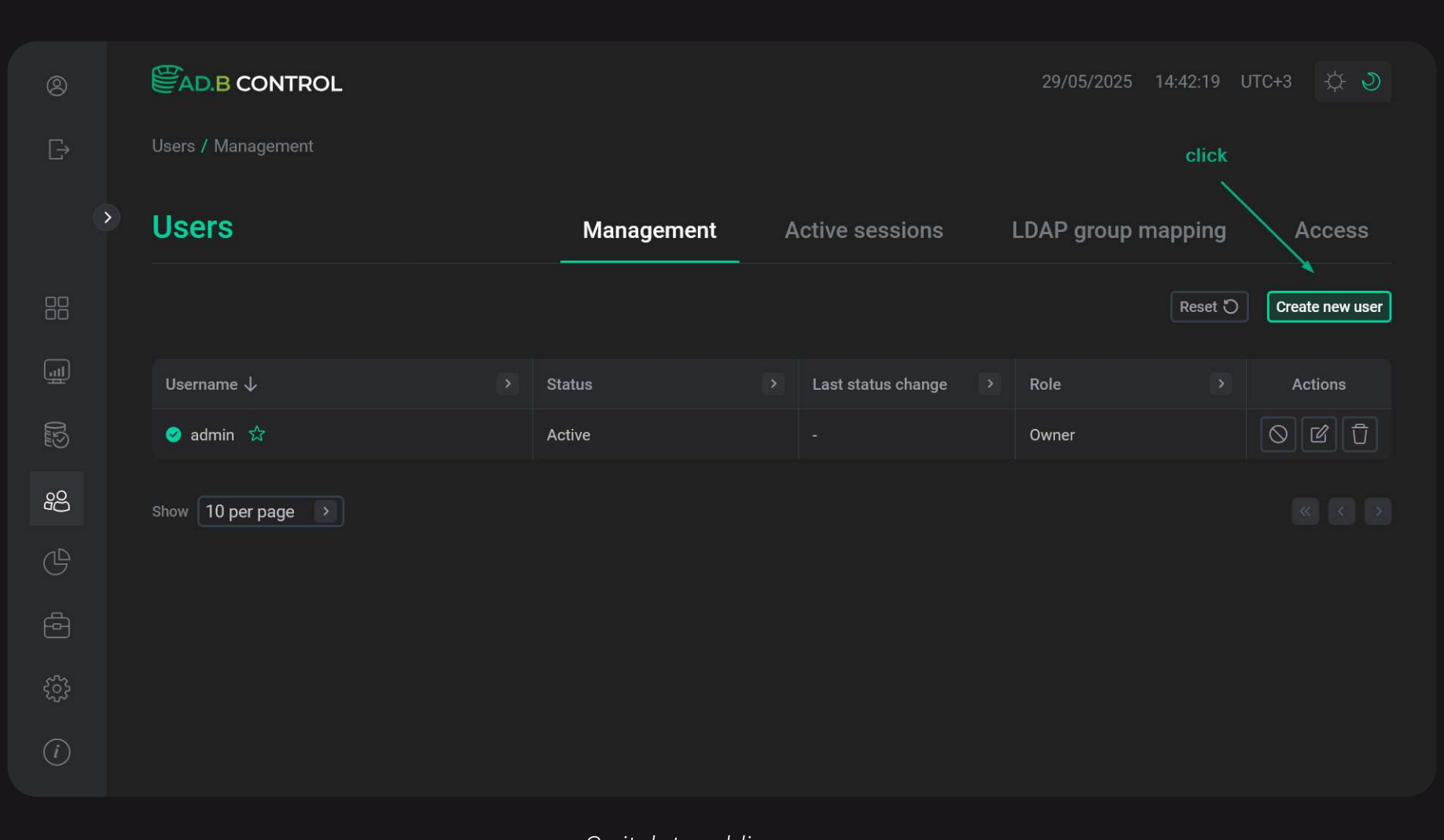
In the column headers of the table with a list of users, there are filters that you can use to select specific data. To open a filter, click the › icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Username***), the search value should be entered. For columns that show date and time (e.g. ***Last status change***), the time range can be selected from the calendar.

The ▽ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Add a user

To add a new user, follow the steps:

1. Click ***Create new user*** on the ***Users*** → ***Management*** tab.



AD.B CONTROL
29/05/2025 14:42:19 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
click
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	

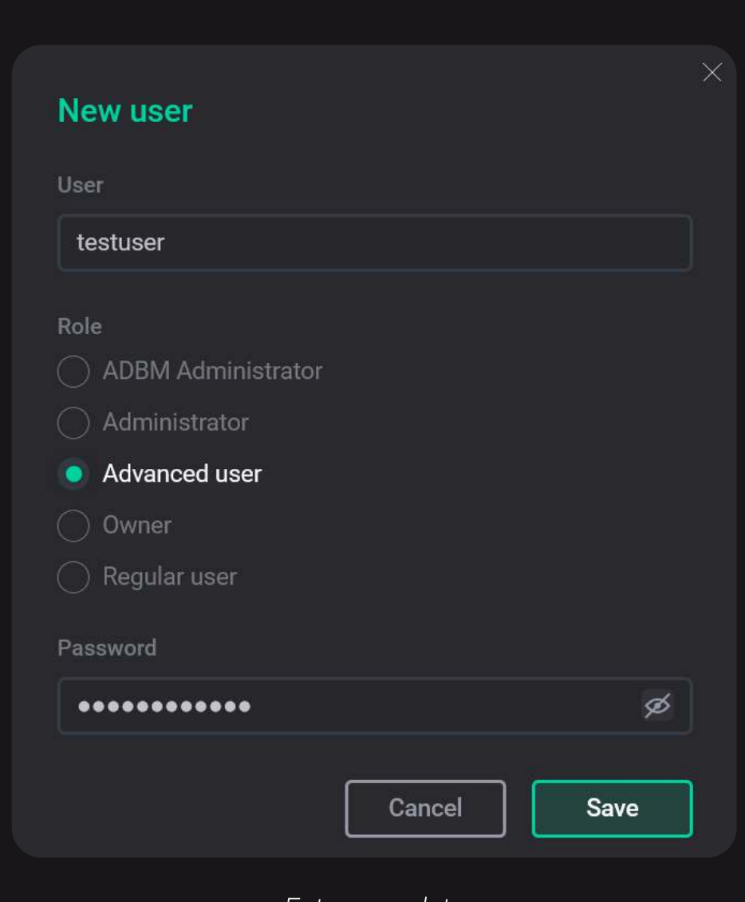
Switch to adding a user

2. In the window that opens, fill in the following fields:

- ***User*** — user name;
- ***Role*** — user role;
- ***Password*** — user password.

TIP

For information on login and password requirements, you can hover the mouse over the ? icon, which appears to the right of the ***User*** and ***Password*** fields during their filling.



New user
Cancel
Save

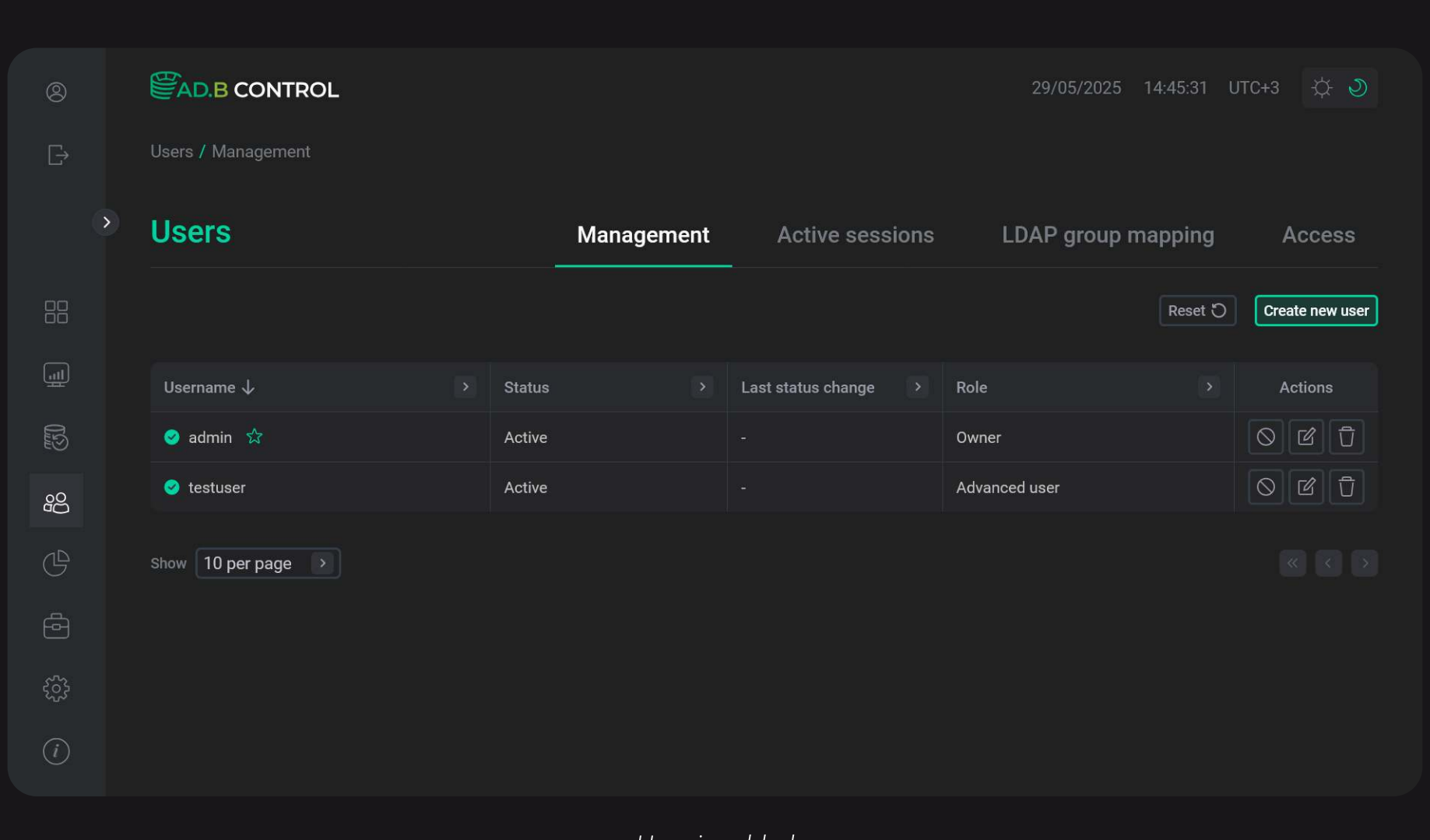
User
testuser

Role
○ ADBM Administrator
○ Administrator
● Advanced user
○ Owner
○ Regular user

Password
••••••••••••

Enter user data

3. Click ***Save***. As a result, a new user is displayed on the ***Users*** → ***Management*** tab.



AD.B CONTROL
29/05/2025 14:45:31 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

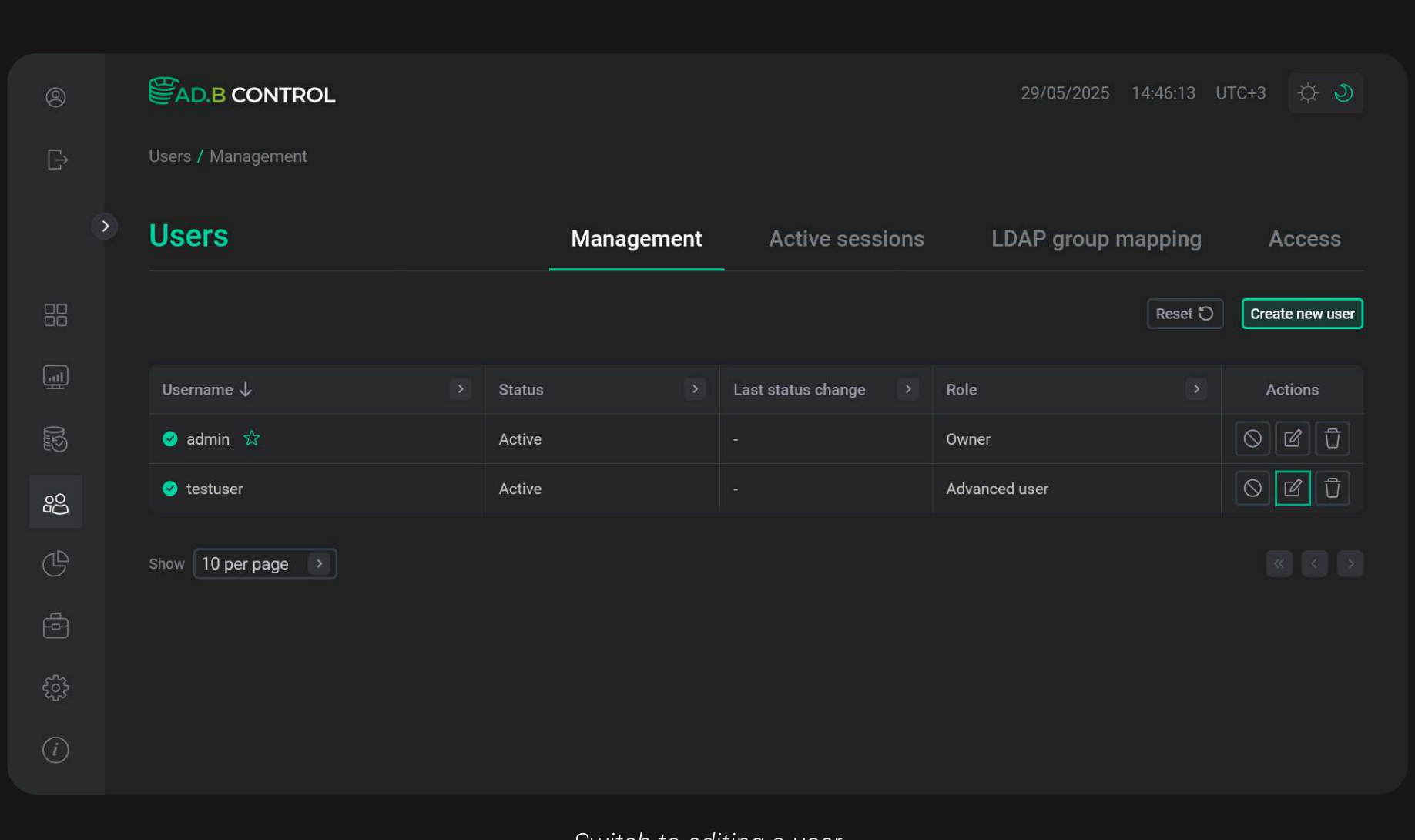
Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Active	-	Advanced user	

User is added

Edit a user

To edit user data, follow the steps:

1. Click the ✎ icon in the ***Actions*** column on the ***Users*** → ***Management*** tab.

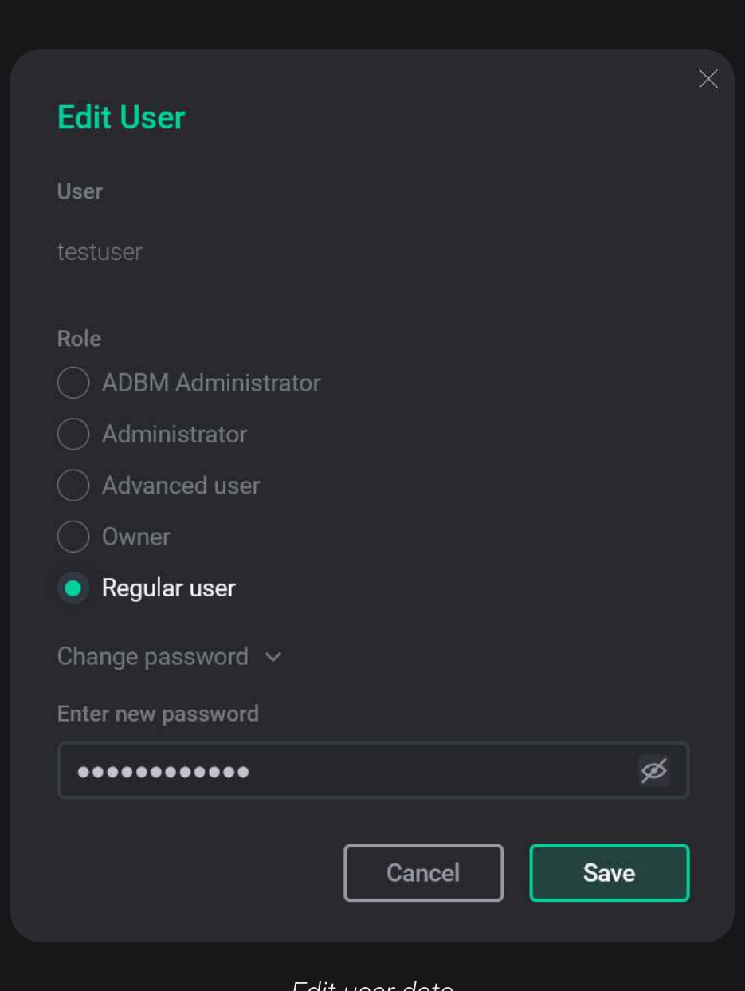


AD.B CONTROL
29/05/2025 14:46:13 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Active	-	Advanced user	

Switch to editing a user

2. In the window that opens, edit necessary fields. You can change a user role in the ***Role*** drop-down list and specify a new password in the ***Enter new password*** field (which is displayed after you open the ***Change password*** section). The ***User*** field value cannot be changed.



Edit User
Change password
Cancel
Save

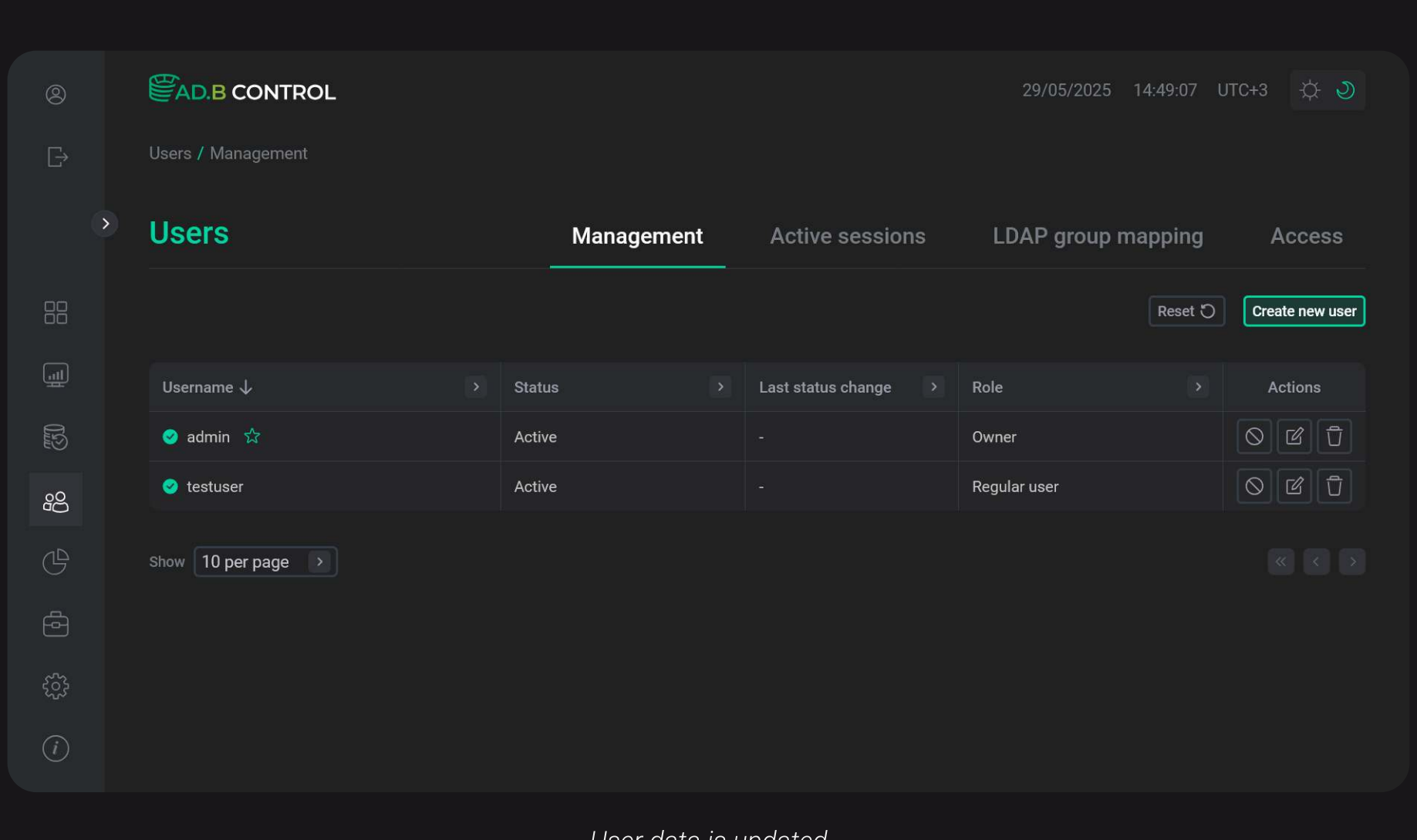
User
testuser

Role
○ ADBM Administrator
○ Administrator
○ Advanced user
○ Owner
● Regular user

Enter new password
••••••••••••

Edit user data

3. Click ***Save***. As a result, the user data is updated on the ***Users*** → ***Management*** tab.



AD.B CONTROL
29/05/2025 14:49:07 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

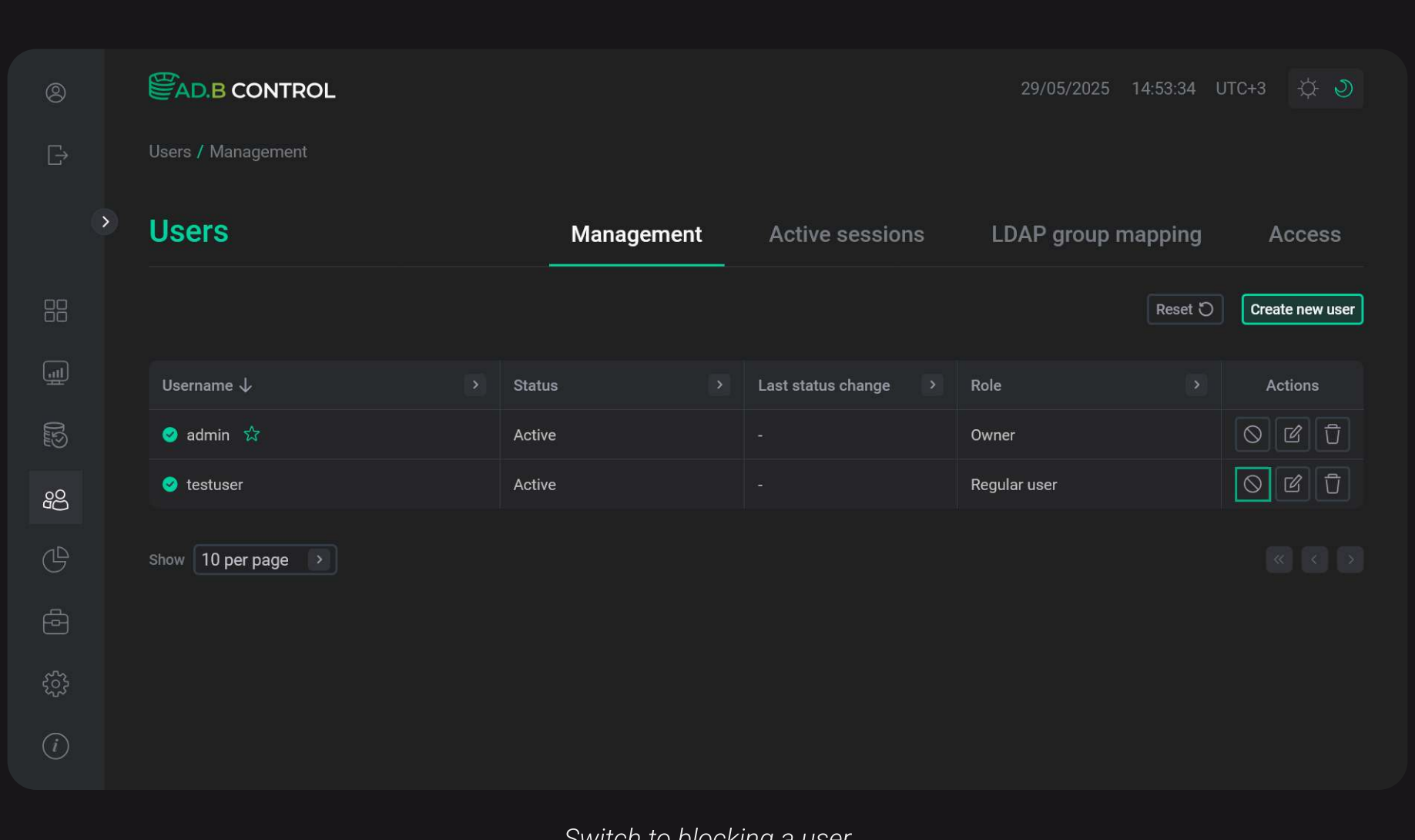
Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Active	-	Regular user	

User data is updated

Block a user

To block a user, follow the steps:

1. Click the ⦸ icon in the ***Actions*** column on the ***Users*** → ***Management*** tab.

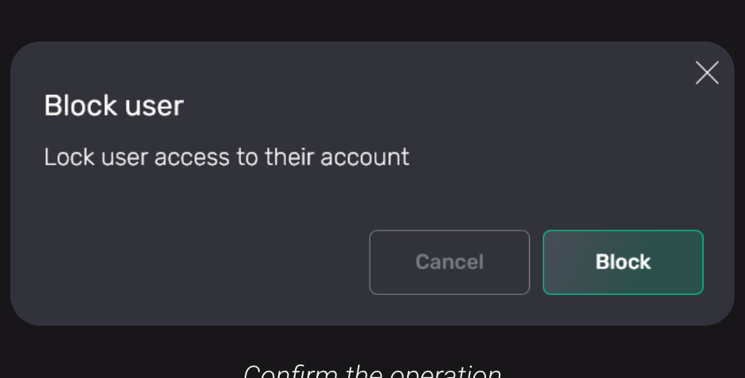


AD.B CONTROL
29/05/2025 14:53:34 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Active	-	Regular user	

Switch to blocking a user

2. In the window that opens, confirm the operation by clicking ***Block***.



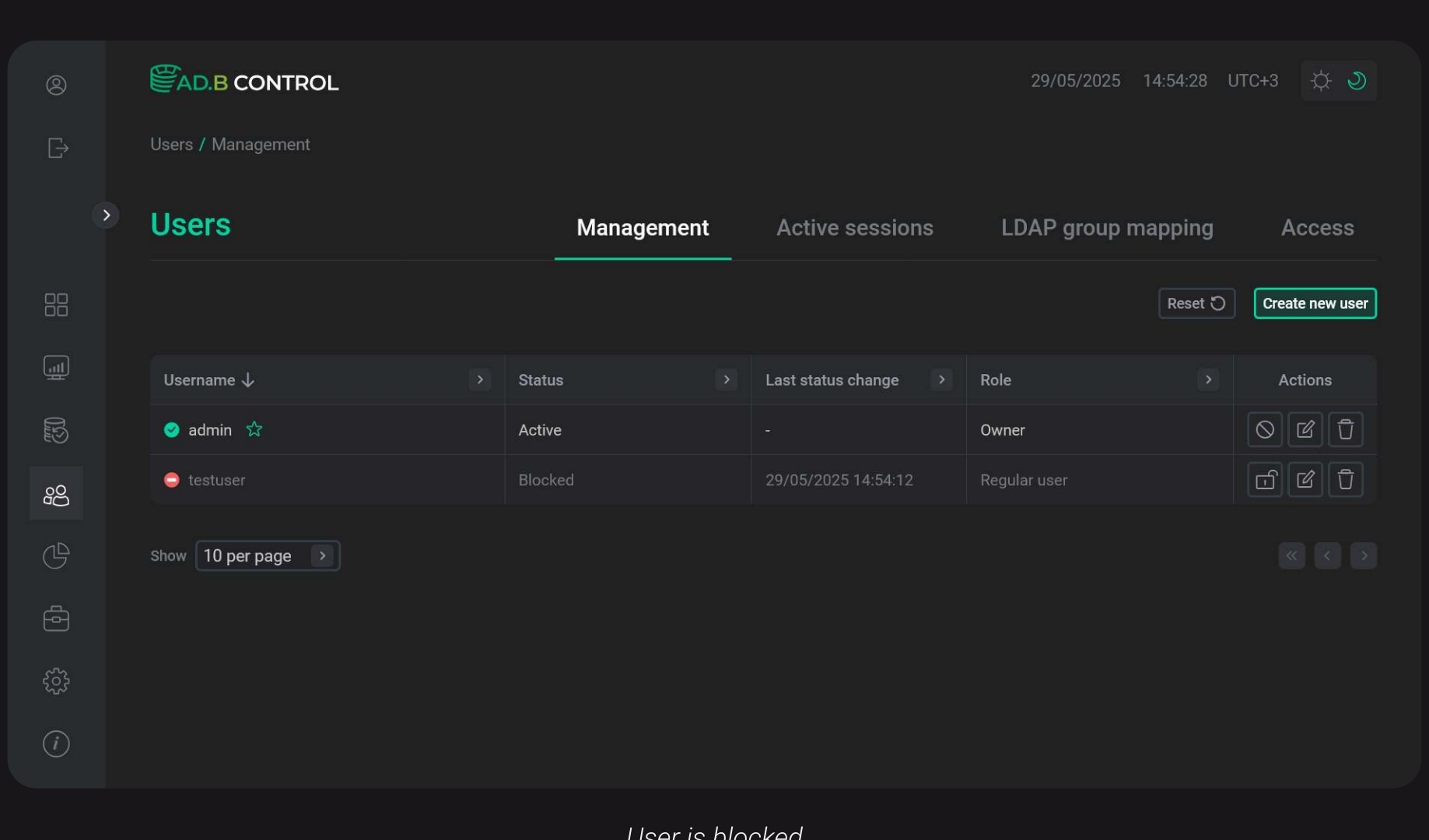
Cancel
Block

Block user

Lock user access to their account

Confirm the operation

As a result, the user status is changed from ***Active*** to ***Blocked***. Note that the ***Last status change*** field is also updated on the ***Users*** → ***Management*** tab.



AD.B CONTROL
29/05/2025 14:54:28 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

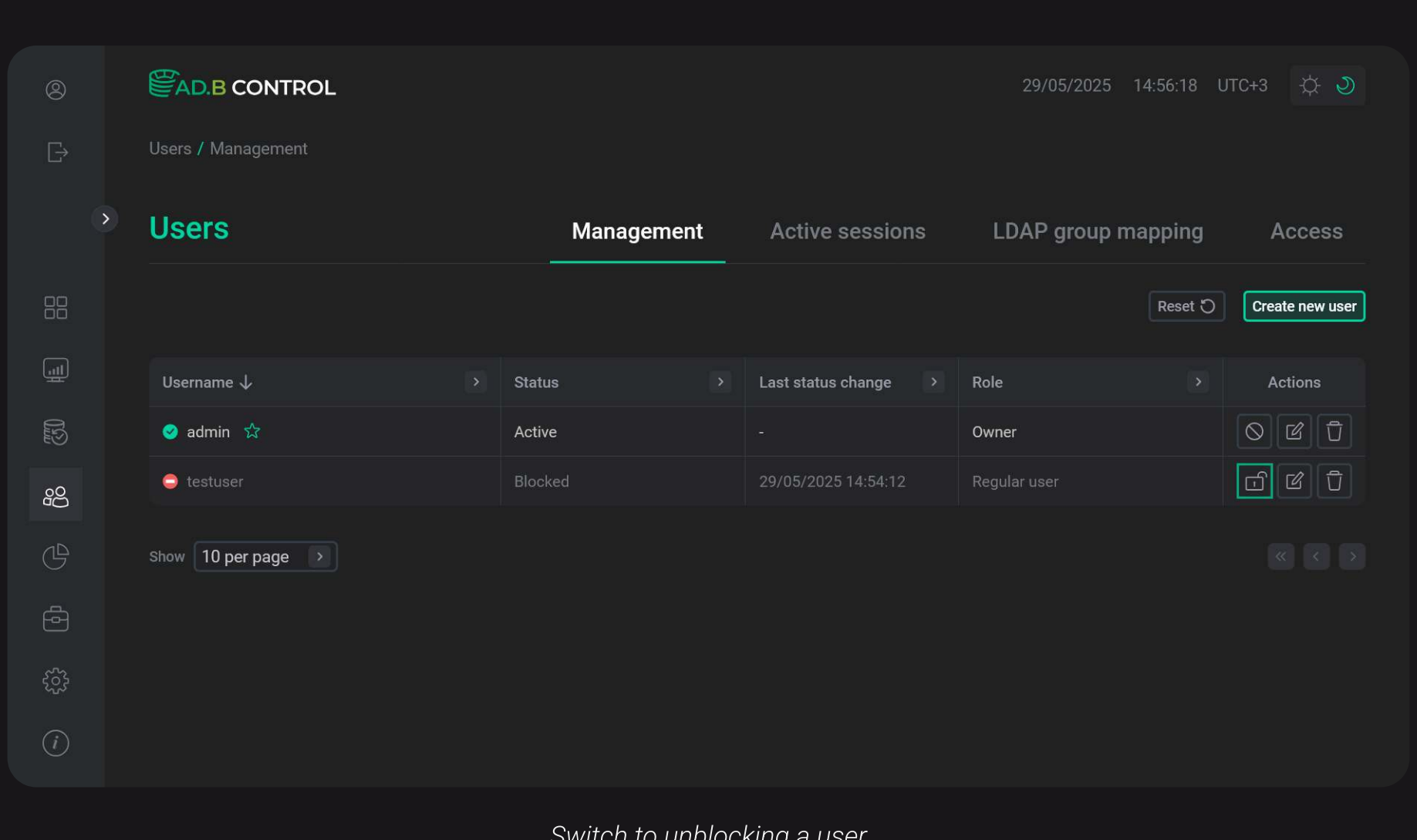
Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Blocked	29/05/2025 14:54:12	Regular user	

User is blocked

Unblock a user

To activate a previously blocked user, follow the steps:

1. Click the 🔓 icon in the ***Actions*** column on the ***Users*** → ***Management*** tab.

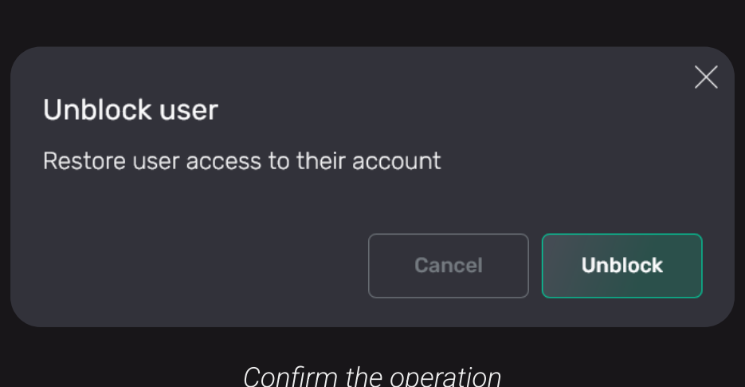


AD.B CONTROL
29/05/2025 14:56:18 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Blocked	29/05/2025 14:54:12	Regular user	

Switch to unblocking a user

2. In the window that opens, confirm the operation by clicking ***Unblock***.



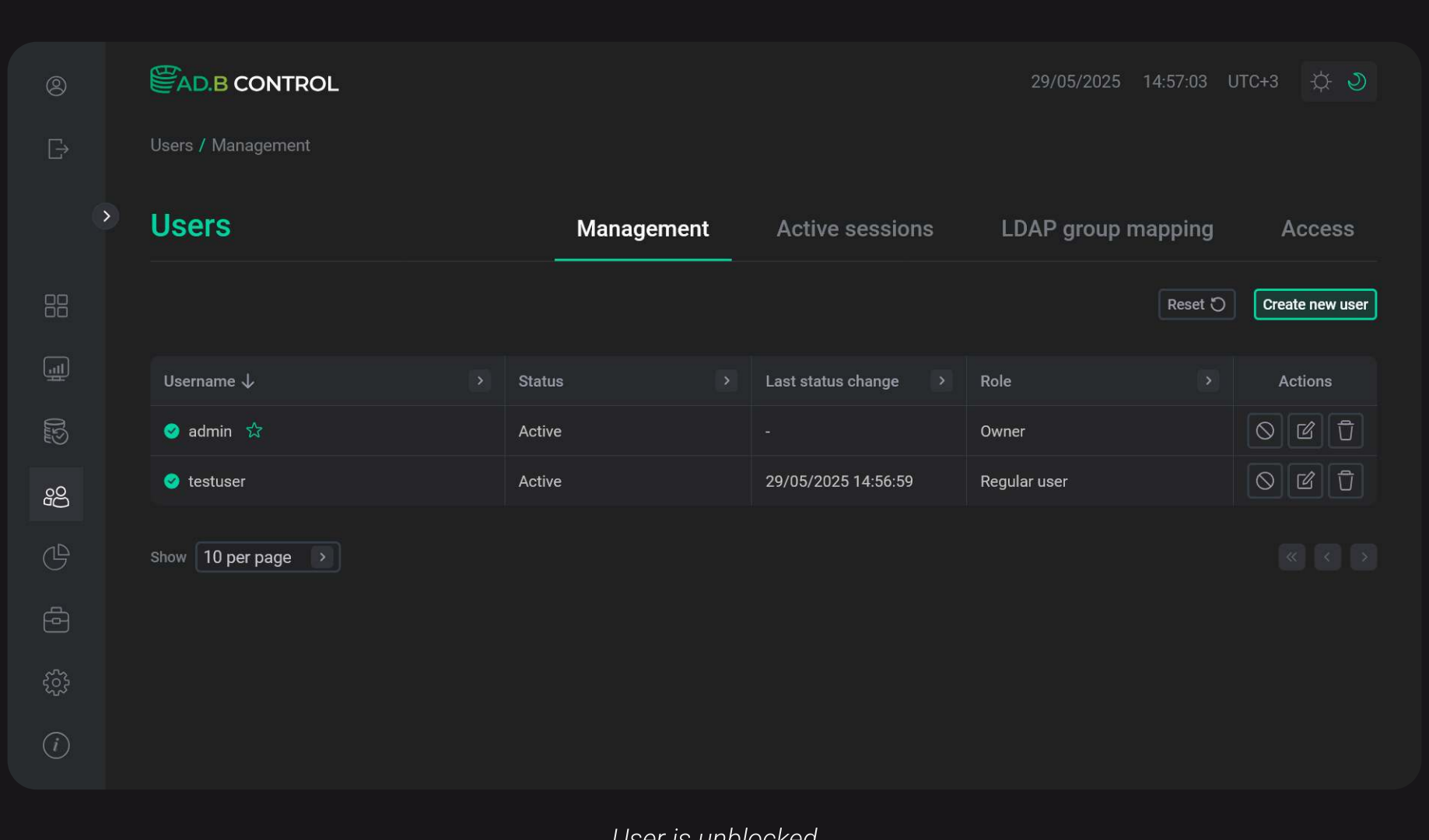
Cancel
Unblock

Unblock user

Restore user access to their account

Confirm the operation

As a result, the user regains access to the system, and the user status is changed from ***Blocked*** to ***Active***.



AD.B CONTROL
29/05/2025 14:57:03 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

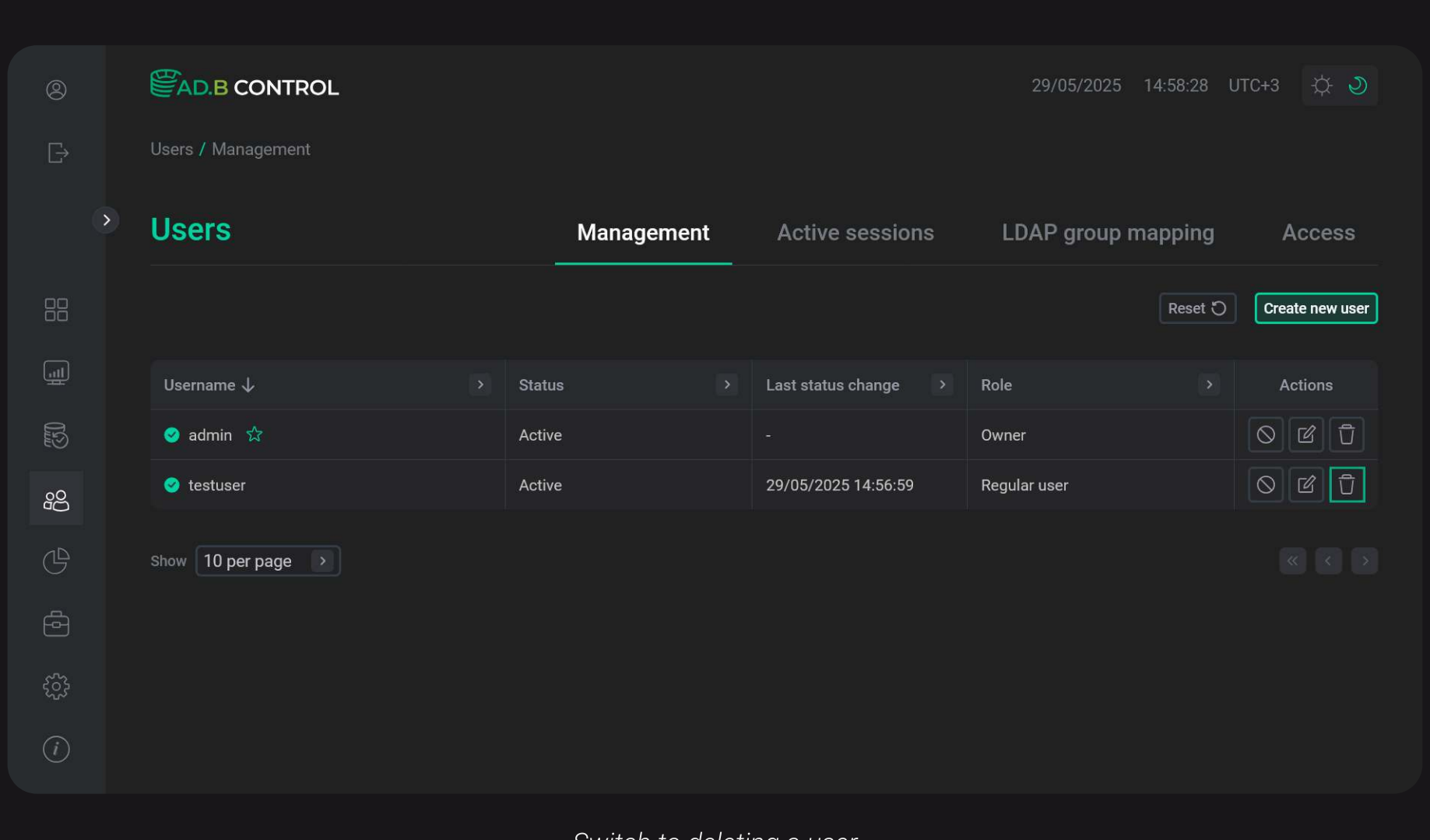
Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Active	29/05/2025 14:56:59	Regular user	

User is unblocked

Delete a user

To delete an existing user, follow the steps:

1. Click the 🗑 icon in the ***Actions*** column on the ***Users*** → ***Management*** tab.

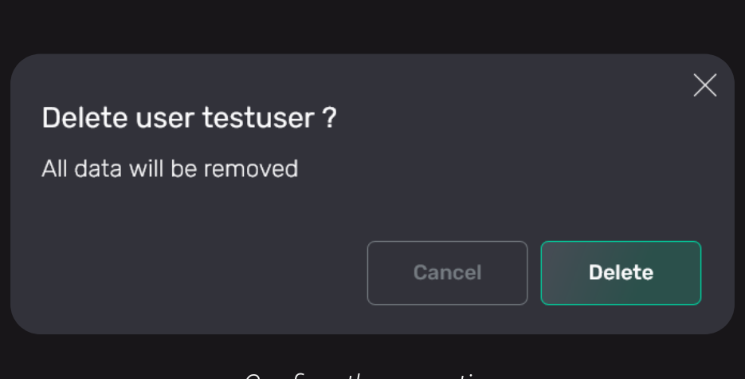


AD.B CONTROL
29/05/2025 14:58:28 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	
testuser	Active	29/05/2025 14:56:59	Regular user	

Switch to deleting a user

2. In the window that opens, confirm the operation by clicking ***Delete***.



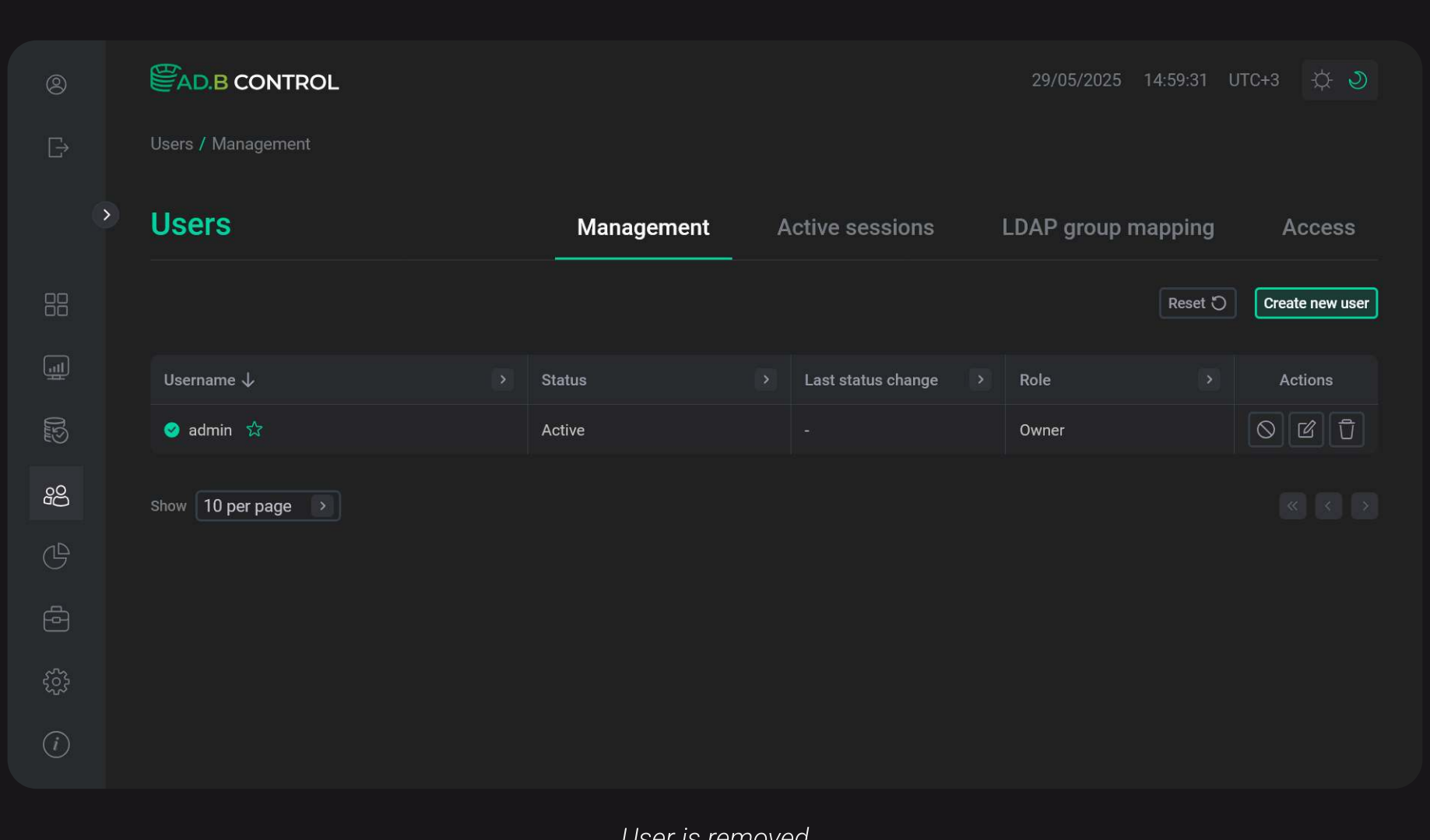
Cancel
Delete

Delete user testuser ?

All data will be removed

Confirm the operation

As a result, the user data is removed from the system and no longer displayed on the ***Users*** → ***Management*** tab.



AD.B CONTROL
29/05/2025 14:59:31 UTC+3
Users / Management
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Create new user
Show 10 per page

Username	Status	Last status change	Role	Actions
admin	Active	-	Owner	

User is removed

LDAP authentication

Daria Barysheva

To Table of Contents

ADB Control supports Lightweight Directory Access Protocol (LDAP) user authentication. Unlike basic authentication, you do not need to add users manually via ADB Control when using LDAP. User accounts and passwords are checked on the selected LDAP server.

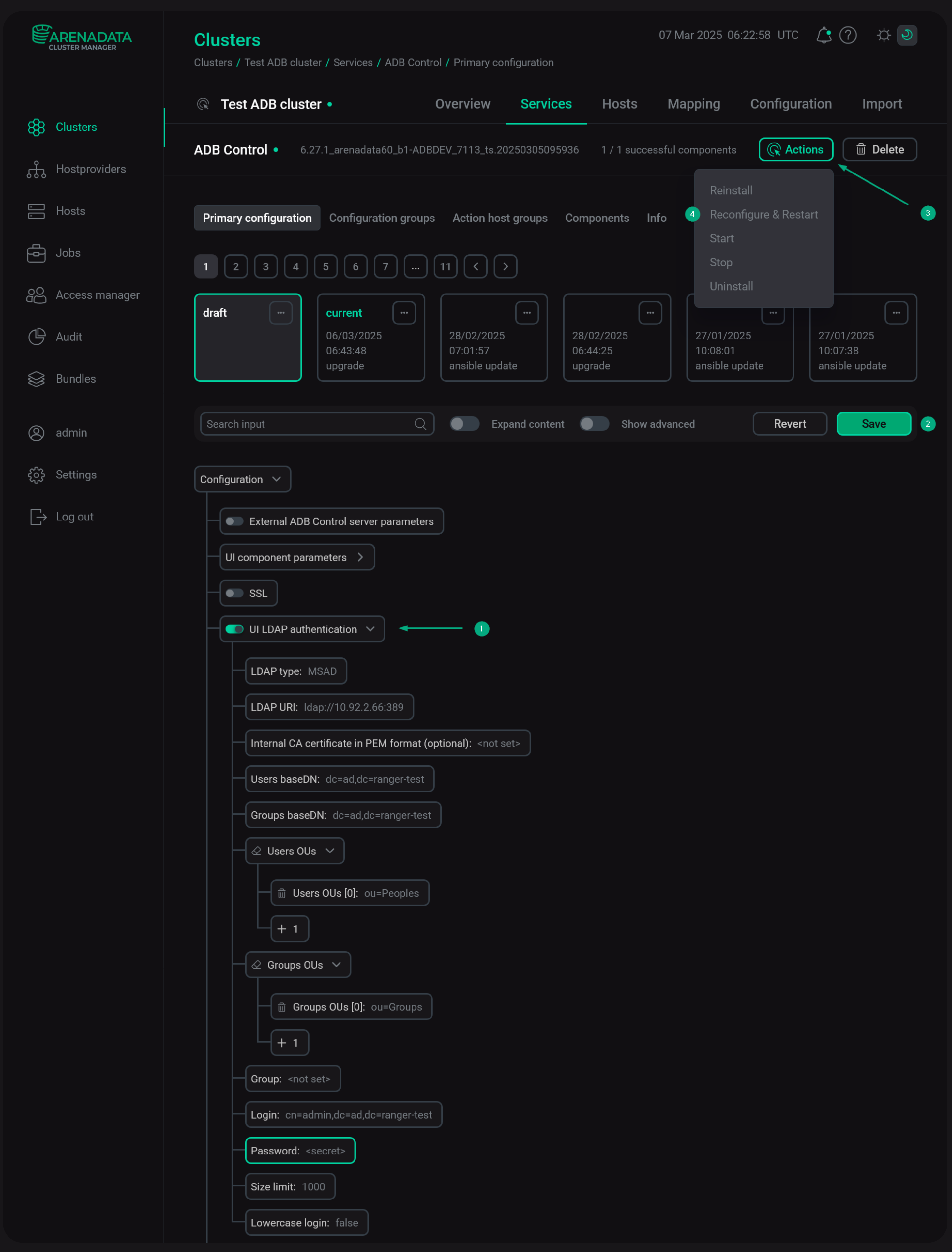
To configure LDAP authentication, follow the steps:

1. Ensure the target LDAP server is configured and available. ADB Control supports two LDAP implementations: Microsoft Active Directory (MS AD) and 389 Directory server as a part of FreeIPA.
2. Open the ADB Control service configuration page in ADCM.
3. Activate the ***UI LDAP authentication*** switcher.
4. Fill in the following fields.

Field	Description	Example in MS AD
LDAP type	A type of the LDAP server. Possible values: - `MSAD` — Microsoft Active Directory; - `FreeIPA` — 389 Directory server as a part of FreeIPA.	MSAD
LDAP URI	A URI for connecting to the LDAP server. Requires the following format: `<schema>://<host>:<port>`. For example, `ldap://example.com:389`, `ldaps://example.com:636`. If a port number is not specified, the standard ports are used: `389` for LDAP and `636` for LDAPS	ldap://10.92.2.66:389
Internal CA certificate in PEM format (optional)	The internal CA certificate or certificate chain in PEM format that signed the LDAP server certificate	—
Users baseDN	A limitation of the search scope of objects in the LDAP directory that is used in users search requests	dc=ad,dc=ranger-test
Groups baseDN	A limitation of the search scope of objects in the LDAP directory that is used in groups search requests	dc=ad,dc=ranger-test
Users OUs	A list of `OU` for user search. If the option is set, users will be searched only in the specified OUs, otherwise — within full ***Users baseDN***. Enter each `OU` value on a separate line using the `ou=<name>` format (e.g. `ou=users1`)	ou=Peoples
Groups OUs	A list of `OU` for group search. If the option is set, groups will be searched only in the specified OUs, otherwise — within full ***Groups baseDN***. Enter each `OU` value on a separate line using the `ou=<name>` format (e.g. `ou=groups1`)	ou=Groups
Group	A group name that is used for service requests to the LDAP server	—
Login	A user name that is used for service requests to the LDAP server	cn=admin,dc=ad,dc=ranger-test
Password	A user password that is used for service requests to the LDAP server	A password of the `admin` user
Size limit	A maximum number of records that can be returned by the LDAP server	1000
Lowercase login	Defines whether to convert user names to lowercase	false

> **IMPORTANT**
>
> - Different LDAP implementations use different names for object types and identifiers. It is recommended to refine the settings format for a specific LDAP implementation from the LDAP server administrator.
> - To use LDAP over SSL (LDAPS) starting with ADB Control 4.10.3, do the following:
>   - Use the `ldaps` protocol in the ***LDAP URI*** field.
>   - Fill in the ***HTTPS parameters enable*** section.
>   - Fill in the ***Internal CA certificate in PEM format (optional)*** field. The certificate should be replicated to all hosts used by ADB Control (imported to *truststore.jks*).

5. Click ***Save***. Then apply the ***Reconfigure & Restart*** action to the ADB Control service.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / Test ADB cluster / Services / ADB Control / Primary configuration
07 Mar 2025 06:22:58 UTC
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADB Control
6.27.1_arenadata60_b1-ADBDEV_7113_ts.20250305095936
1 / 1 successful components
Actions
Delete
Reinstall
Reconfigure & Restart
Start
Stop
Uninstall
Primary configuration
Configuration groups
Action host groups
Components
Info
1 2 3 4 5 6 7 ... 11
draft
current 06/03/2025 06:43:48 upgrade
28/02/2025 07:01:57 ansible update
28/02/2025 06:44:25 upgrade
27/01/2025 10:08:01 ansible update
27/01/2025 10:07:38 ansible update
Search input
Expand content
Show advanced
Revert
Save
Configuration
External ADB Control server parameters
UI component parameters
SSL
UI LDAP authentication
+ 1
+ 1
1
2
3
4

LDAP type: MSAD
LDAP URI: ldap://10.92.2.66:389
Internal CA certificate in PEM format (optional): <not set>
Users baseDN: dc=ad,dc=ranger-test
Groups baseDN: dc=ad,dc=ranger-test
Users OUs
Users OUs [0]: ou=Peoples
Groups OUs
Groups OUs [0]: ou=Groups
Group: <not set>
Login: cn=admin,dc=ad,dc=ranger-test
Password: <secret>
Size limit: 1000
Lowercase login: false

Configure LDAP authentication for ADB Control via ADCM

6. In the ADB Control web interface, map LDAP user groups to ADB Control roles. This is necessary for LDAP users to get the appropriate permissions to work in ADB Control. For more information, see Authorization.

If all steps are completed successfully, users can connect to ADB Control using the accounts registered for them on the LDAP server.

Authorization

Daria Barysheva

Contents

To Table of Contents

- Roles and access permissions
- Configure role mappings with LDAP groups
  - Add a mapping
  - Edit a mapping
  - Delete a mapping
  - Actualize LDAP group names
- Set advanced permissions
  - Add a permission
  - Edit a permission
  - Delete a permission

Roles and access permissions

Authorization in ADB Control is based on Role Based Access Control (RBAC). This mechanism allows you to control access to the various functions of the system depending on the role assigned to each user.

Currently, ADB Control supports the following roles:

- `Owner` — system owner.
- `Administrator` — ADB Control administrator.
- `ADBM Administrator` — ADBM administrator.
- `Advanced User` — user with advanced permissions.
- `Regular User` — ordinary user.

> **NOTE**
>
> For users with the `Regular User` role, logins are associated with role names (`ROLE`) in ADB on the basis of a full match, which gives users the permissions to work with certain queries and transactions on the ***Monitoring*** page.

The table below shows the distribution of access rights between the abovementioned roles.

> **IMPORTANT**
>
> The resource group management feature, which was available in ADB Control 4.14, is absent in the current ADB Control version, 4.15. Support for this functionality will be restored in future releases of ADB Control.

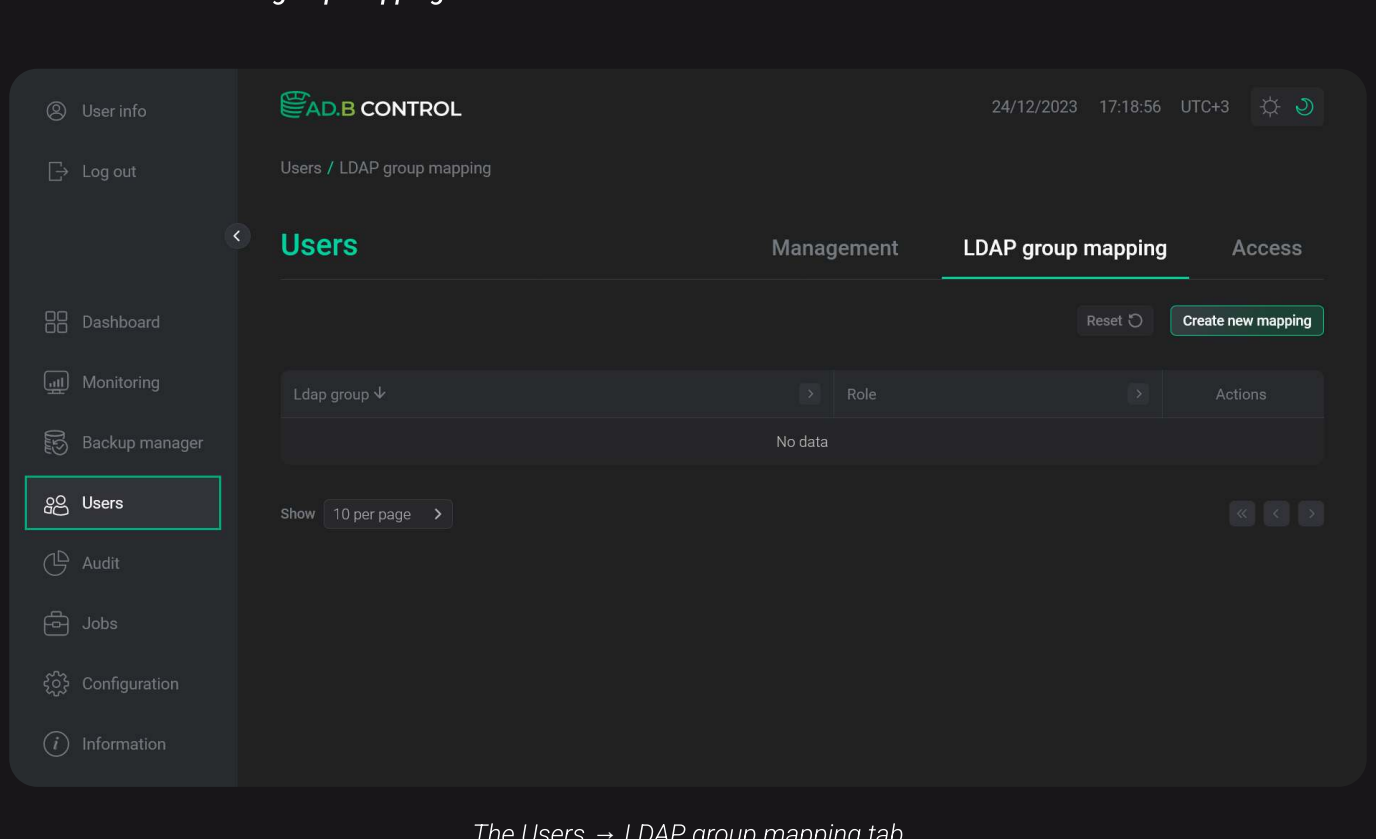
Access permissions in ADB Control

Permission	Description	Regular User	Advanced User	Administrator	Owner	ADBM Administrator
View self queries	Ability to view own transactions and commands (including detailed information on them). These are transactions and queries that are committed in ADB under the role which name matches the user name in ADB Control or belongs to the LDAP group for which a mapping is configured in ADB Control	Yes	Yes	Yes	Yes	No
Kill self queries	Ability to terminate and cancel own transactions and commands. These are transactions and queries that are committed in ADB under the role which name matches the user name in ADB Control or belongs to the LDAP group for which a mapping is configured in ADB Control	Yes	Yes	Yes	Yes	No
View some queries	Ability to view the transactions and commands to which a user has relevant permissions (including detailed information)	No	Yes	Yes	Yes	No
Kill some queries	Ability to terminate and cancel the transactions to which a user has relevant permissions	No	Yes	Yes	Yes	No
View all queries	Ability to view all transactions and commands (including detailed information)	No	No	Yes	Yes	No
Kill all queries	Ability to terminate and cancel all transactions	No	No	Yes	Yes	No
View relation audit page	Ability to view the Audit → Relations page where statistics on the number of access attempts to ADB relations are displayed	No	No	Yes	Yes	No
View secured relation audit	Ability to view a table with secured statistics on ADB relations in a separate modal window	No	No	Yes	Yes	No
View Operations audit page	Access to the Audit → Operations page where statistics on launches of various operations in ADB Control is displayed	No	No	Yes	Yes	No
View Authorization audit page	Access to the Audit → Authorizations page where information on user authorizations in ADB Control/ADB is displayed	No	No	Yes	Yes	No
View ADB jobs	Access to the Jobs → ADB Control page where information on scheduled (***Schedule***) and completed (***Audit***) jobs in ADB Control is displayed	No	No	Yes	Yes	No
View backup manager jobs	Access to the Jobs → Backup manager page where information on scheduled (***Schedule***) and completed (***Audit***) jobs in ADB Backup Manager is displayed	No	No	Yes	Yes	Yes
View backup manager page	Access to the ***Backup manager*** page with the ability to view information on clusters, actions, backups, and restores	No	No	Yes	Yes	Yes
Run backup manager actions	Ability to run actions in ADB Backup Manager	No	No	Yes	Yes	Yes
Create backup manager configuration	Ability to create and edit configurations in ADB Backup Manager	No	No	Yes	Yes	Yes
View ADB information page	Access to the Information → ADB Control page where the current status of ADB Control components is displayed	No	No	Yes	Yes	No
View backup manager information page	Access to the Information → Backup manager page where the current status of ADB Backup Manager components is displayed	No	No	Yes	Yes	Yes
View configuration page	Access to the ***Configuration*** page with the ability to manage clusters and configure policies for jobs and security	No	No	No	Yes	No
View users page	Access to the ***Users*** page with the ability to manage users (create, edit, block, delete), set mappings with LDAP groups, and set advanced permissions	No	No	No	Yes	No
View sessions	Access to the Monitoring → Sessions page functionality	No	No	Yes	Yes	No
Manage user sessions	Ability to terminate sessions of ADB Control users on the ***Users → Active sessions*** page	No	No	Yes	Yes	No

Configure role mappings with LDAP groups

If basic authentication is used, roles are assigned to users at the creation step. However, if LDAP authentication is configured, you need to map user groups of the selected LDAP server to roles in ADB Control. It allows users to get appropriate permissions in the monitoring system.

You can manage such mappings on the ***Users → LDAP group mapping*** tab in the ADB Control web interface.

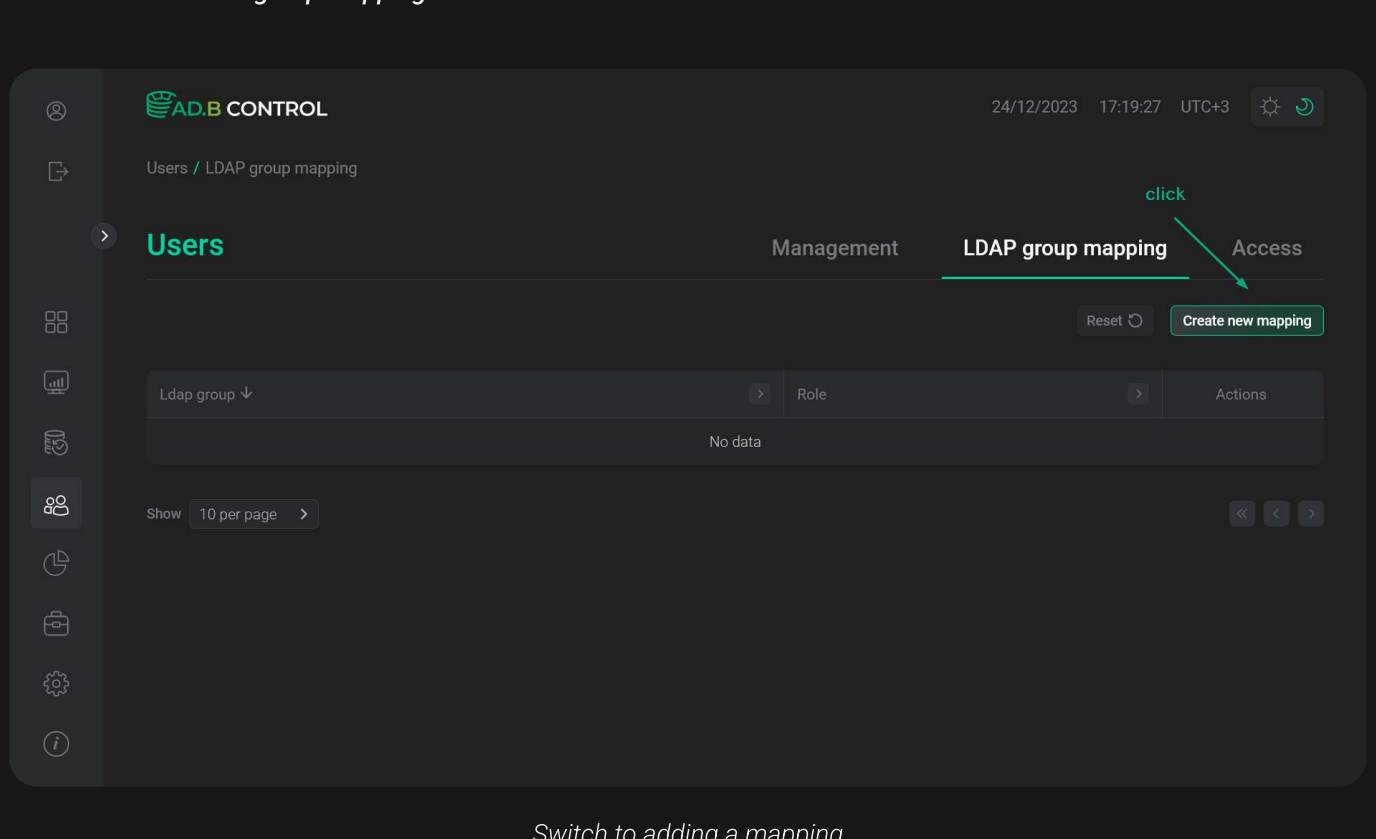

User info
Log out
AD.B CONTROL
24/12/2023 17:18:56 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
Reset
Create new mapping
Ldap group
Role
Actions
No data
Show 10 per page
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information

The Users → LDAP group mapping tab

Add a mapping

To add a new mapping, follow the steps:

1. Click ***Create new mapping*** on the ***Users → LDAP group mapping*** tab.

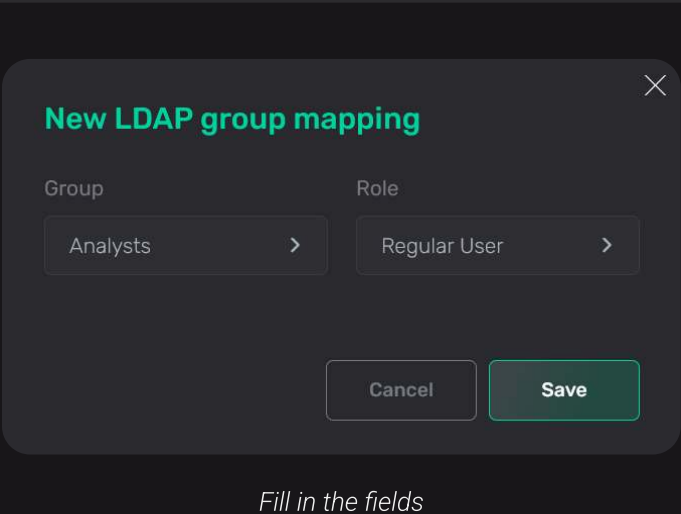

AD.B CONTROL
24/12/2023 17:19:27 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
click
Reset
Create new mapping
Ldap group
Role
Actions
No data
Show 10 per page

Switch to adding a mapping

2. In the window that opens, fill in the following fields:

   - ***Group*** — a user group name on the LDAP server. A drop-down list with group names becomes available after the LDAP authentication is successfully configured.
   - ***Role*** — a role in ADB Control.

> **IMPORTANT**
>
> Each LDAP group can be mapped only to one role in ADB Control.



New LDAP group mapping

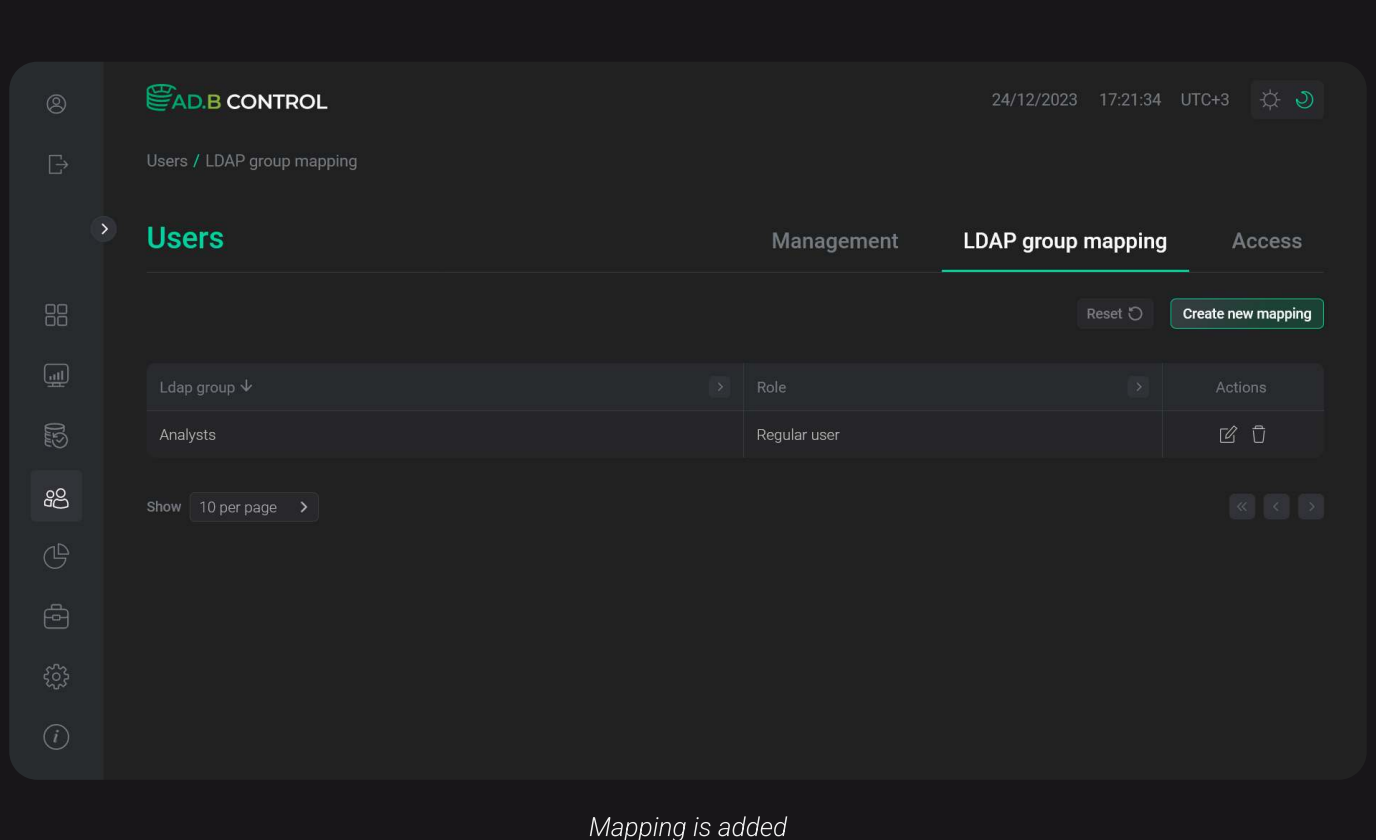
Group: Analysts

Role: Regular User

Cancel
Save

Fill in the fields

3. Click ***Save***. As a result, a new mapping is displayed on the ***Users → LDAP group mapping*** tab.


AD.B CONTROL
24/12/2023 17:21:34 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
Reset
Create new mapping
Ldap group
Role
Actions
Show 10 per page

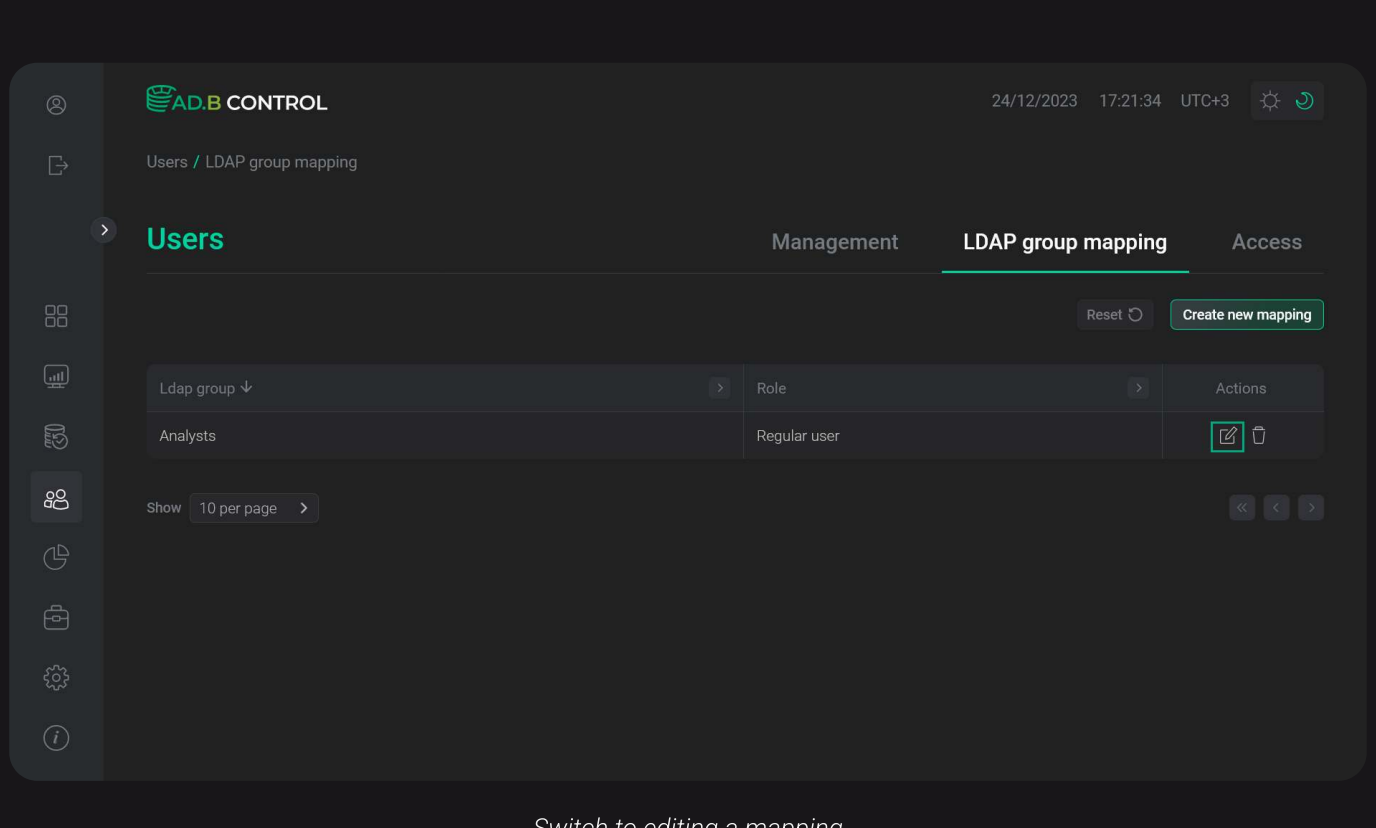
Analysts | Regular user

Mapping is added

Edit a mapping

To edit an existing mapping, follow the steps:

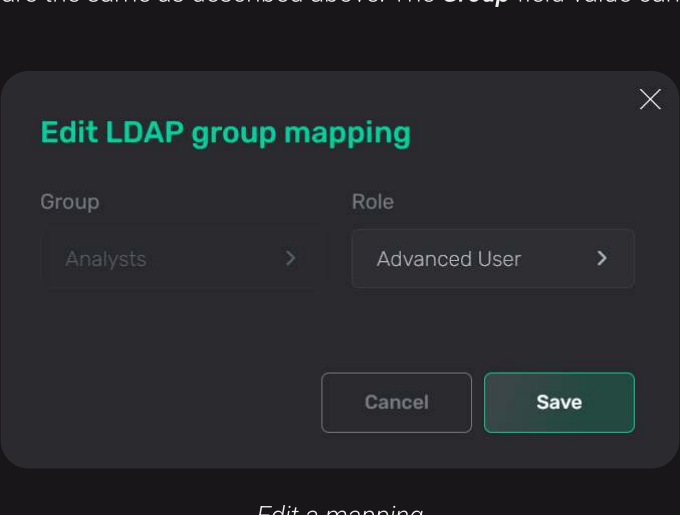
1. Click the icon in the ***Actions*** column on the ***Users → LDAP group mapping*** tab.


AD.B CONTROL
24/12/2023 17:21:34 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
Reset
Create new mapping
Ldap group
Role
Actions
Show 10 per page

Analysts | Regular user

Switch to editing a mapping

2. In the window that opens, edit necessary fields. All fields are the same as described above. The ***Group*** field value cannot be changed.



Edit LDAP group mapping

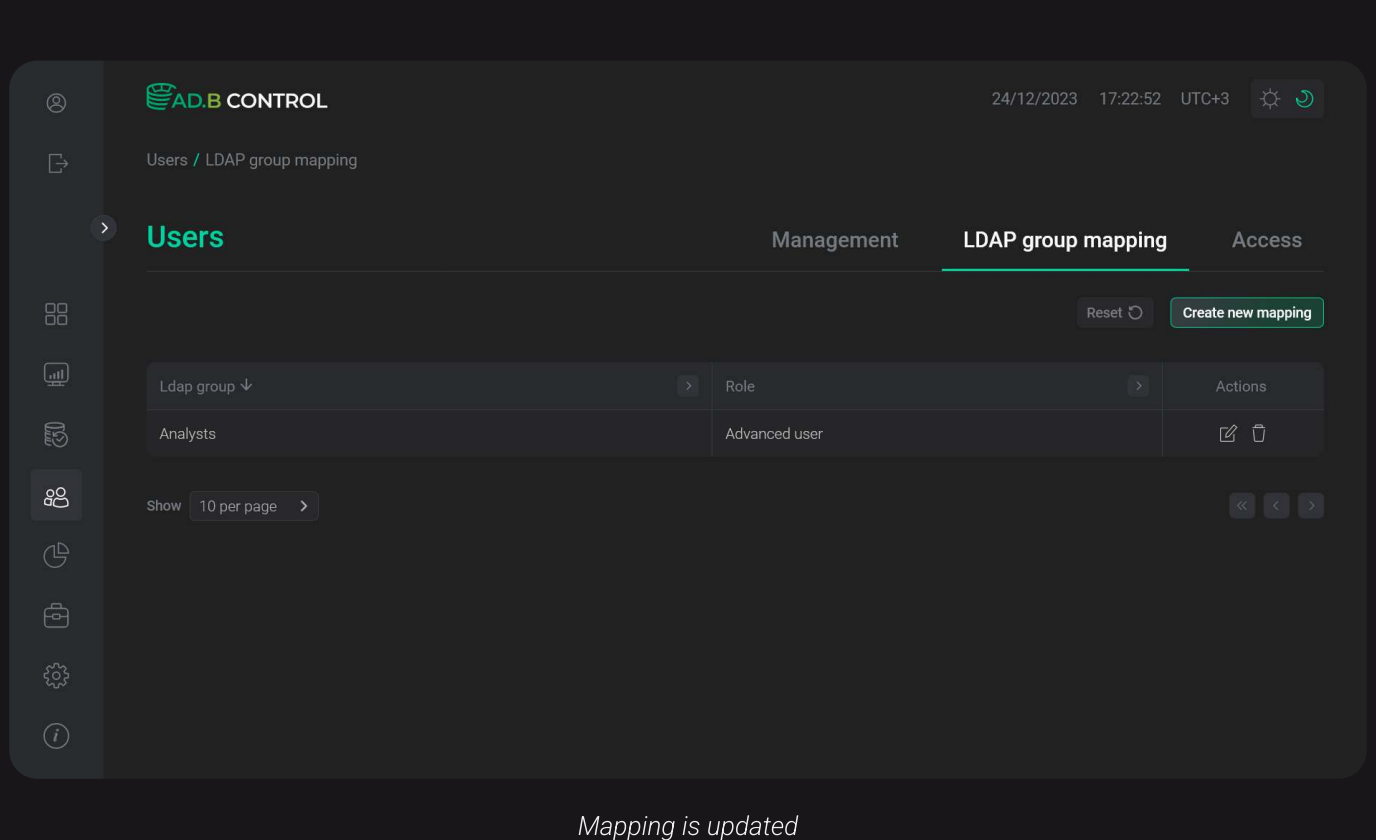
Group: Analysts

Role: Advanced User

Cancel
Save

Edit a mapping

3. Click ***Save***. As a result, the mapping data is updated on the ***Users → LDAP group mapping*** tab.


AD.B CONTROL
24/12/2023 17:22:52 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
Reset
Create new mapping
Ldap group
Role
Actions
Show 10 per page

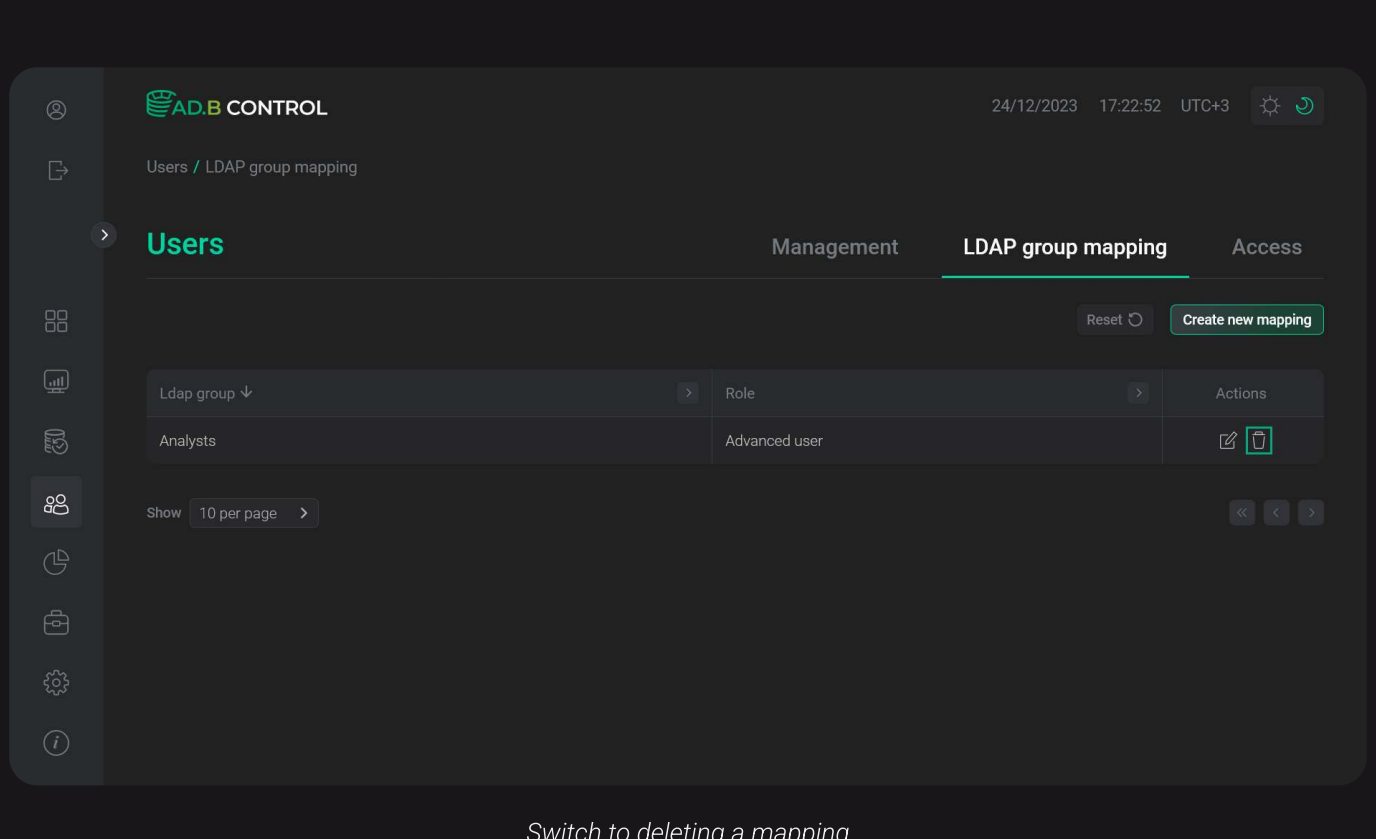
Analysts | Advanced user

Mapping is updated

Delete a mapping

To delete an existing mapping, follow the steps:

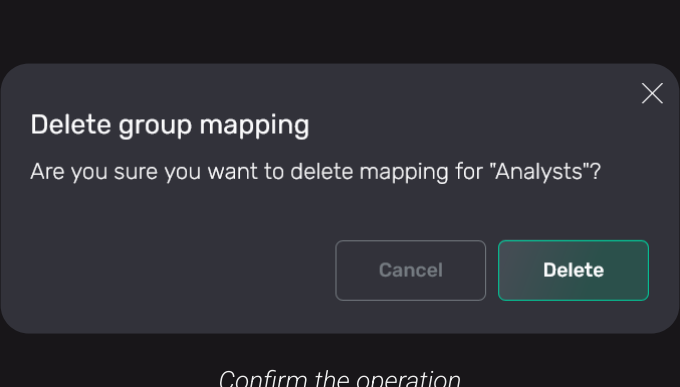
1. Click the icon in the ***Actions*** column on the ***Users → LDAP group mapping*** tab.


AD.B CONTROL
24/12/2023 17:22:52 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
Reset
Create new mapping
Ldap group
Role
Actions
Show 10 per page

Analysts | Advanced user

Switch to deleting a mapping

2. In the window that opens, confirm the operation by clicking ***Delete***.



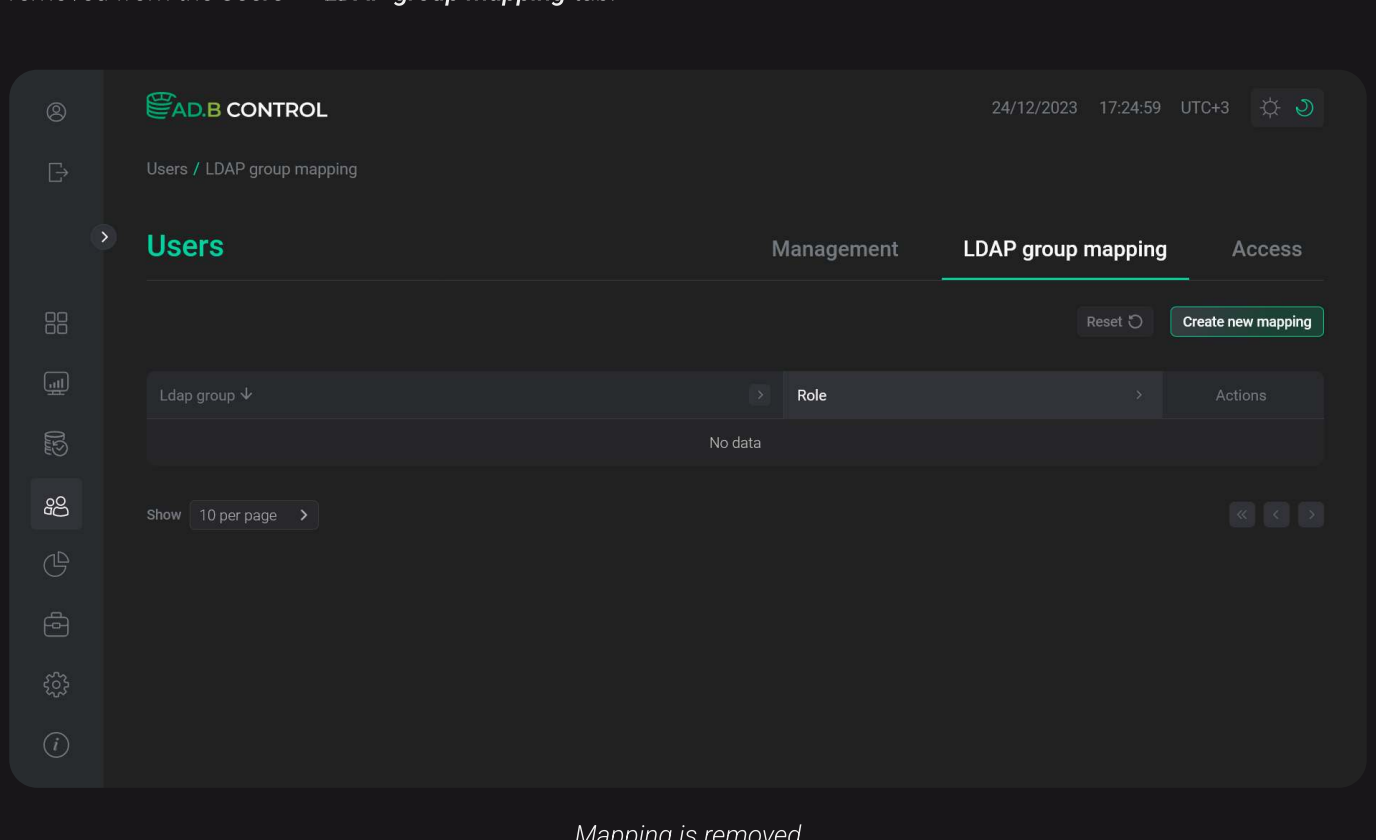
Delete group mapping

Are you sure you want to delete mapping for "Analysts"?

Cancel
Delete

Confirm the operation

As a result, the mapping is removed from the ***Users → LDAP group mapping*** tab.


AD.B CONTROL
24/12/2023 17:24:59 UTC+3
Users / LDAP group mapping
Users
Management
LDAP group mapping
Access
Reset
Create new mapping
Ldap group
Role
Actions
No data
Show 10 per page

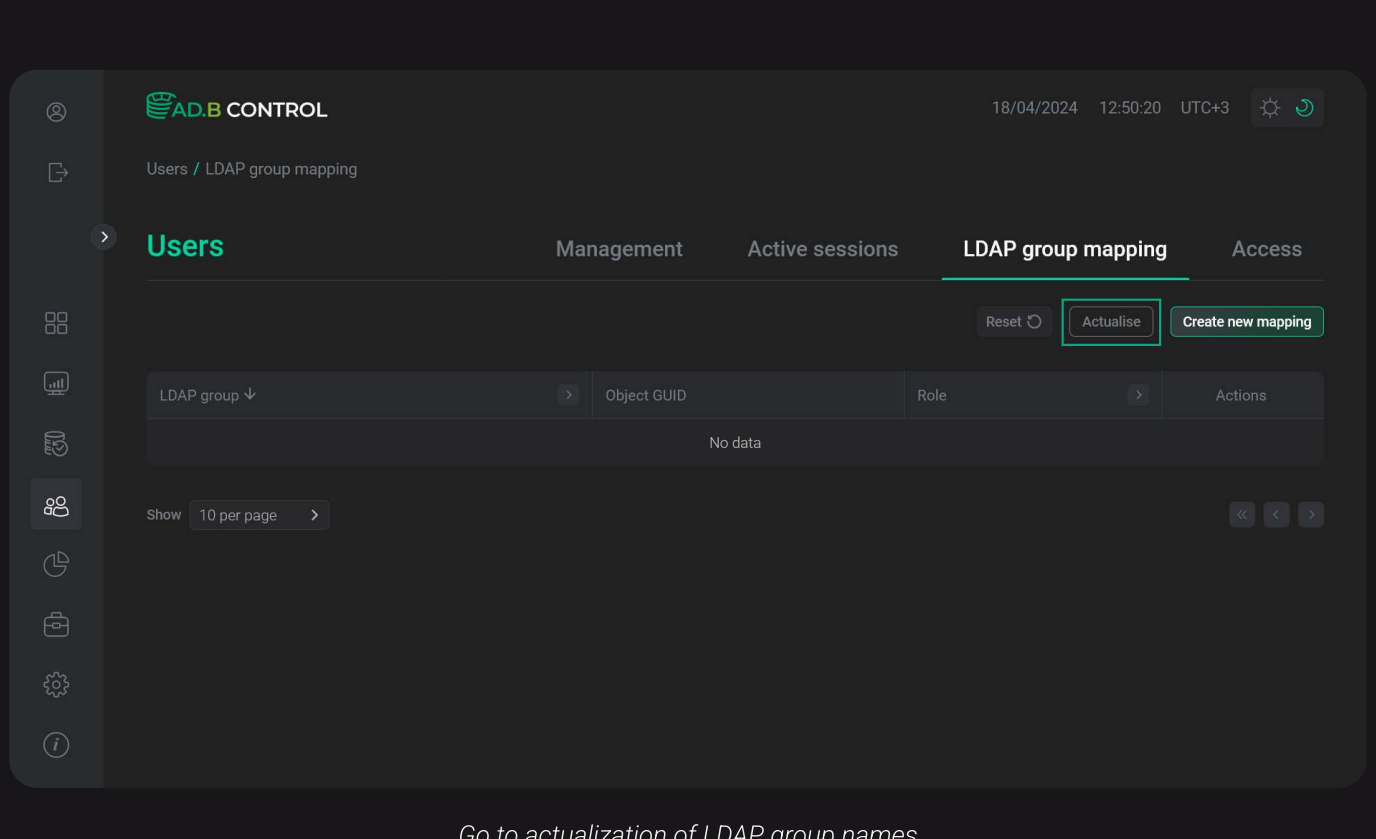
Mapping is removed

Actualize LDAP group names

> **NOTE**
>
> If LDAP authentication is configured, LDAP groups are synchronized automatically each time when ADB Control is being started. The following feature is designed for on-demand updates of LDAP group names.

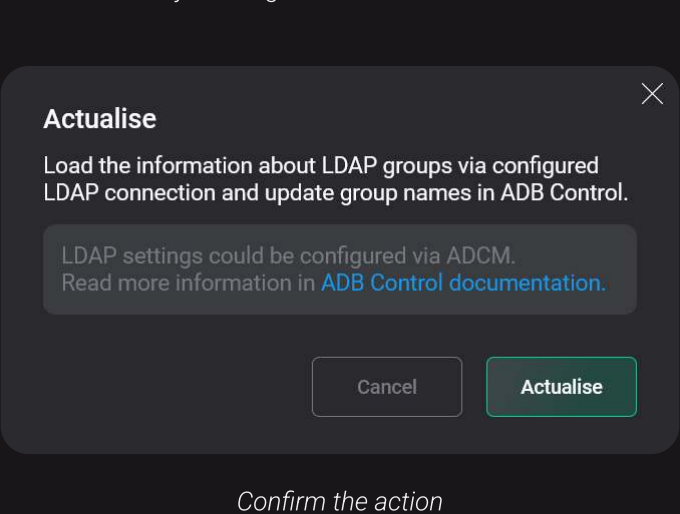
Starting with ADB Control 4.7.5, you can actualize LDAP group names in the previously created mappings by running synchronization with an LDAP server on the ***Users → LDAP group mapping*** page. To run synchronization:

1. Ensure that LDAP authentication for ADB Control is successfully configured and ADB Control roles are mapped to LDAP groups on the ***Users → LDAP group mapping*** page.
2. Click ***Actualise***.


AD.B CONTROL
18/04/2024 12:50:20 UTC+3
Users / LDAP group mapping
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Actualise
Create new mapping
LDAP group
Object GUID
Role
Actions
No data
Show 10 per page

Go to actualization of LDAP group names

3. Read the information in the window that opens and confirm the action by clicking ***Actualise***.



Actualise

Load the information about LDAP groups via configured LDAP connection and update group names in ADB Control.

LDAP settings could be configured via ADCM.
Read more information in ADB Control documentation.

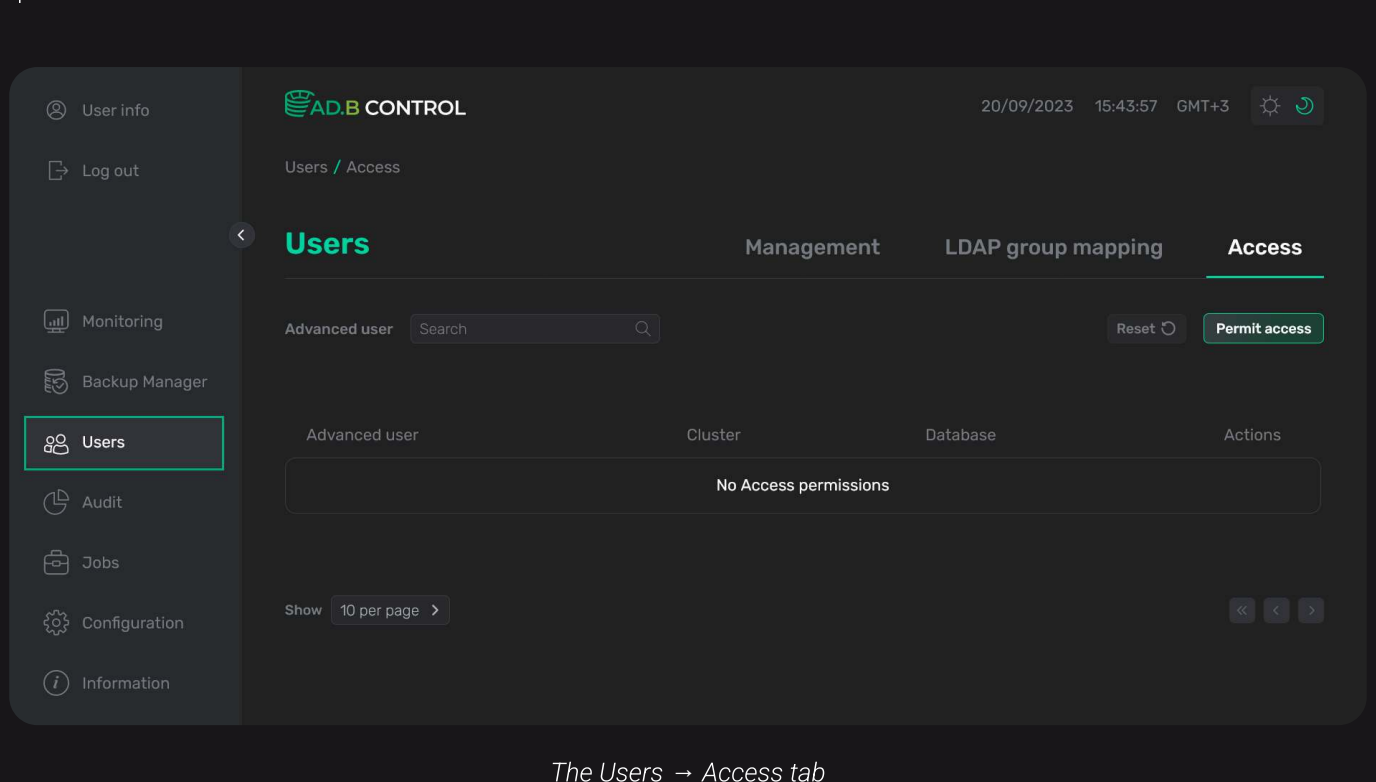
Cancel
Actualise

Confirm the action

Set advanced permissions

Users with the `Advanced User` role can be granted extended access rights to specific ADB databases. With such rights, users are able to view all transactions and commands running in the respective databases (regardless of who launched them) and interrupt or cancel them if necessary (see `View some queries` and `Kill some queries` permissions in the table above).

You can manage such advanced permissions on the ***Users → Access*** tab in the ADB Control web interface.

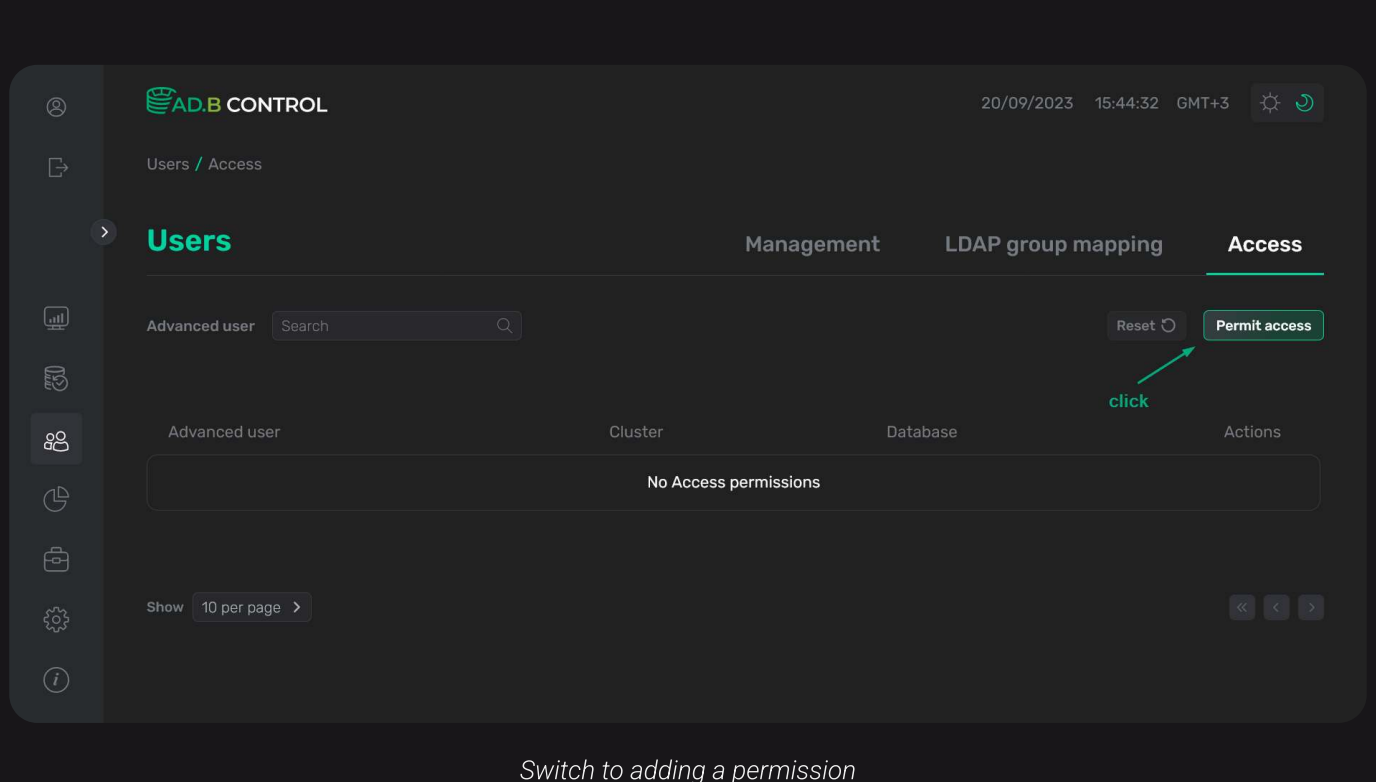

User info
Log out
AD.B CONTROL
20/09/2023 15:43:57 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
Advanced user
Cluster
Database
Actions
No Access permissions
Show 10 per page
Monitoring
Backup Manager
Users
Audit
Jobs
Configuration
Information

The Users → Access tab

Add a permission

To add a new permission, follow the steps:

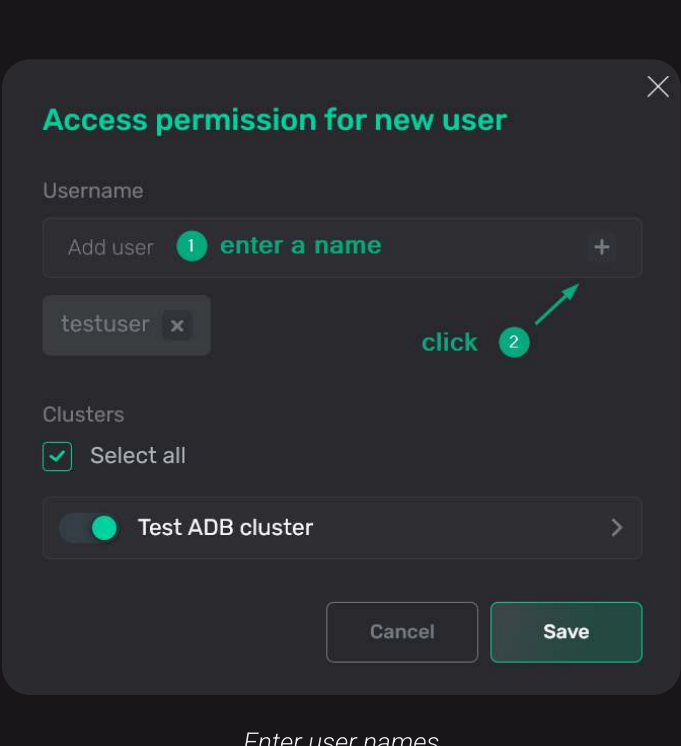
1. Click ***Permit access*** on the ***Users → Access*** tab.


AD.B CONTROL
20/09/2023 15:44:32 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
click
Advanced user
Cluster
Database
Actions
No Access permissions
Show 10 per page

Switch to adding a permission

2. In the window that opens, fill in the following information:

   - In the ***Username*** field, specify one or several users. To add each user, add a user name and then click ***+***. As a result, a list of users is displayed under the ***Username*** field. You can delete wrong records if necessary.



Access permission for new user

Username: testuser

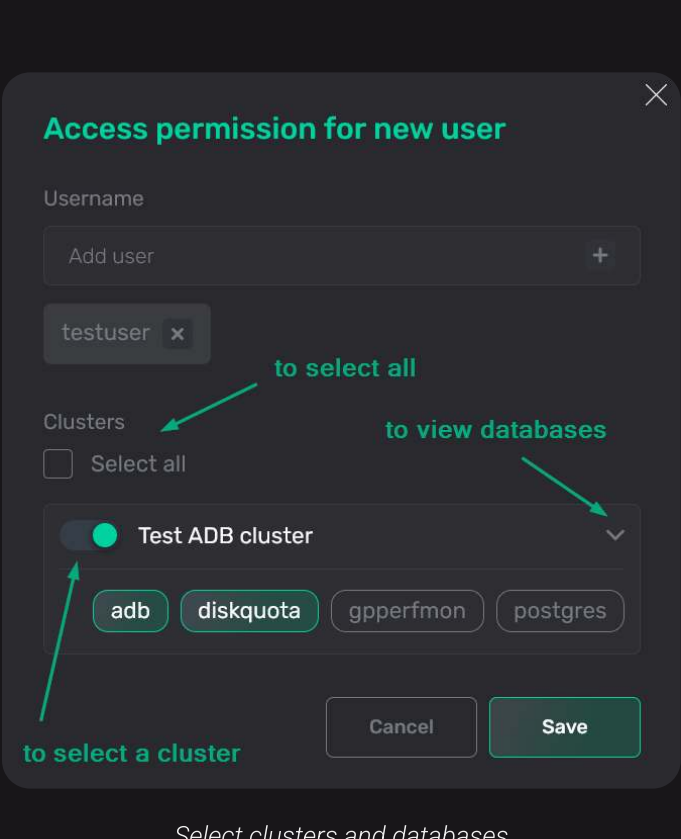
Clusters: ☑ Select all

Test ADB cluster

Add user
1 enter a name
click 2
Cancel
Save

Enter user names

   - In the ***Clusters*** section, select ADB clusters and databases to be accessed by the users added in the previous step. To select all clusters with all databases, set the ***Select all*** flag (it is set by default).



Access permission for new user

Username: testuser

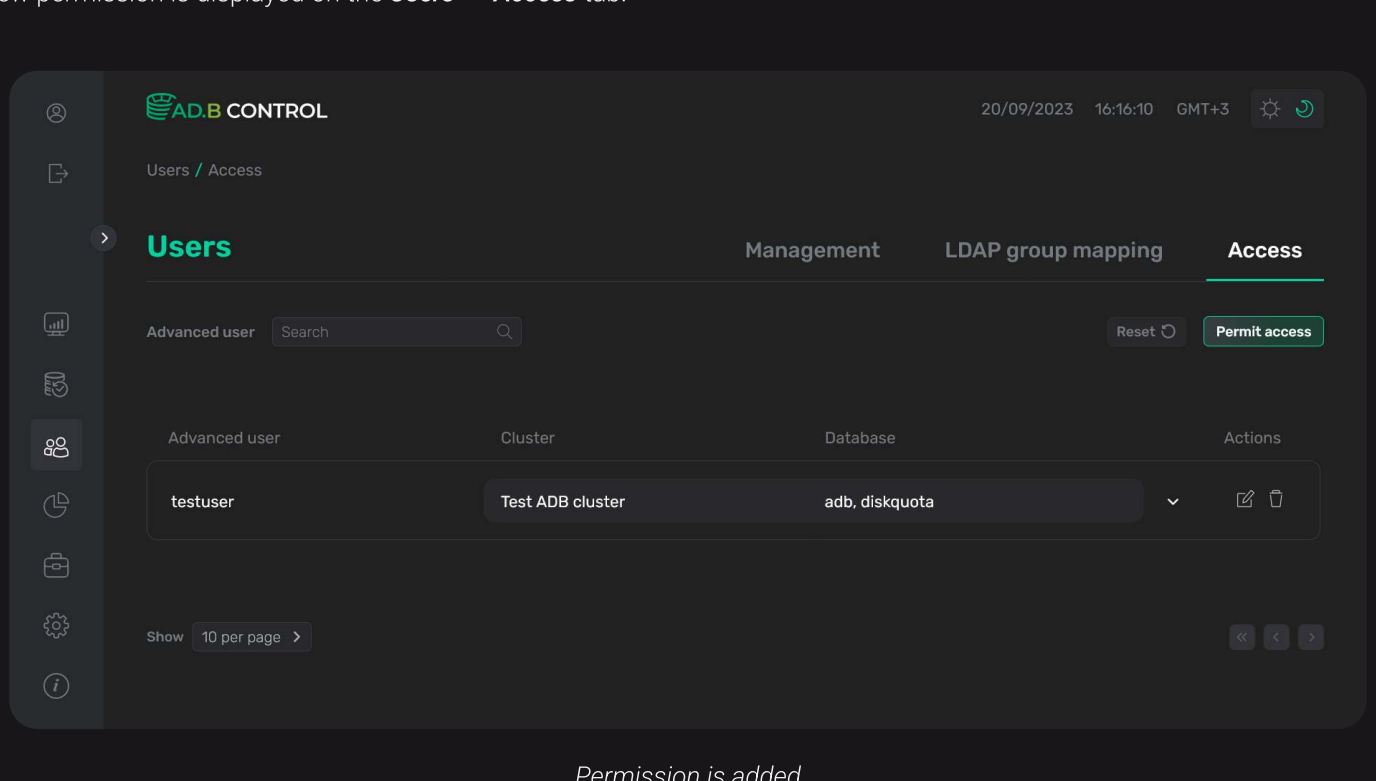
Clusters: ☐ Select all

Test ADB cluster: adb, diskquota

Add user
to select all
to view databases
gpperfmon
postgres
to select a cluster
Cancel
Save

Select clusters and databases

3. Click ***Save***. As a result, a new permission is displayed on the ***Users → Access*** tab.


AD.B CONTROL
20/09/2023 16:16:10 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
Advanced user
Cluster
Database
Actions
Show 10 per page

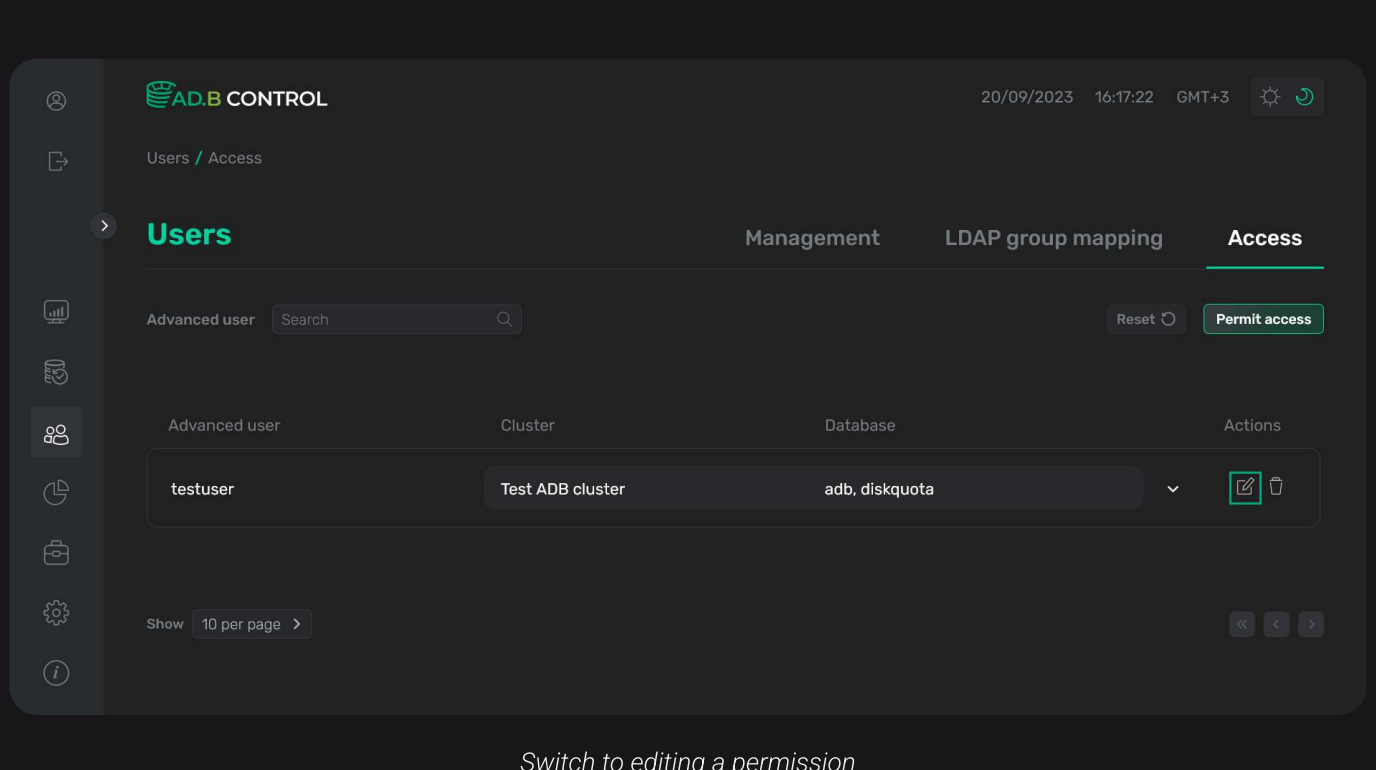
testuser | Test ADB cluster | adb, diskquota

Permission is added

Edit a permission

To edit an existing permission, follow the steps:

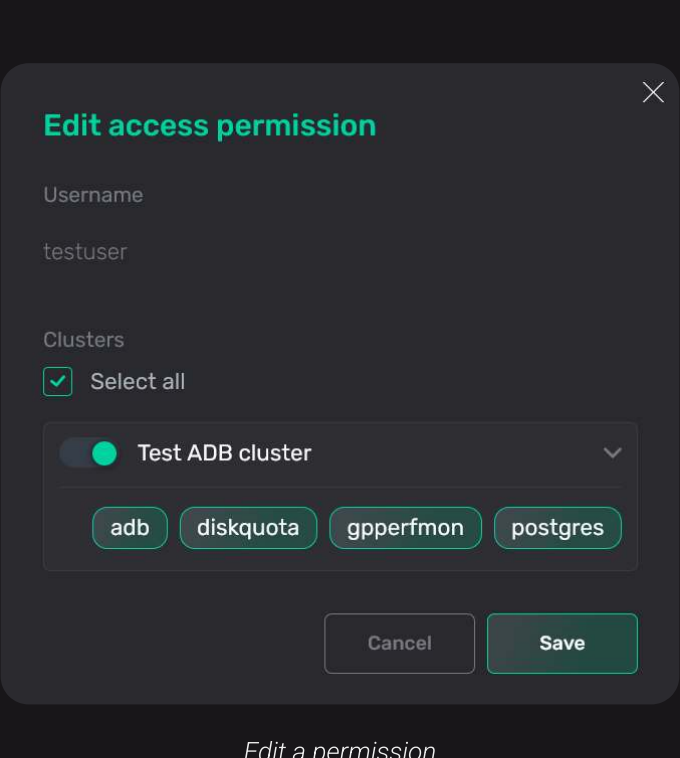
1. Click the icon in the ***Actions*** column on the ***Users → Access*** tab.


AD.B CONTROL
20/09/2023 16:17:22 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
Advanced user
Cluster
Database
Actions
Show 10 per page

testuser | Test ADB cluster | adb, diskquota

Switch to editing a permission

2. In the window that opens, edit necessary fields. All fields are the same as described above. The ***Username*** field value cannot be changed.



Edit access permission

Username: testuser

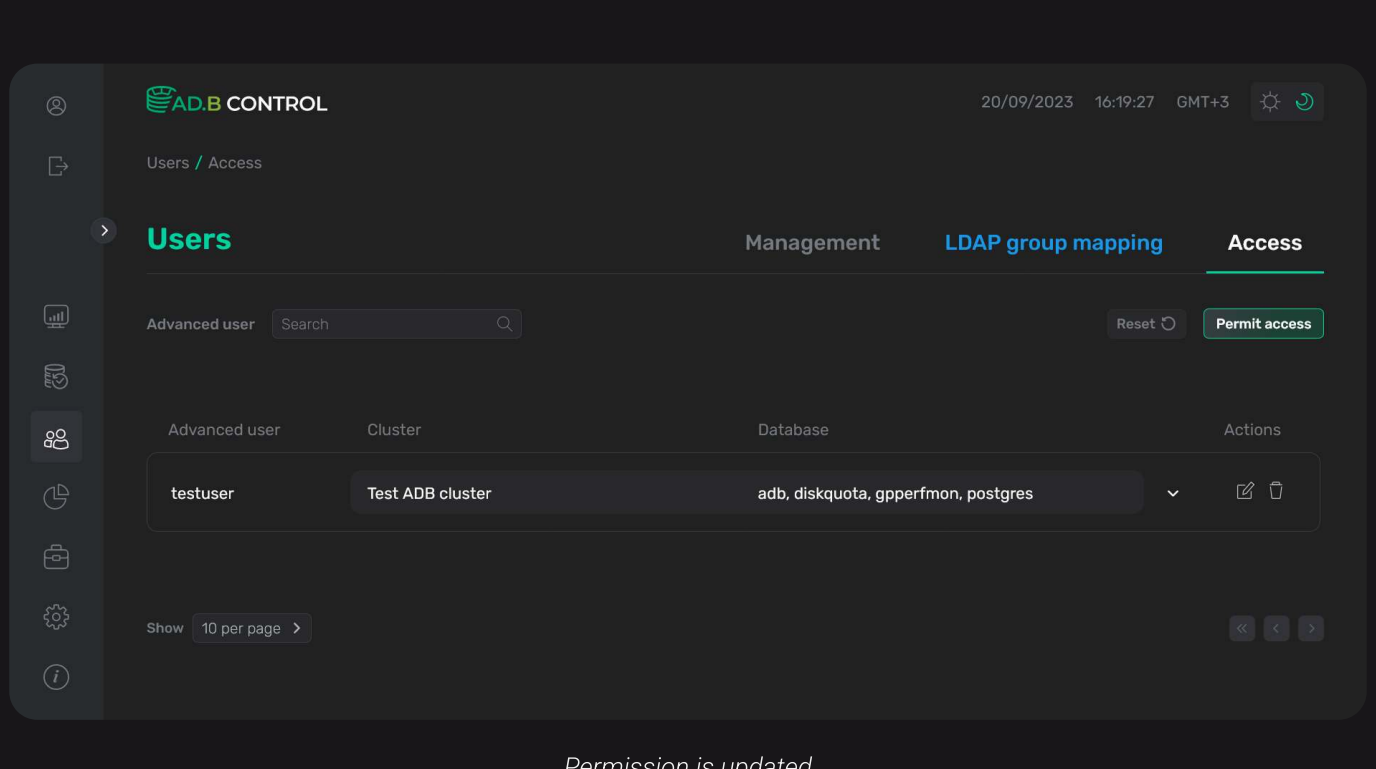
Clusters: ☑ Select all

Test ADB cluster: adb, diskquota, gpperfmon, postgres

Cancel
Save

Edit a permission

3. Click ***Save***. As a result, the permission data is updated on the ***Users → Access*** tab.


AD.B CONTROL
20/09/2023 16:19:27 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
Advanced user
Cluster
Database
Actions
Show 10 per page

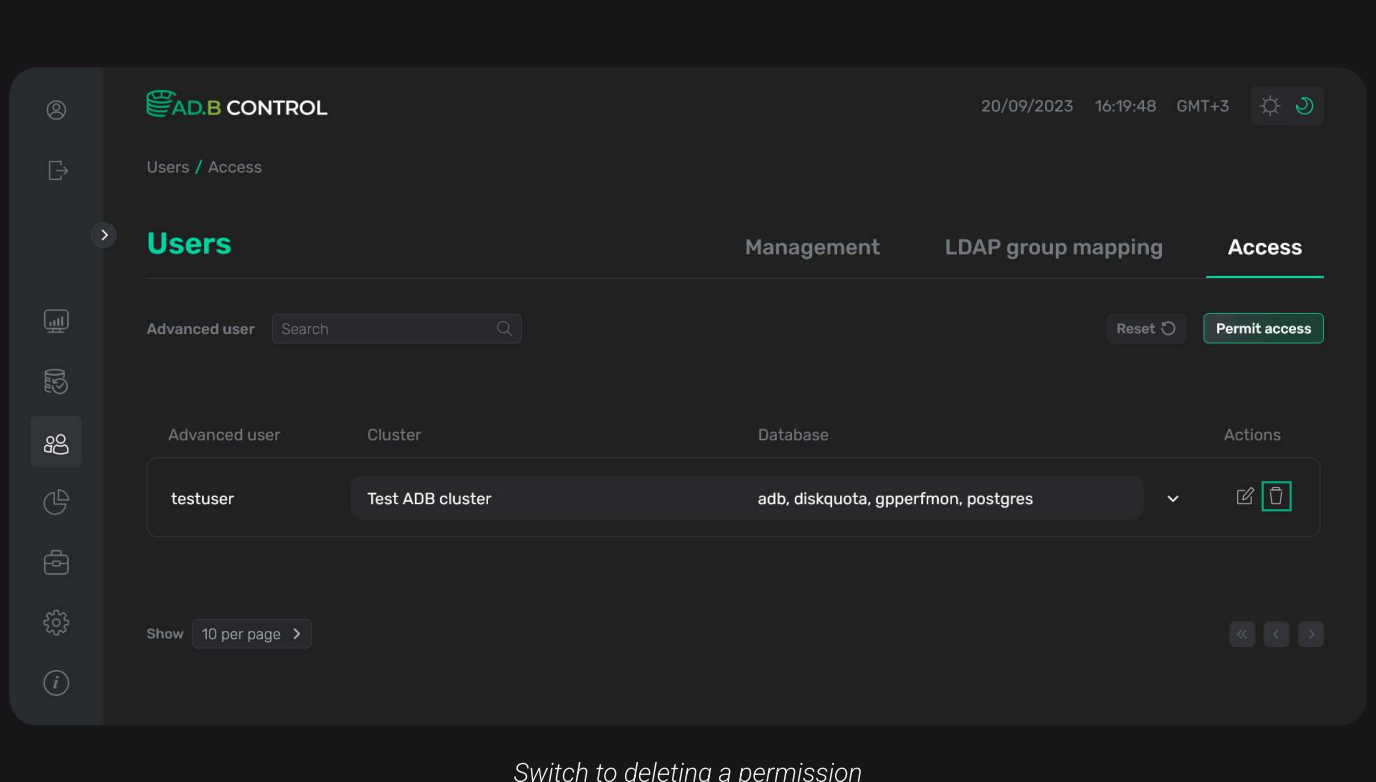
testuser | Test ADB cluster | adb, diskquota, gpperfmon, postgres

Permission is updated

Delete a permission

To delete an existing permission, follow the steps:

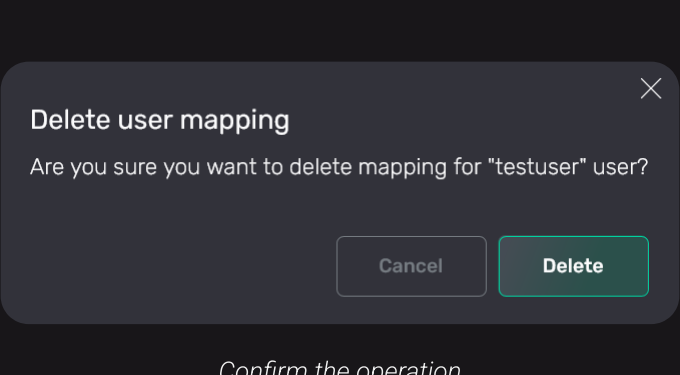
1. Click the icon in the ***Actions*** column on the ***Users → Access*** tab.


AD.B CONTROL
20/09/2023 16:19:48 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
Advanced user
Cluster
Database
Actions
Show 10 per page

testuser | Test ADB cluster | adb, diskquota, gpperfmon, postgres

Switch to deleting a permission

2. In the window that opens, confirm the operation by clicking ***Delete***.



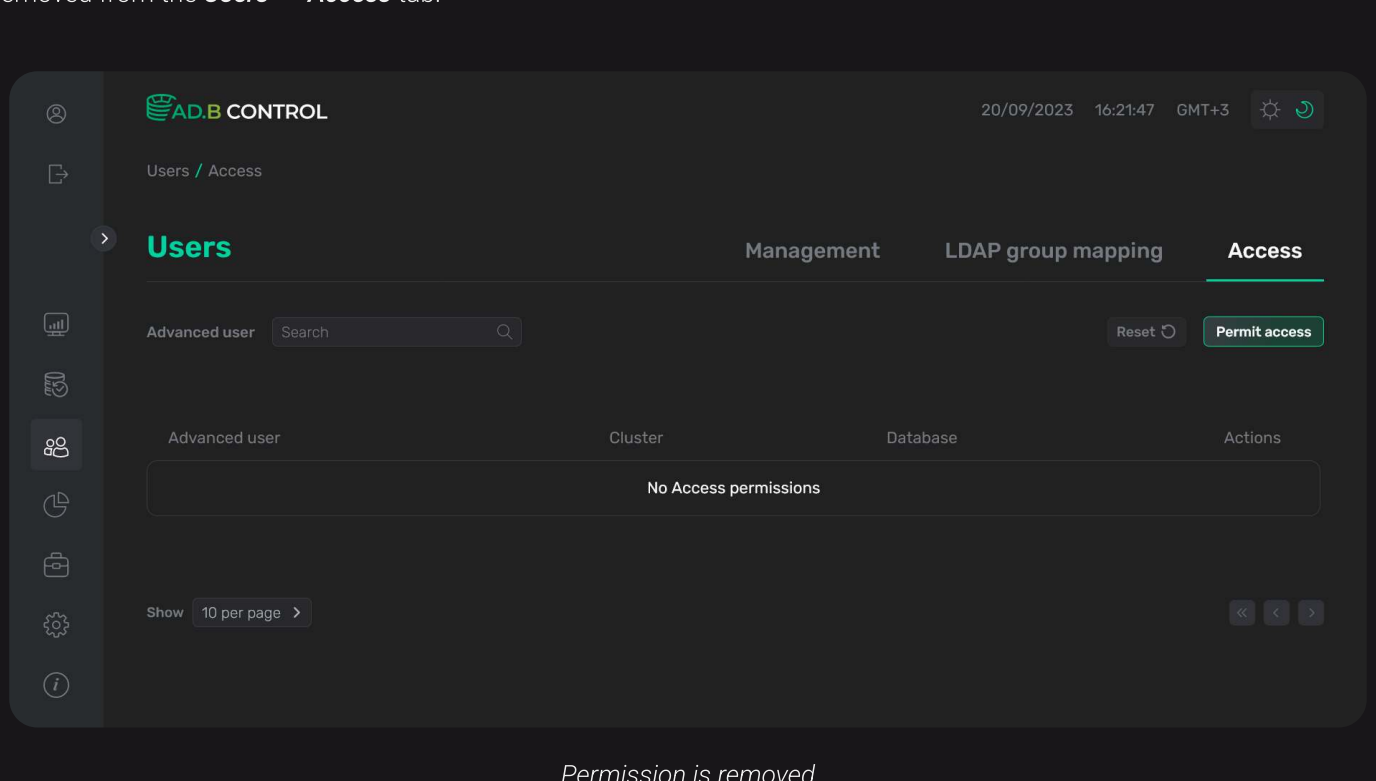
Delete user mapping

Are you sure you want to delete mapping for "testuser" user?

Cancel
Delete

Confirm the operation

As a result, permission is removed from the ***Users → Access*** tab.


AD.B CONTROL
20/09/2023 16:21:47 GMT+3
Users / Access
Users
Management
LDAP group mapping
Access
Advanced user
Search
Reset
Permit access
Advanced user
Cluster
Database
Actions
No Access permissions
Show 10 per page

Permission is removed

User sessions

Daria Barysheva

Contents

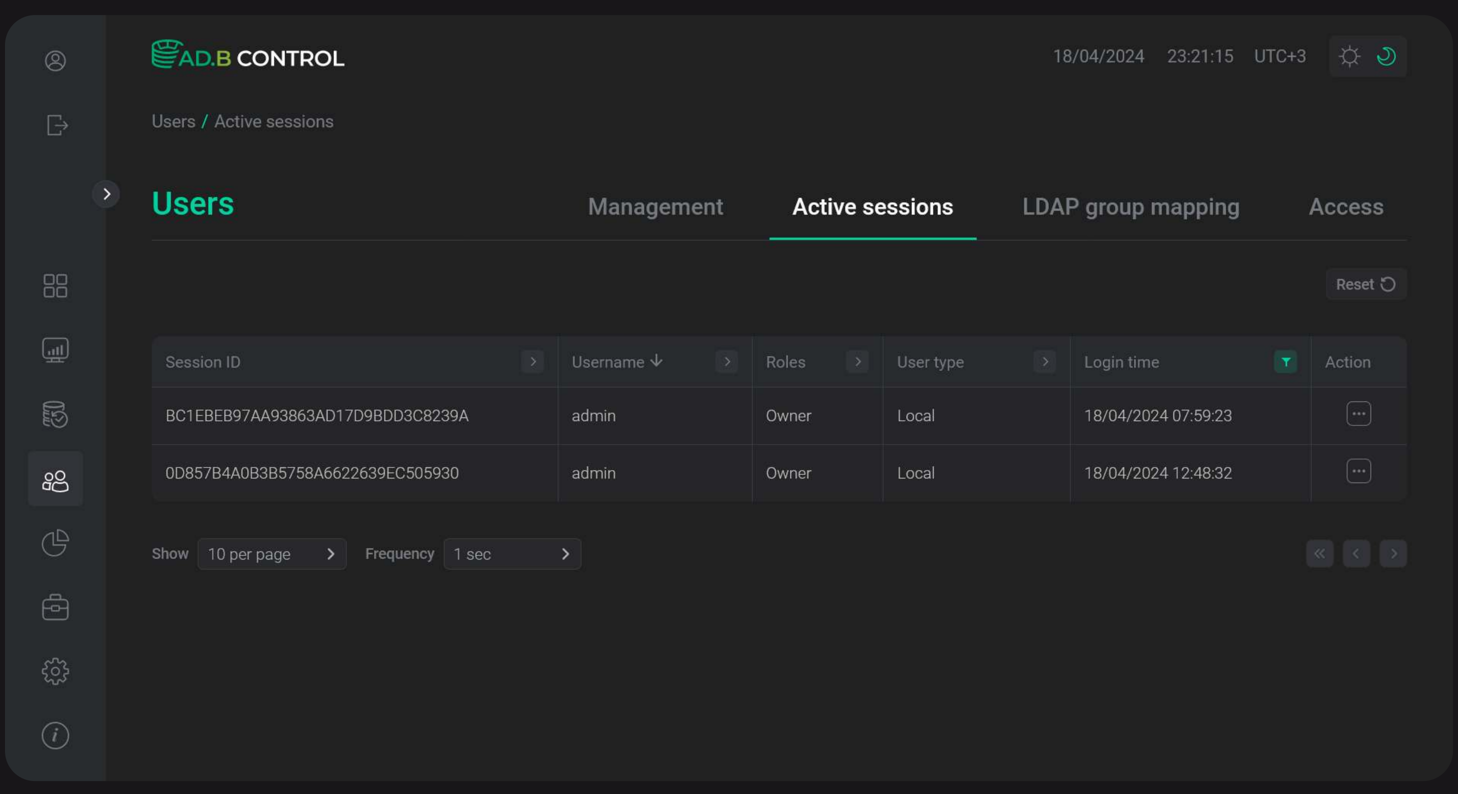
To Table of Contents

View a list of sessions
Terminate sessions

The ***Users*** **→** ***Active sessions*** tab is designed to monitor active sessions of ADB Control users. This page also allows you to terminate selected sessions.

View a list of sessions

On the ***Users*** **→** ***Active sessions*** tab, there is a table with the following information on sessions.


AD.B CONTROL
18/04/2024 23:21:15 UTC+3
Users / Active sessions
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Show 10 per page
Frequency 1 sec

Session ID	Username	Roles	User type	Login time	Action
BC1EBEB97AA93863AD17D9BDD3C8239A	admin	Owner	Local	18/04/2024 07:59:23	•••
0D857B4A0B3B5758A6622639EC505930	admin	Owner	Local	18/04/2024 12:48:32	•••

List of sessions

The "Users → Active sessions" tab fields

Field	Description
Session ID	A session identifier
Username	A name of the user who connected to ADB Control
Roles	A role of the user who connected to ADB Control
User type	A type of the user who connected to ADB Control. Possible values: ▪ `AD` — an LDAP user; ▪ `Local` — an ADB Control user who uses basic authentication; ▪ `System` — a system user.
Login time	A session start timestamp

In the column headers of the table with a list of sessions, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Roles***), you can select a value from the drop-down list. For some columns (e.g. ***Session ID***), the search value should be entered. For columns that show date and time (e.g. ***Login time***), the time range can be selected from the calendar.

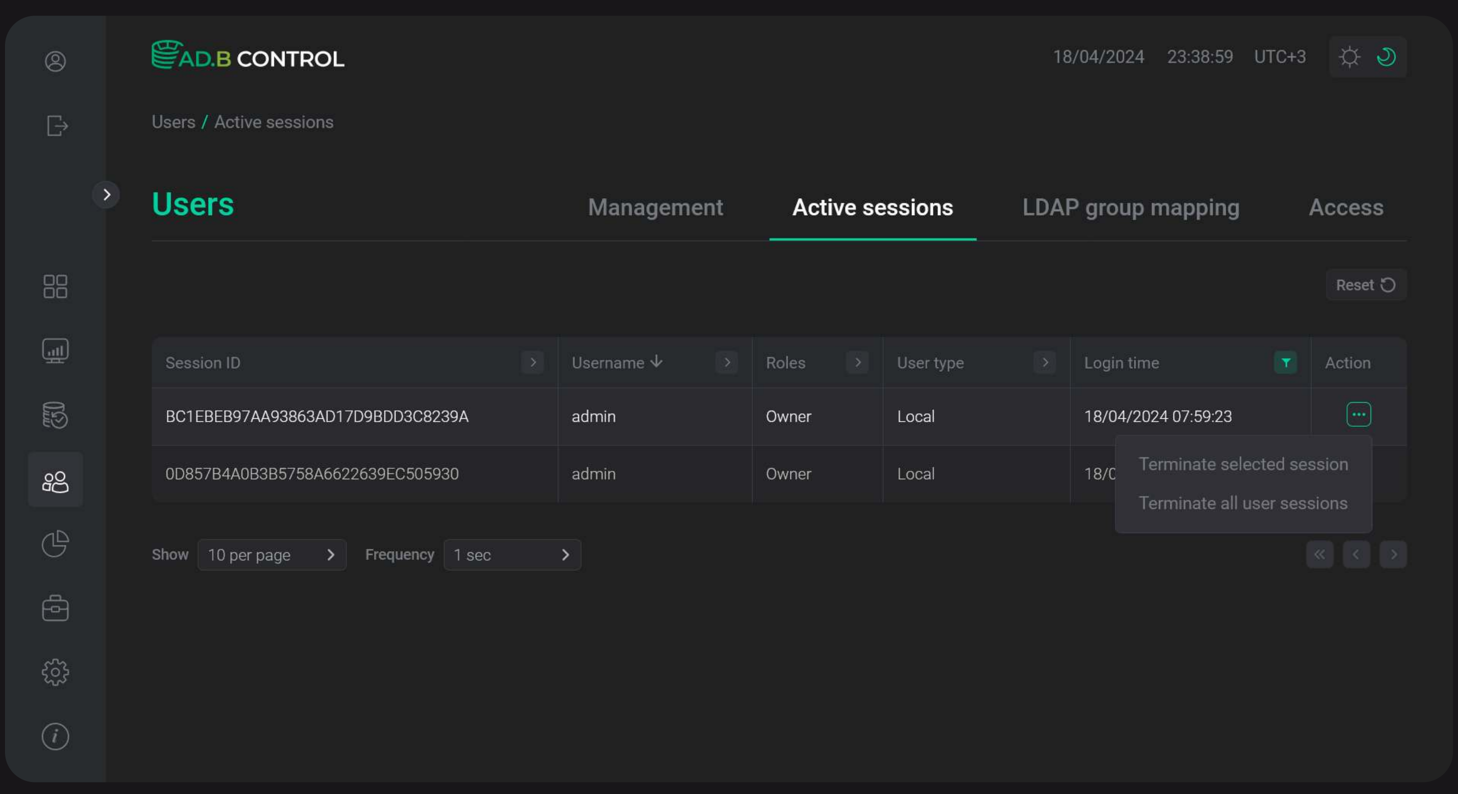
The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Terminate sessions

NOTE

Termination of user sessions is available for ADB Control users with appropriate permissions (see `Manage user sessions` in the Authorization article).

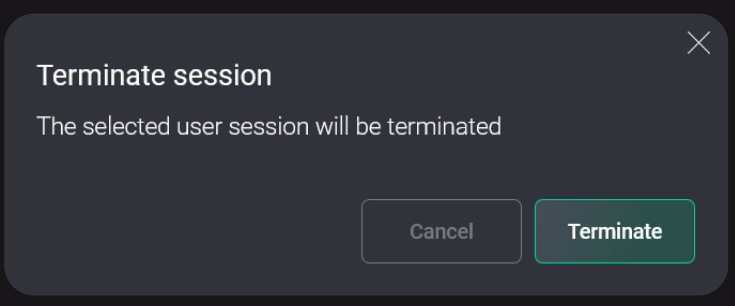
To terminate a session, click ••• in the ***Actions*** column on the ***Users*** **→** ***Active sessions*** tab.


AD.B CONTROL
18/04/2024 23:38:59 UTC+3
Users / Active sessions
Users
Management
Active sessions
LDAP group mapping
Access
Reset
Session ID
Username
Roles
User type
Login time
Action
BC1EBEB97AA93863AD17D9BDD3C8239A
admin
Owner
Local
18/04/2024 07:59:23
0D857B4A0B3B5758A6622639EC505930
admin
Owner
Local
Terminate selected session
Terminate all user sessions
Show 10 per page
Frequency 1 sec

Go to terminating a session

Select one of the options:

- ***Terminate selected session*** — terminates only one selected session. To confirm the action, click ***Terminate*** in the window that opens.

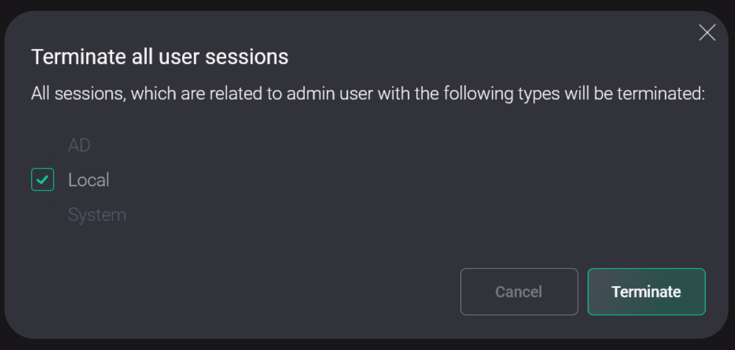

Cancel
Terminate

Terminate session

The selected user session will be terminated

Confirm termination of one session

- ***Terminate all user sessions*** — terminates all sessions of the user who started the selected session. In the window that opens, you can set the user type (see ***User type*** above). To confirm the action, click ***Terminate***.


Cancel
Terminate

Terminate all user sessions

All sessions, which are related to admin user with the following types will be terminated:

☐ AD
☑ Local
☐ System

Confirm termination of several sessions

SSL encryption

Daria Barysheva

Contents

To Table of Contents

ADB Control

External databases

- PostgreSQL
- ClickHouse

ADB Control

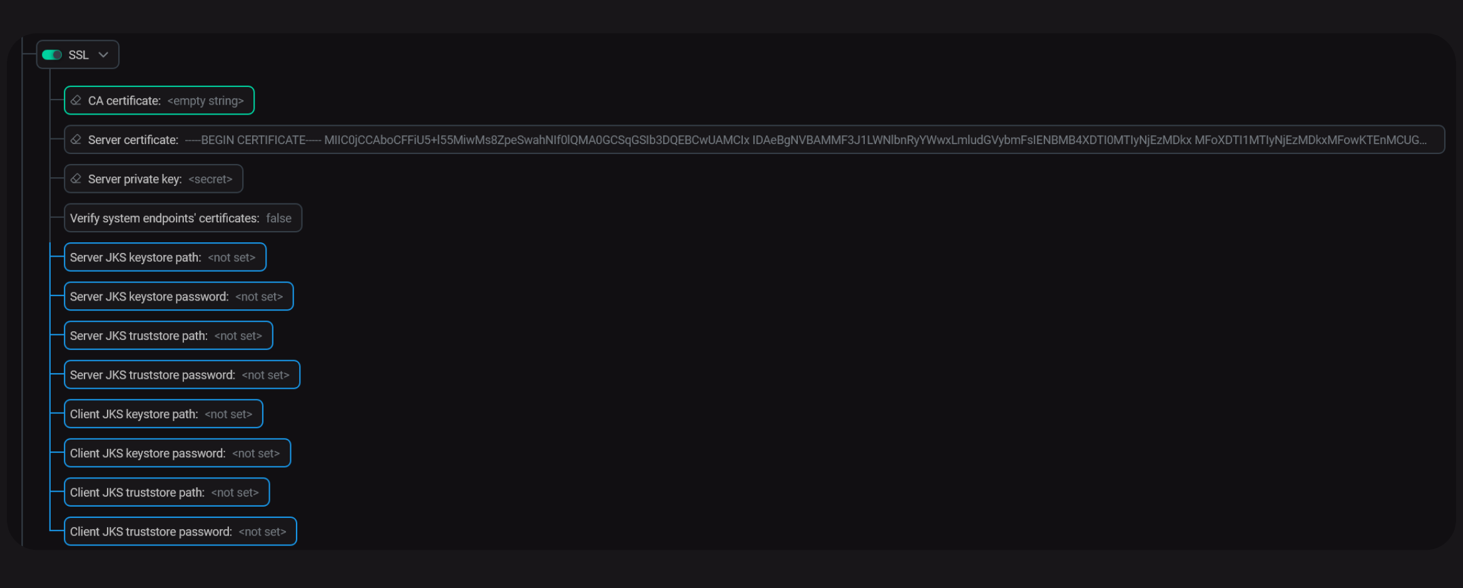
You can secure communication between ADB Control web server and clients using the HTTPS protocol, which is based on SSL encryption. To configure access to ADB Control via HTTPS, follow the steps:

1. Open the ADB Control service configuration page in ADCM.
2. Set the ***Show advanced*** flag.
3. Activate the ***SSL*** switcher.
4. Fill in the following fields if you need to use your own certificates.

IMPORTANT

If no settings are filled in, ADB Control will create certificates by itself.

Parameter	Description	Default value
CA certificate	Server certificate authority (CA) file contents in the CRT format (**.crt* file)	—
Server certificate	Server certificate contents in the CRT format (**.crt* file)	—
Server private key	Server private key contents (**.key* file)	—
Verify system endpoints' certificates	A flag that indicates whether to verify certificates of system endpoints. When using the ***Verify system endpoints' certificates*** option in the ADB Control configuration with self-signed certificates, ensure that you added the root and intermediate CA certificates to the OS trusted root certificate stores. In order to use the ***Verify system endpoints' certificates*** option with your own certificates, for proper verification, add the ADB Control IP address to the `CN` field of the *san.cnf* file (`SAN`) and exclude DNS from the `alt_names` section (leave only IP)	false
Server JKS keystore path	A path to the keystore file in the JKS format for the ADB Control server part. For example, */opt/adcc/ssl/keystore.jks*	—
Server JKS keystore password	A password that is used for connecting to ***Server JKS keystore path***	—
Server JKS truststore path	A path to the truststore file in the JKS format for the ADB Control server part. For example, */opt/adcc/ssl/truststore.jks*	—
Server JKS truststore password	A password that is used for connecting to ***Server JKS truststore path***	—
Client JKS keystore path	A path to the keystore file in the JKS format for the ADB Control client part (ADB Control/ADBM agents). For example, */opt/ssl/keystore.jks*	—
Client JKS keystore password	A password that is used for connecting to ***Client JKS keystore path***	—
Client JKS truststore path	A path to the truststore file in the JKS format for the ADB Control client part (ADB Control/ADBM agents). For example, */opt/ssl/truststore.jks*	—
Client JKS truststore password	A password that is used for connecting to ***Client JKS truststore path***	—



CA certificate: <empty string>
Server certificate: ——BEGIN CERTIFICATE—— [illegible]
Server private key: <secret>
Verify system endpoints' certificates: false
Server JKS keystore path: <not set>
Server JKS keystore password: <not set>
Server JKS truststore path: <not set>
Server JKS truststore password: <not set>
Client JKS keystore path: <not set>
Client JKS keystore password: <not set>
Client JKS truststore path: <not set>
Client JKS truststore password: <not set>

Fill in the SSL parameters

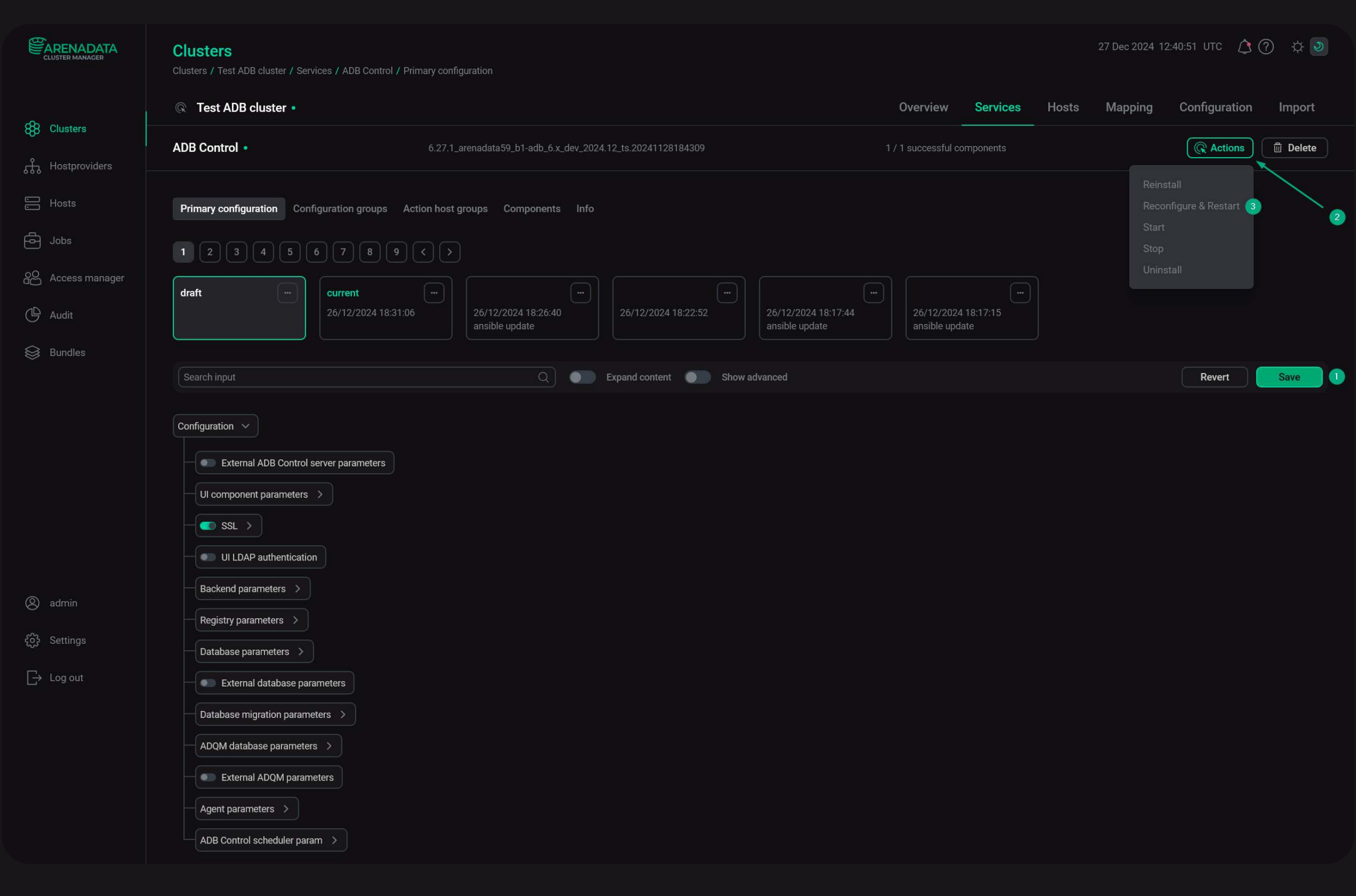
Recommendations

- The *adcc_ca.crt*, *adcc.crt*, and *adcc.key* files are created by default on the host where ADB Control is deployed. They are located in the */opt/adcc/ssl* folder. You can specify these files for testing purposes.
- For production systems, you should use a trusted certificate authority (CA) to sign certificates. To create a real SSL certificate and register it with the CA, use the `openssl` utility.
- In case of expanding ADB, SSL certificates should be re-generated to add the corresponding `ServerAltNames` for new ADB hosts.
- To connect an external ADB cluster to existing ADB Control with SSL enabled, certificates with `ServerAltNames` for hosts of both ADB clusters should be specified in the settings of a main and an external ADB cluster.
- To hide the truststore password in Vault, set the ***Use PEM CA for vault*** parameter in the ***Vault integration*** section of the ADB cluster configuration to `true`.
- In the ***CA certificate*** field, it is necessary to enter certificate chains from the root and intermediate CA for the correct cluster operation.
  The order of PEM CA certificates in the chain should be as follows:
  - Sub-CA Certificate — start.
  - Root CA Certificate — end.

  This also applies to the custom JKS created with `keytool`.

  When using ADB Control as an external system, ensure that the client truststore contains CA from the server.

5. Click ***Save***.
6. Apply the ***Reconfigure & Restart*** action to the ADB Control service.


ARENADATA
CLUSTER MANAGER
Clusters
Clusters / Test ADB cluster / Services / ADB Control / Primary configuration
27 Dec 2024 12:40:51 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADB cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADB Control
6.27.1_arenadata59_b1-adb_6.x_dev_2024.12_ts.20241128184309
1 / 1 successful components
Actions
Delete
Reinstall
Reconfigure & Restart
Start
Stop
Uninstall
3
2
Primary configuration
Configuration groups
Action host groups
Components
Info
draft
current
26/12/2024 18:31:06
26/12/2024 18:26:40
ansible update
26/12/2024 18:22:52
26/12/2024 18:17:44
ansible update
26/12/2024 18:17:15
ansible update
Search input
Expand content
Show advanced
Revert
Save
1
Configuration
External ADB Control server parameters
UI component parameters
SSL
UI LDAP authentication
Backend parameters
Registry parameters
Database parameters
External database parameters
Database migration parameters
ADQM database parameters
External ADQM parameters
Agent parameters
ADB Control scheduler param

Apply changes

After all steps are completed successfully, the ADB Control server port (by default, `81`) will be used for SSL. The ADB Control address will look as follows: *https://<ADB Control IP address>:81*.

External databases

Starting with ADB Control 4.11.0, after you enable SSL as described above, SSL encryption is automatically used for connections to internal databases of ADB Control and ADBM. If external databases are used in ADB Control (Metrics DB (ClickHouse), Query DB (PostgreSQL), or Offload DB (PostgreSQL)) and/or external PostgreSQL is used in ADBM, you should manually enable SSL encryption in these databases.

PostgreSQL

1. Add the following options to the *postgresql.conf* file:

   ```
   ssl = on
   ssl_cert_file = '<server_crt_path>'
   ssl_key_file = '<server_key_path>'
   ```

   where:

   - `<server_crt_path>` — a path to the server certificate on the PostgreSQL server. For example, */pg_data1/adpg14/server.crt*.
   - `<server_key_path>` — a path to the server private key on the PostgreSQL server. For example, */pg_data1/adpg14/server.key*.

2. Add the following entry to the *pg_hba.conf* file:

```
hostssl    all  all       0.0.0.0/0      md5
```

3. Restart PostgreSQL to apply changes.
4. To ensure that SSL is successfully configured, you can connect to PostgreSQL via `psql`. When connecting, define the following options: `-U` — user name, `-d` — database, `-h` — host IP:

```
$ psql -U adcc -d metrics -h 10.92.41.125
```

   After you enter the user password, the following information should be displayed:

```
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
```

5. Perform standard actions to configure external PostgreSQL for ADB Control or ADBM. If you configure Offload DB, refer to Configure job policies → Metrics offload.

IMPORTANT

- To fill in the `<server_crt_path>` and `<server_key_path>` fields, you should first generate your own certificate and key accordingly. You can also reuse the certificate and key generated by ADB Control (see *adcc.crt* and *adcc.key* in ADB Control). But it is important that the ***SSL*** section in the ADB Control configuration contains the same certificates that are specified in the PostgreSQL file *postgresql.conf*.
- The current implementation does not provide authentication based on client certificates, only SSL encryption of traffic is provided. For this reason, the `md5` authentication type is used in *pg_hba.conf*, not `cert`. Also, it is not required to enter a CA certificate in the `ssl_ca_file` field of the *postgresql.conf* file.
- If ADPG is used as external PostgreSQL, you can configure SSL (including certificates generation) according to the Enable SSL encryption article.

ClickHouse

For more information on how to configure SSL encryption in ClickHouse, you can see ClickHouse documentation. The minimal configuration steps are listed below:

1. Apply the following changes to the server configuration file *config.xml* on each node of the ClickHouse cluster:

   - Enable HTTPS ports and comment HTTP ports:

     ```
     <https_port>8443</https_port>
     <!--<http_port>8123</http_port>-->
     <interserver_https_port>9010</interserver_https_port>
     <!--<interserver_http_port>9009</interserver_http_port>-->
     ```

   - Configure the `OpenSSL` section as follows:

     ```
     <openSSL>
         <server>
             <certificateFile>/etc/clickhouse-server/certs/server.crt</certificateFile>
             <privateKeyFile>/etc/clickhouse-server/certs/server.key</privateKeyFile>
             <verificationMode>relaxed</verificationMode>
             <caConfig>/etc/clickhouse-server/certs/ca.crt</caConfig>
             <cacheSessions>true</cacheSessions>
             <disableProtocols>sslv2,sslv3</disableProtocols>
             <preferServerCiphers>true</preferServerCiphers>
         </server>
         <client>
             <loadDefaultCAFile>false</loadDefaultCAFile>
             <caConfig>/etc/clickhouse-server/certs/ca.crt</caConfig>
             <cacheSessions>true</cacheSessions>
             <disableProtocols>sslv2,sslv3</disableProtocols>
             <preferServerCiphers>true</preferServerCiphers>
             <verificationMode>relaxed</verificationMode>
             <invalidCertificateHandler>
                 <name>RejectCertificateHandler</name>
             </invalidCertificateHandler>
         </client>
     </openSSL>
     ```

     where the following paths are used (can be different in your environment):

     - */etc/clickhouse-server/certs/server.crt* — a path to the server certificate on the ClickHouse node.
     - */etc/clickhouse-server/certs/server.key* — a path to the server private key on the ClickHouse node.
     - */etc/clickhouse-server/certs/ca.crt* — a path to the CA certificate on the ClickHouse node.

2. Restart ClickHouse to apply changes.
3. Perform standard actions to configure external ClickHouse for ADB Control.

IMPORTANT

To fill in the `<certificateFile>` and `<privateKeyFile>` fields, you should first generate your own certificate and key accordingly. You can also reuse the certificate and key generated by ADB Control (see *adcc.crt* and *adcc.key* in ADB Control). But it is important that the ***SSL*** section in the ADB Control configuration contains the same certificates that are specified in the ClickHouse file *config.xml*.

Configure job policies

Daria Barysheva

AD.B CONTROL

Contents

To Table of Contents

Data cleanup

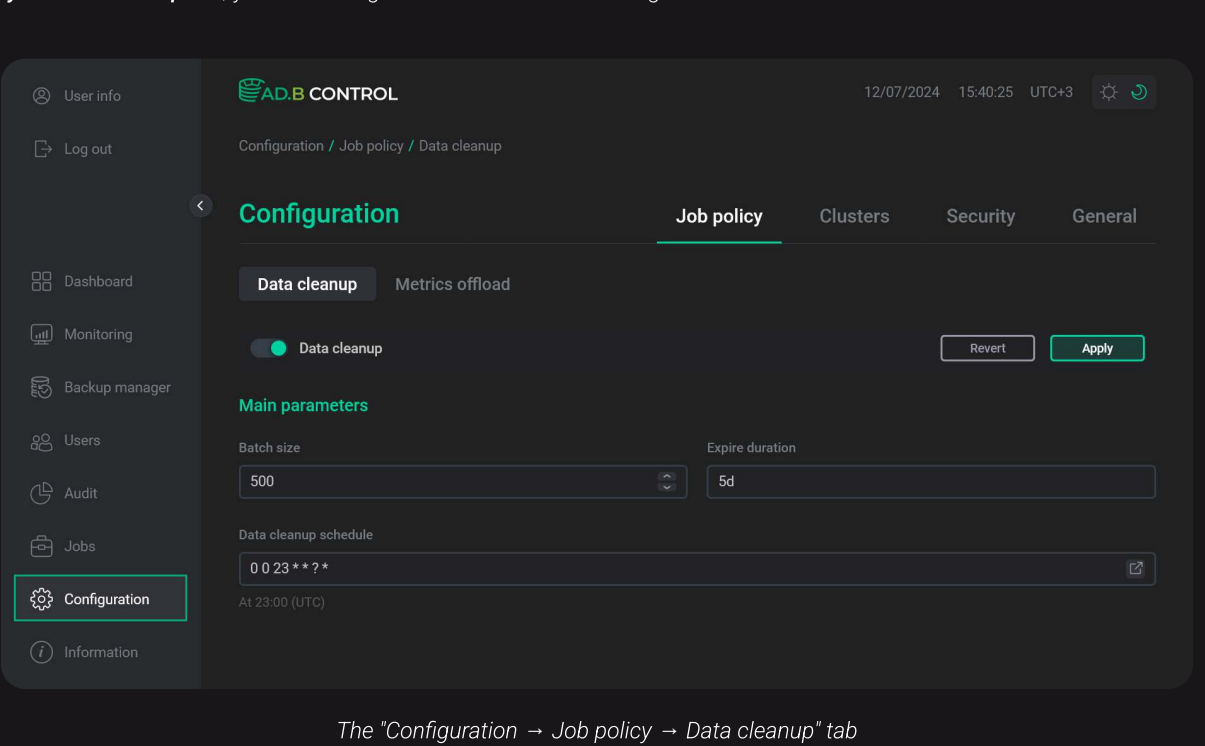
Metrics offload

- Configuration
- Structure of offloaded data

The ***Configuration*** → ***Job policy*** page in the ADB Control web interface allows you to configure schedules for different system jobs. This page includes two tabs, each of which is described in detail below.

Data cleanup

On the ***Configuration*** → ***Job policy*** → ***Data cleanup*** tab, you can configure a schedule for removing old metrics from ADB Control.



Batch size: 500

Expire duration: 5d

Data cleanup schedule: 0 0 23 * * ? *

At 23:00 (UTC)

The "Configuration → Job policy → Data cleanup" tab

To configure the job, follow the steps:

1. Activate the ***Data cleanup*** switcher.
2. Fill in the following parameters in the ***Main parameters*** section.

Field	Description	Default value
Batch size	A maximum number of rows in the data batch	500
Expire duration	A maximum time period that is used to store metrics in ADB Control. If this period expires, the metrics will be removed during the next job launch. Use the following format — `<value><unit>` (without spaces), where `<unit>` can have the following values: - `ms` — milliseconds; - `sec` — seconds; - `min` — minutes; - `hr` — hours; - `d` — days; - `w` — weeks. Examples: `5min`, `1hr`, `1d`. The minimal value is `100ms`; the maximum is `1w`	5d
Data cleanup schedule	A schedule for automatic data cleanup (`cron` expression). The ↗ icon allows you to edit the schedule in the separate form ***Cron expression generator***	0 0 23 * * ? *

3. Click ***Apply***. The ***Revert*** button is designed to undo the changes that have not been yet saved by clicking ***Apply***.

If all steps are completed successfully, the next launch time of the `Data cleanup` job is updated on the Jobs → ADB Control → Schedule page.

> **CAUTION**
>
> - The data cleanup job is enabled in ADB Control by default. It is not recommended to disable the job, since it can cause a lack of disk space.
> - For long-term storage of historical data, you can configure loading metrics to an external database.

Metrics offload

Configuration

> **CAUTION**
>
> - Metrics offload to ADB as a target database cannot be configured if the `gp_default_storage_options` configuration parameter in the source ADB cluster has the `appendonly=true` value.
> - Ensure that you do not set up for metrics offload a database of the same ADB cluster that is monitored by ADB Control (where metrics are collected). It is necessary to avoid monitoring and exporting records about export itself.

On the ***Configuration*** → ***Job policy*** → ***Metrics offload*** tab, you can configure a schedule for uploading ADB Control metrics and events to an external database (PostgreSQL/ADB). To configure the job, follow the steps:

1. On the target host, ensure you have a database to store the uploaded metrics. Its name will be used when configuring the JDBC connection later.
2. Provide the ADB Control access to the selected external database. PostgreSQL requires the following record in the *pg_hba.conf* file.

Without SSL

```
host    <database_name>  <user_name>       <adbc_address>      trust
```

where:

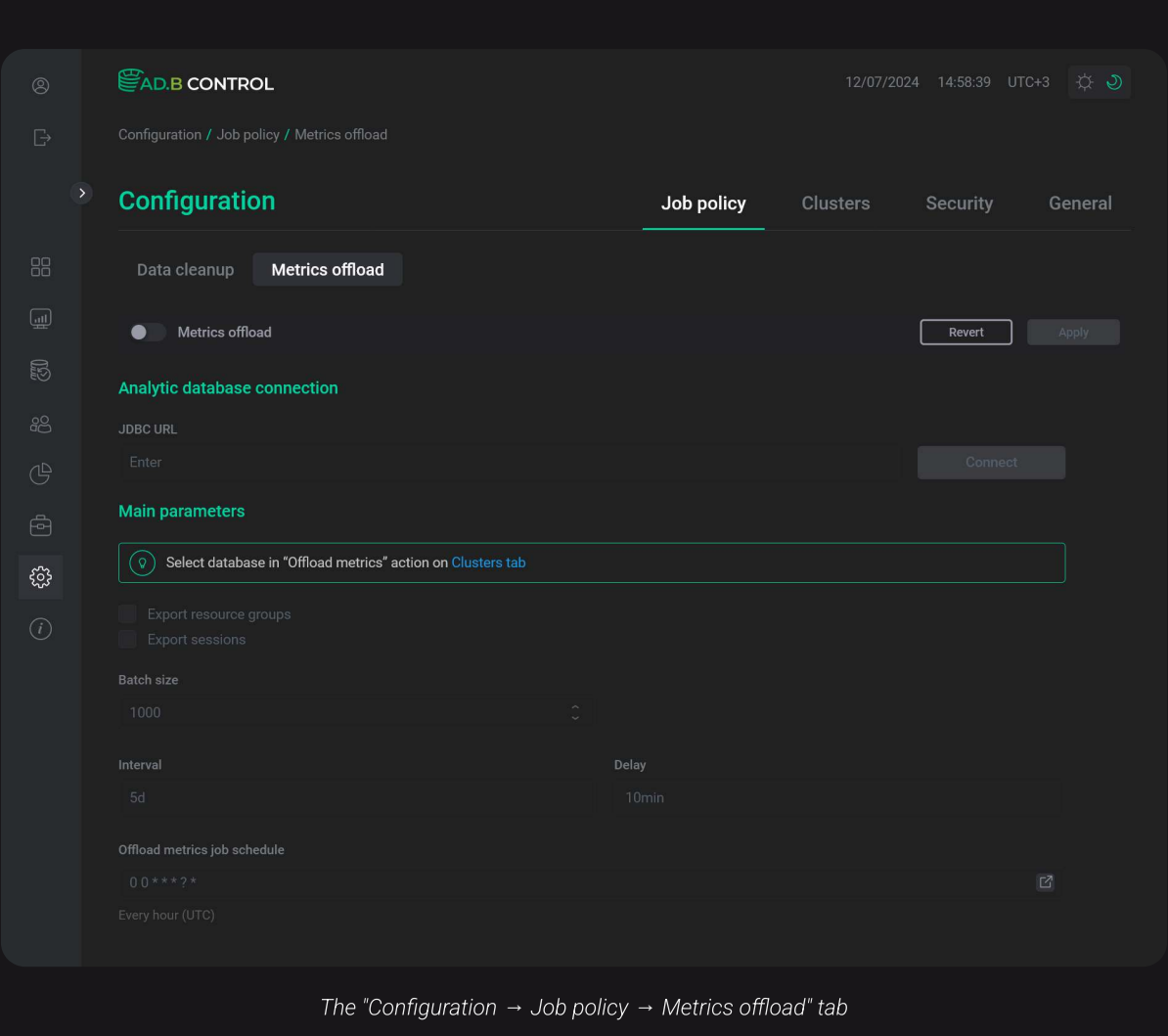
- `<database_name>` — a database name in PostgreSQL. For example, `adbc_metrics`.
- `<user_name>` — a user name in PostgreSQL. For example, `postgres`.
- `<adbc_address>` — an IP address of the host where ADB Control is deployed (with a subnet number). For example, `10.92.43.139/32`.

With SSL

```
hostssl    all  all       0.0.0.0/0      md5
```

For more information on how to configure PostgreSQL for using SSL, see SSL encryption.

3. Go to the ***Configuration*** → ***Job policy*** → ***Metrics offload*** tab in the ADB Control interface.



Batch size: 1000

Interval: 5d

Delay: 10min

Offload metrics job schedule: 0 0 * * * ? *

Every hour (UTC)

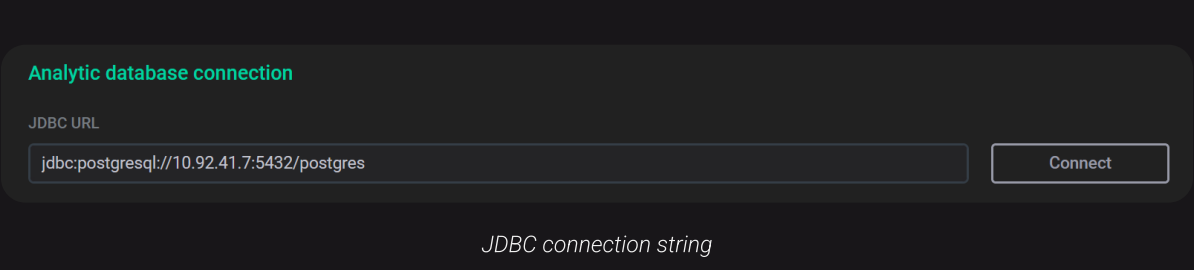
The "Configuration → Job policy → Metrics offload" tab

4. Activate the ***Metrics offload*** switcher.
5. In the ***JDBC URL*** field, enter a URL of the JDBC connection to the host where the target database is located. For PostgreSQL, use the following format:

```
jdbc:postgresql://<external_ip>:5432/<db_name>
```

where:

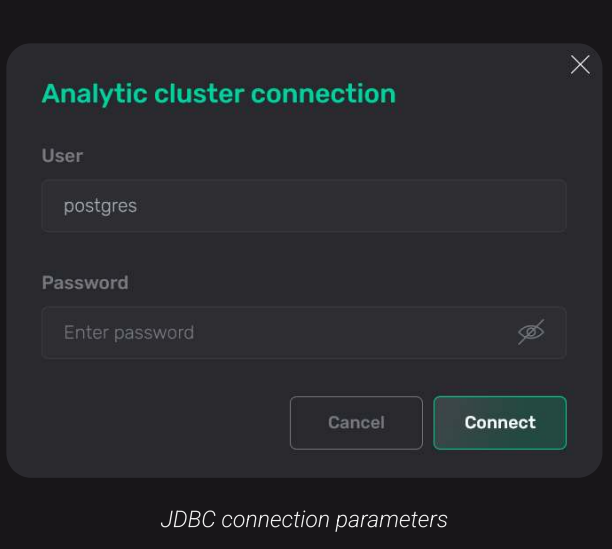
- `<external_ip>` — an IP address of the host where the target database is located. For example, `10.92.41.7`.
- `<db_name>` — a target database name. For example, `postgres`.



JDBC URL: jdbc:postgresql://10.92.41.7:5432/postgres

JDBC connection string

6. Click ***Connect***. In the window that opens, fill in the following fields:
   - ***User*** — a user name. In the following example, the default user `postgres` is used.
   - ***Password*** — a user password.



User

postgres

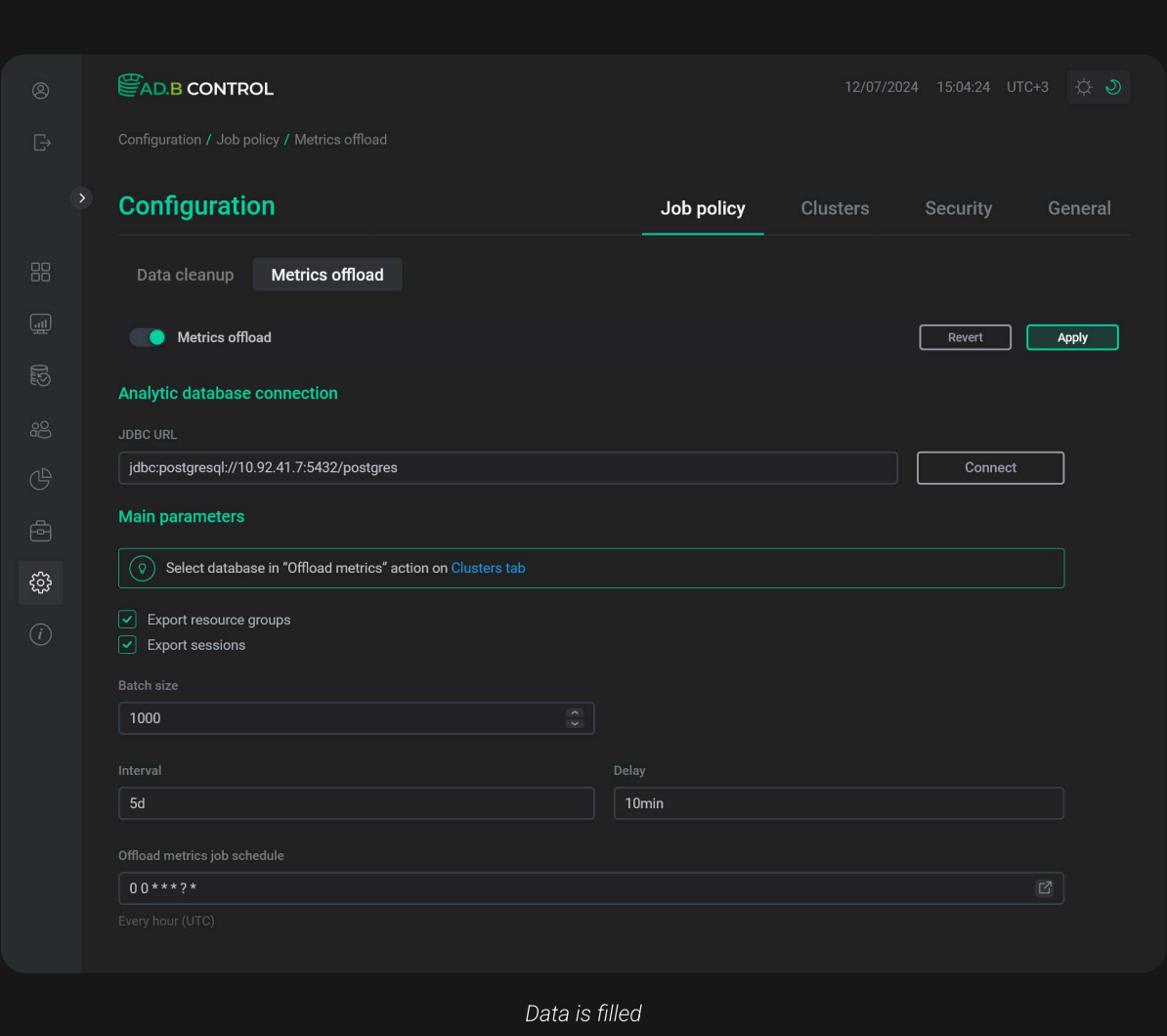
Password

JDBC connection parameters

7. Click ***Connect***.
8. Fill in the following fields in the ***Main parameters*** section.

Field	Description	Default value
Export sessions	The flag that indicates whether to export information on sessions (see session)	False
Batch size	A maximum number of rows in the data batch that is sent to an external database	1000
Interval	A maximum date range for metrics offload (in days)	5d
Delay	The time delay that is used to look for new metrics to export (in minutes)	10min
Offload metrics job schedule	A schedule for the automatic data export to an external database (`cron` expression). The ↗ icon allows you to edit the schedule in the separate form ***Cron expression generator***	0 0 * * * ? *

9. Click ***Apply***. The ***Revert*** button is designed to undo the changes that have not been yet saved by clicking ***Apply***.



JDBC URL: jdbc:postgresql://10.92.41.7:5432/postgres

☑ Export resource groups

☑ Export sessions

Batch size: 1000

Interval: 5d

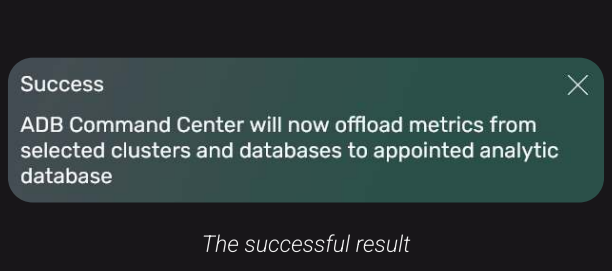
Delay: 10min

Offload metrics job schedule: 0 0 * * * ? *

Every hour (UTC)

Data is filled

10. Get the following message on the successful result.



Success

ADB Command Center will now offload metrics from selected clusters and databases to appointed analytic database

The successful result

11. As a result, the `adcc` schema should be created in the external database. The tables in this schema will store metrics and events collected from ADB Control. To check whether all necessary objects are created, you can run the following query in PostgreSQL:

```sql
SELECT table_name
FROM information_schema.tables
WHERE table_schema = 'adcc' ORDER BY  table_name;
```

Result:

```
        table_name
-----------------------------
 audit_auth
 audit_auth_adb
 audit_operation
 cluster
 cluster_query_metric
 query
 query_error
 relation_statistic
 session
 transaction
(10 rows)
```

If all steps are completed successfully, the new scheduled job `Export job` should be added to the Jobs → ADB Control → Schedule page.

> **IMPORTANT**
>
> In addition to the job schedule, you should specify the clusters and databases which metrics should be exported to the external database. For more information, see Cluster management.

Structure of offloaded data

The following tables are used in external databases to store metrics and events offloaded from ADB Control. Note that the conditions for offloading data to the listed tables are different:

- `audit_auth`, `audit_auth_adb`, `audit_operation`, `relation_statistic` — it is enough to configure `Export job` on the Configuration → Job policy → Metrics offload tab.
- `cluster`, `cluster_query_metric`, `query`, `query_error`, `transaction` — besides configuring `Export job` on the Configuration → Job policy → Metrics offload tab, it is necessary to specify the databases which metrics should be exported. You should do it on the Configuration → Clusters tab.
- `session` — besides configuring `Export job` on the Configuration → Job policy → Metrics offload tab (with the ***Export sessions*** checkbox selected), it is necessary to set the ***Sessions*** flag on the Configuration → General → Sessions tab.

audit_auth

This table stores user login attempts to ADB Control, as well as failed ADB authorizations. For more information on the below mentioned parameters, see the ***Authorizations*** section of the Audit article.

Column	Description
id	A unique identifier of the authorization event
time	An event timestamp
username	A user name
session_time	A start timestamp of the session within which the current event took place
session	An identifier of the session within which the current event took place
host	IP address of the host from which the event was initiated
auth_type	An authorization type: - `ADCC` — ADB Control login attempt via basic authentication; - `ADCC LDAP` — ADB Control login attempt via LDAP authentication; - `ADB` — failed ADB login attempt; - `ADB LDAP` — failed ADB login attempt via LDAP authentication.
type	An event type. Possible values: - `Auth` - `Logout` - `Session expired`
result	An event result. Possible values: - `Success` - `Failed`
message	An error message

audit_auth_adb

This table contains additional information about failed login attempts to ADB.

Column	Description
auth_id	An identifier of the authorization event. Refers to `audit_auth.id` (see audit_auth)
pid	An identifier of the process within which the current event took place
database	A name of the database to which the connection was made
port	The port number that was used for the current connection
segment	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables → gp_segment_configuration*** section of the Tables overview article
transaction	A transaction identifier
cmd_count	A command identifier
cluster	A name of the ADB cluster to which the connection was made

audit_operation

This table contains information about user operations in ADB Control. For more information on the below mentioned parameters, see the ***Operations*** section of the Audit article.

Column	Description
id	A unique operation identifier
object_name	A name of the object on which the operation was performed
object_type	A type of the object on which the operation was performed. Possible values: - `User` — user; - `User forbidden DB` — assigning advanced permissions to a specific database (actual for users with the `Advanced User` role); - `LDAP mapping` — role mapping with LDAP group; - `Transaction` — transaction; - `Cluster` — cluster; - `Cluster export databases` — databases, whose metrics will be offloaded; - `Job` — ADB Control job; - `System settings` — system configuration parameters; - `System user` — system user; - `Session` — ADB session; - `User session` — ADB Control user session; - `Cluster guc` — cluster GUC; - `Cluster databases` — databases that should be monitored in ADB Control.
operation_type	A type of the operation that was applied to the `object_name` object. Possible values: - `Cancel` - `Create` - `Delete` - `Resource group change` - `Terminate` - `Update` - `Archive` - `Activate`
operation_result	An operation result. Possible values: - `Success` - `Failed` - `Denied`
operation_time	An operation end timestamp
username	A name of the user who performed the operation
session	An identifier of the ADB Control user session
hostname	IP address of the host from which the operation was initiated
diffs	Information about changed attributes of the `object_name` object in the JSON format. For each changed attribute, the following data is stored: - `name` — an attribute name; - `oldValue` — a previous value; - `newValue` — a new value. The example:

```
[
    {
        "name":"CLUSTER_ARCHIVED",
        "oldValue":true,
        "newValue":false
    }
]
```

cluster

This table contains information on ADB clusters selected in ADB Control for export of metrics and events to the external database.

Column	Description
id	A cluster unique identifier
name	A cluster name

cluster_query_metric

This table stores statistics of system resource consumption by commands at the cluster level. For more information, see the ***System metrics collected for commands*** section of the Monitoring of commands article.

Column	Description
query_id	A command identifier. Refers to `query.id` (see query)
cpu_usage_total	The total CPU consumption during the command execution (in seconds)
cpu_usage_percent_sum	The CPU amount consumed at the end of the command execution (percentage)
cpu_usage_percent_avg	The average amount of CPU consumed across cluster segments at the end of the command execution (percentage)
cpu_usage_percent_skew	The indicator that displays the CPU consumption skew across cluster segments at the end of the command execution (percentage)
mem_sum	The RAM amount consumed at the end of the command execution (in bytes)
mem_max	The maximum amount of RAM consumed at the end of the command execution (in bytes)
mem_avg	The average amount of RAM consumed across cluster segments at the end of the command execution (in bytes)
mem_skew	The indicator that displays the RAM consumption skew across cluster segments at the end of the command execution (percentage)
vmem_sum	The virtual memory amount allocated at the end of the command execution (in bytes)
vmem_max	The maximum amount of virtual memory allocated at the end of the command execution (in bytes)
vmem_avg	The average amount of virtual memory allocated across cluster segments at the end of the command execution (in bytes)
vmem_skew	The indicator that displays the virtual memory consumption skew across cluster segments at the end of the command execution (percentage)
spill	The amount of spill files on cluster segment hosts (excluding coordinator) at the end of the command execution (in bytes)
spill_hosts	The amount of spill files on cluster hosts (including coordinator) at the end of the command execution (in bytes)
spill_max	The maximum amount of spill files produced on cluster segments at the end of the command execution (in bytes)
spill_skew	The indicator that displays the spill file amount skew at the end of the command execution (percentage)
read_bytes_total	The total amount of data read during the command execution (in bytes)
read_bytes_sum	The amount of data read at the end of the command execution (in bytes)
read_bytes_max	The maximum amount of data read at the end of the command execution (in bytes)
read_bytes_avg	The average amount of data read across cluster segments per second at the end of the command execution (in bytes)
read_bytes_skew	The indicator that displays the data reading skew across cluster segments at the end of the command execution (percentage)
write_bytes_total	The total amount of data written during the command execution (in bytes)
write_bytes_sum	The amount of data written at the end of the command execution (in bytes)
write_bytes_max	The maximum amount of data written at the end of the command execution (in bytes)
write_bytes_avg	The average amount of data written across cluster segments per second at the end of the command execution (in bytes)
write_bytes_skew	The indicator that displays the data writing skew across cluster segments at the end of the command execution (percentage)

query

This table contains information on completed SQL commands that were registered in ADB Control.

Column	Description
query_id	A unique command identifier
status	A command status. Possible values: - `Done` — successfully completed commands. - `Cancelled` — cancelled commands (as a part of the transaction cancellation or termination). - `Error` — the commands during which some errors occurred. - `Unknown` — the commands, the final status of which is undefined. It is set on a timeout for queries in the `Queued`, `Running`, `Cancelling` statuses, for which there are no corresponding rows in the `pg_stat_activity` view.
pid	An identifier of the process, within which the command was started
submitted	The timestamp when the user submitted the command
run_at	A command start timestamp
finish_at	A command end timestamp
username	A name of the user who started the command
database	A name of the database where the command was started
resource_group	A name of the resource group that was used for the command
planner	A name of the query optimizer that was used to produce the query execution plan. Possible value: - `GPORCA` — GPORCA is used (`optimizer = on`). - `Fallbacked` — after the attempt to produce a plan via GPORCA (`optimizer = on`) failed, Postgres query optimizer is used. - `Legacy` — Postgres query optimizer is used (`optimizer = off`).
querytext	A text of the SQL query
plantext	A text of the command execution plan
session_id	An identifier of the session, within which the command was started
cluster_id	An identifier of the ADB cluster where the command was started. Refers to `cluster.id` (see cluster)
queued_time	The total time that the command was queued (in hours, minutes, seconds)
run_time	The total command duration (in hours, minutes, seconds)
sql_id	A common identifier for SQL commands with the same structure
tags	The query tags

query_error

This table contains information about errors that occurred during execution of SQL commands.

Column	Description
id	A unique identifier of the error record
query_id	A command identifier. Refers to `query.id` (see query)
seg_id	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables → gp_segment_configuration*** section of the Tables overview article
slice_id	An identifier of the slice during which execution the current error occurred
message	An error message text

session

This table contains information on sessions that were registered in ADB Control.

Column	Description
cluster_id	An identifier of the ADB cluster. Refers to `cluster.id` (see cluster)
datetime	The timestamp when the data was received
datid	A database OID
datname	A database name
pid	A backend process identifier
sess_id	A session identifier
usesysid	An OID of the user who connected to the backend
usename	A name of the user who connected to the backend
application_name	A name of the application that is connected to the backend
client_addr	A client IP address. The `null` value indicates the use of a Unix socket on the server or the execution of an internal process (e.g. `autovacuum`)
client_hostname	A host name of the connected client, as reported by a reverse DNS lookup of the `client_addr` value. The field is not `null` only for IP connections, and only when the server configuration parameter (GUC) log_hostname is set to `on`
client_port	The TCP port number that is used by the client to communicate with the backend, or `-1` if a Unix socket is used
backend_start	The time when the backend process was started
xact_start	A transaction start timestamp
query_start	A command execution start timestamp
state_change	The last time the value of the `state` field was changed
waiting	`true` in case of waiting on a lock, otherwise `false`
state	The current session state. Possible values: - `active` — the backend is running a query. - `idle` — the backend is waiting for a new client command. - `idle in transaction` — the backend is in a transaction, but there are no currently running queries.
query	A text of the most recent SQL query that was run at the backend. If `state` is `active`, the field shows the currently running query
rsgid	A resource group OID or `0` if definition is not possible (for more information, see pg_stat_activity in the Greenplum documentation)
rsgname	A resource group name or `unknown` if definition is not possible (for more information, see pg_stat_activity in the Greenplum documentation)
rsgqueueduration	The total time the query has been queued (if applicable)

transaction

This table contains information on completed transactions that were registered in ADB Control.

Column	Description
txn_id	A unique transaction identifier in ADB Control
status	A transaction status. Possible values: - `Committed` — successfully completed transactions. - `Aborted` — cancelled/terminated transactions. - `Unknown` — the transactions, the final status of which is undefined. It is set on a timeout for transactions in the `Active` status, for which there are no corresponding rows in the `pg_stat_activity` view.
pid	An identifier of the process, within which the transaction was started
start_time	A transaction start timestamp
end_time	A transaction end timestamp
username	A name of the user who started the transaction
database	A name of the database where the transaction was started
resource_group	A name of the resource group that was used for the transaction
session_id	An identifier of the session, within which the transaction was started
cluster_id	An identifier of the ADB cluster where the transaction was started. Refers to `cluster.id` (see cluster)
tags	The transaction tags

Cluster management

Daria Barysheva

Contents

To Table of Contents

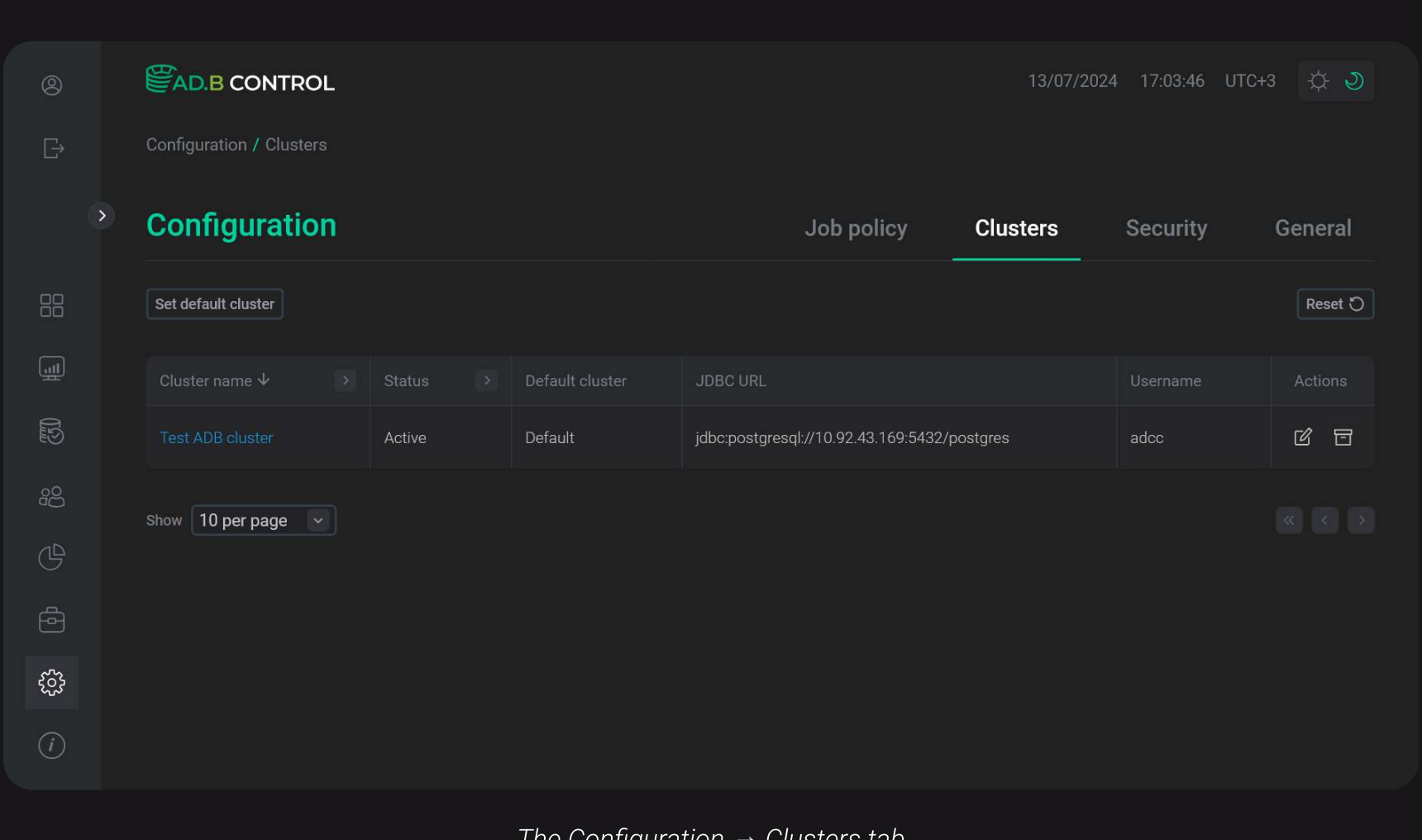
- View a list of clusters
- Add a cluster
- Edit a cluster connection
- Set a default cluster
- Archive a cluster
- Configure monitoring of the selected cluster
  - Databases monitoring
  - GUC management
  - Metric offload

ADB Control allows you to monitor multiple ADB clusters simultaneously. In this case, the target cluster (within which the ADB Control web interface will be used) and other clusters (which will send metrics to the first cluster) do not have to use the same ADCM.

To manage clusters, you can use the ***Clusters*** tab on the ***Configuration*** page in the ADB Control web interface.

View a list of clusters

The ***Configuration*** → ***Clusters*** tab displays the following information on clusters.


AD.B CONTROL
13/07/2024 17:03:46 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

Cluster name	Status	Default cluster	JDBC URL	Username	Actions
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

The Configuration → Clusters tab

The "Configuration → Clusters" tab fields

Field	Description
Cluster name	A cluster name
Status	A cluster status: ▪ `Active` — the cluster is currently monitored. ▪ `Archived` — the cluster monitoring is suspended.
Default cluster	For a default cluster, the `Default` label is shown. Otherwise, `No`
JDBC URL	A URL of the JDBC connection
Username	A user name that is used to connect to a cluster

In the column headers of the table with a list of clusters, there are filters that you can use to select specific data. To open a filter, click the › icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Cluster name***), the search value should be entered.

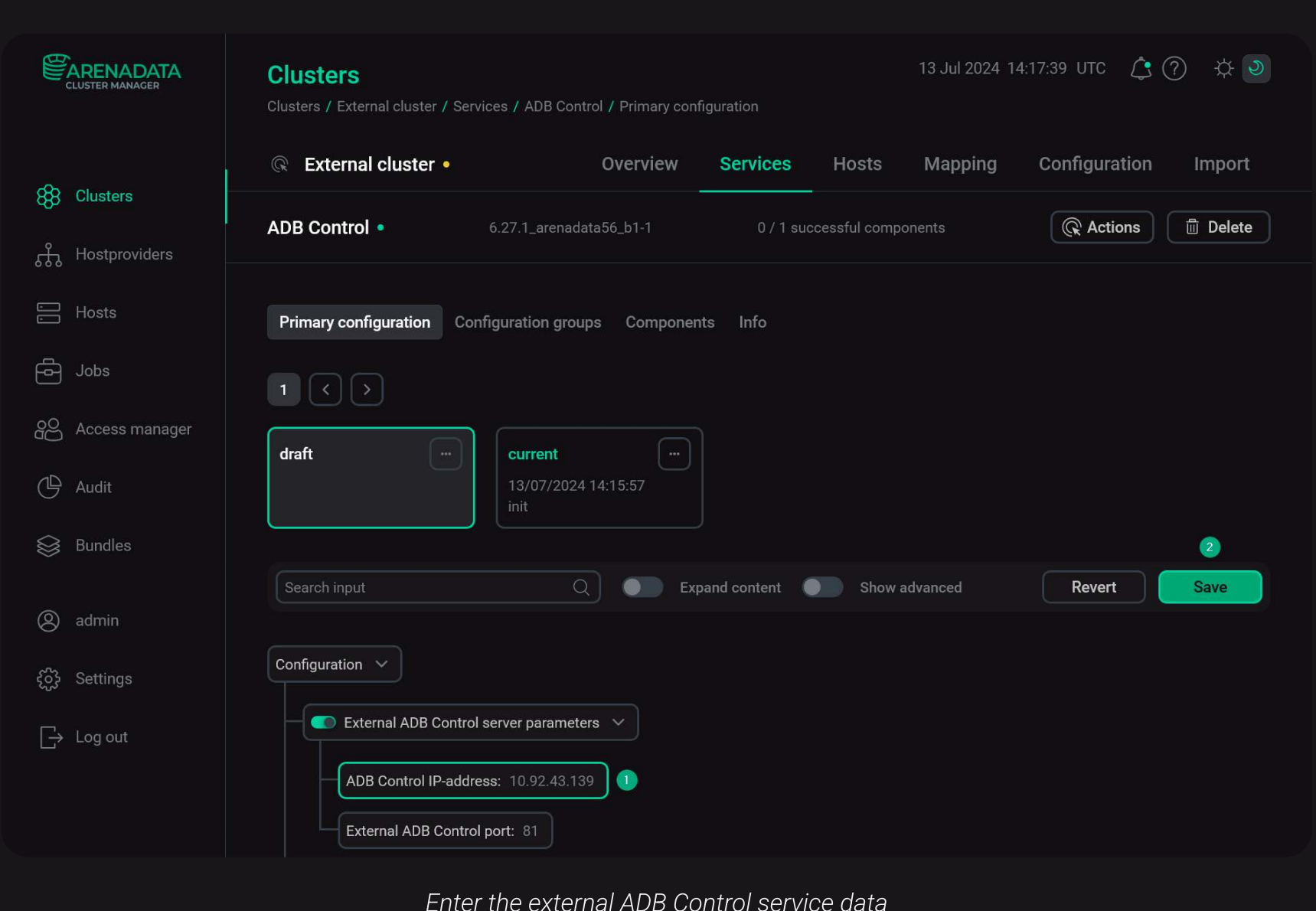
The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Add a cluster

> **IMPORTANT**
>
> - A cluster name should be unique within ADB Control. You cannot add several clusters with the same name (for example, created via different ADCM instances).
> - When upgrading clusters via ADCM, first, update the ADB cluster where the main ADB Control service is installed (not external).

To add a new ADB cluster to be monitored in ADB Control, follow the steps:

1. Using ADCM, add the ADB Control service to the cluster that should be added to the existing monitoring system.
2. When configuring the service, do the following:
   - Activate the ***External ADB Control server parameters*** switcher.
   - In the ***ADB Control IP-address*** field, enter the IP address of the host where the target ADB Control service is deployed. Its web interface is planned to be used for monitoring the new cluster.
   - In the ***External ADB Control port*** field, enter the number of the port where the target ADB Control service is deployed. The default value is `81`.
   - Click ***Save***.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / External cluster / Services / ADB Control / Primary configuration
13 Jul 2024 14:17:39 UTC
External cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADB Control
6.27.1_arenadata56_b1-1
0 / 1 successful components
Actions
Delete
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Primary configuration
Configuration groups
Components
Info
draft
current
13/07/2024 14:15:57
init
Search input
Expand content
Show advanced
Revert
Save
Configuration

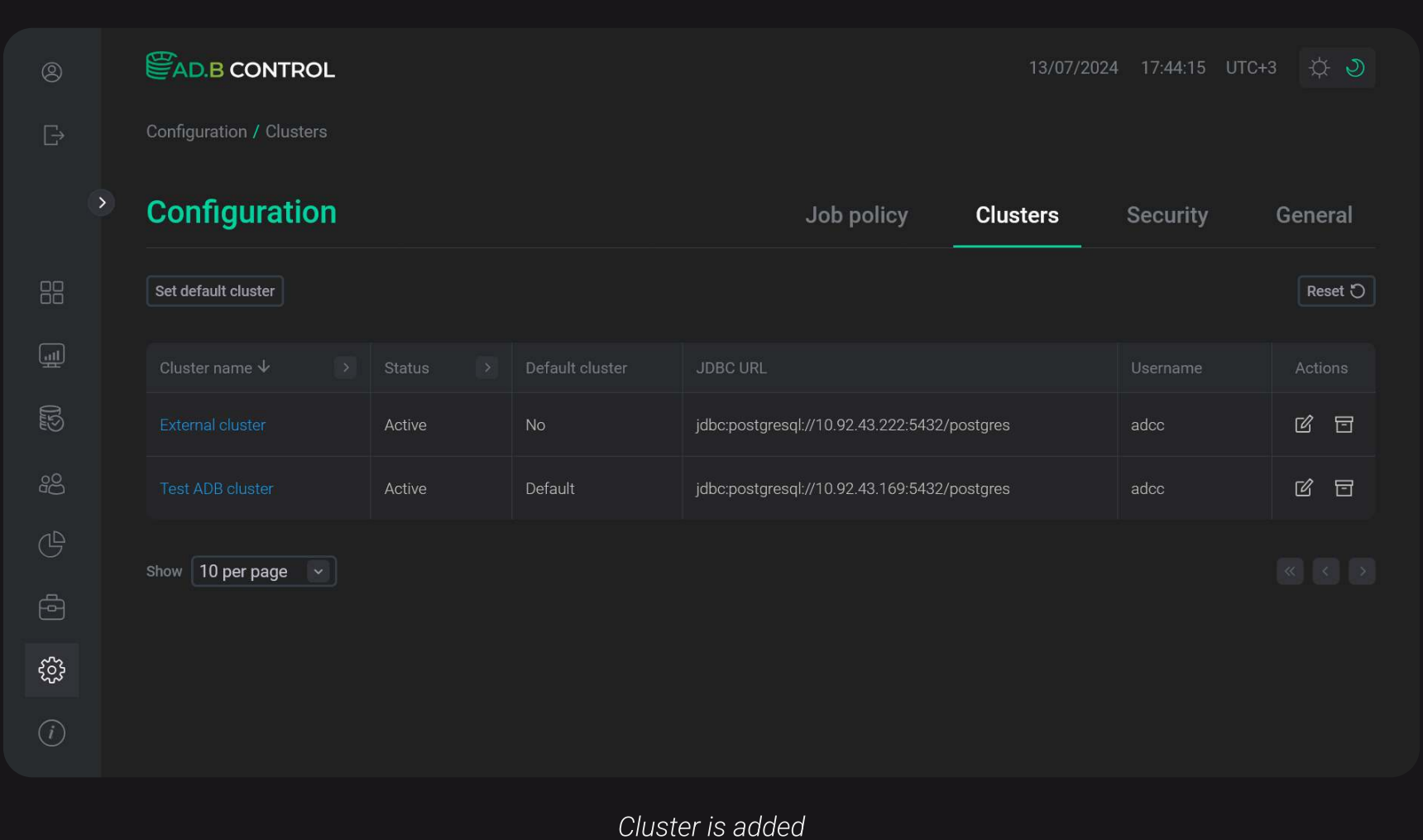
External ADB Control server parameters

ADB Control IP-address: 10.92.43.139

External ADB Control port: 81

Enter the external ADB Control service data

3. Install the service. Note that you should not distribute service components between cluster hosts since the external ADB Control is to be used.
4. In the web interface of the target ADB Control (which will be used for monitoring), open the ***Configuration*** → ***Clusters*** tab. The new cluster should be added automatically.

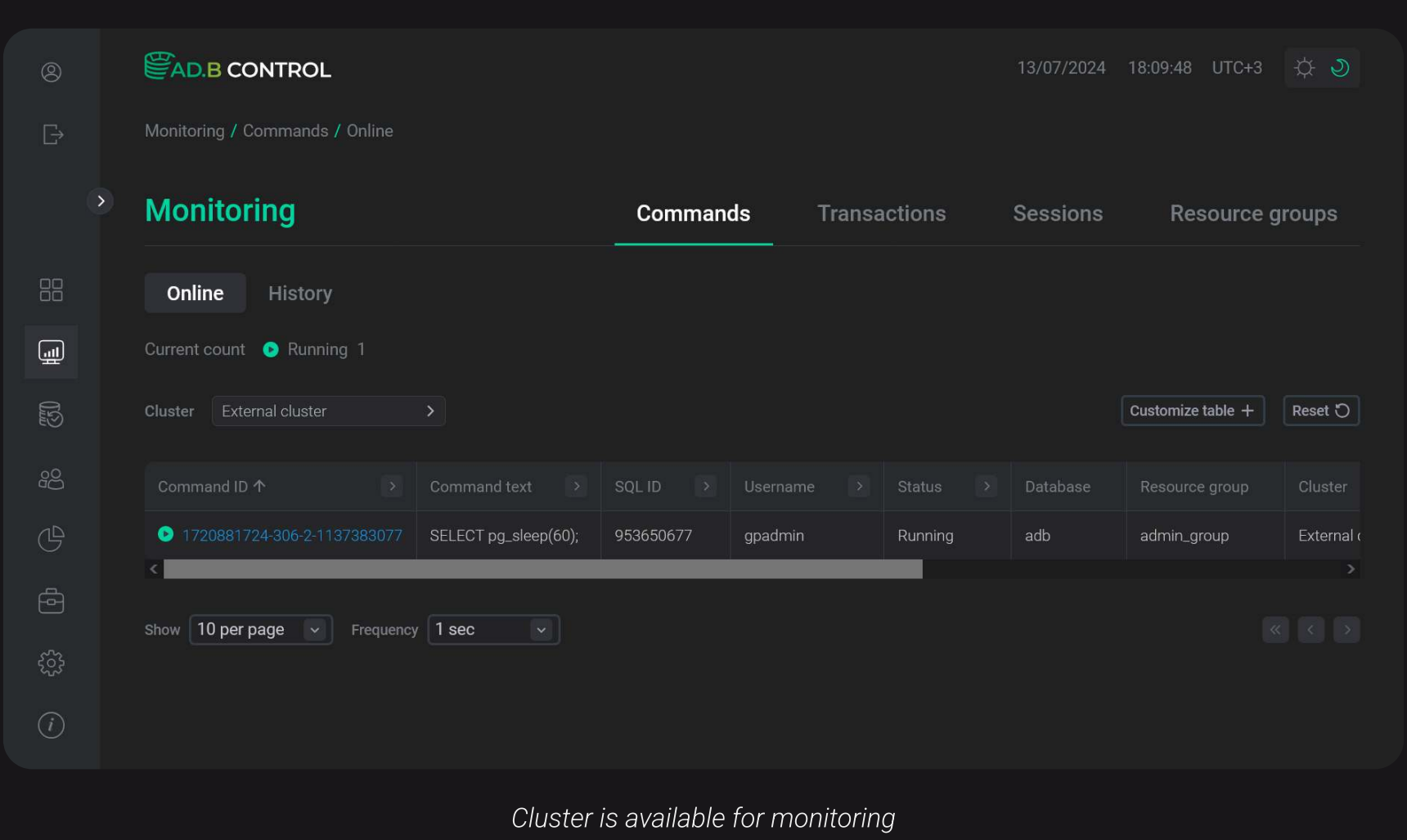

AD.B CONTROL
13/07/2024 17:44:15 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Active	No	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Cluster is added

5. To ensure that the cluster connection is successful, you can run an SQL query in one of the cluster databases that were previously selected for monitoring, and verify that the query information appears on the ***Monitoring*** page. Note that the added cluster must be pre-selected in the filters on the page (if the cluster is not marked as default). You can use a test query as follows:

```
SELECT pg_sleep(60);
```


AD.B CONTROL
13/07/2024 18:09:48 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 1
Customize table
Reset
Show 10 per page
Frequency 1 sec

Cluster: External cluster

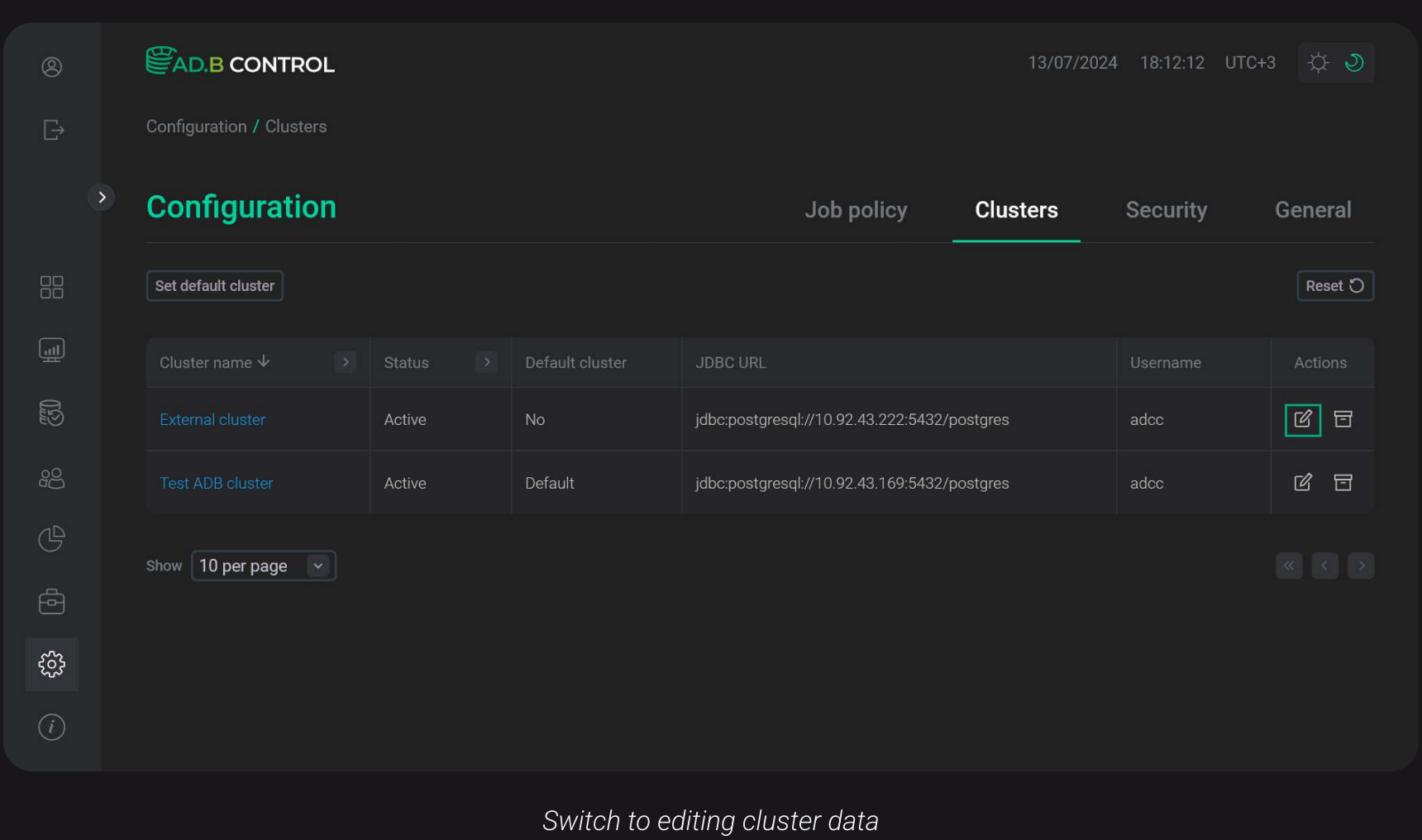
Command ID	Command text	SQL ID	Username	Status	Database	Resource group	Cluster
1720881724-306-2-1137383077	SELECT pg_sleep(60);	953650677	gpadmin	Running	adb	admin_group	External

Cluster is available for monitoring

Edit a cluster connection

To edit cluster data, follow the steps:

1. Click the ✎ icon in the ***Actions*** column on the ***Configuration*** → ***Clusters*** tab.


AD.B CONTROL
13/07/2024 18:12:12 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

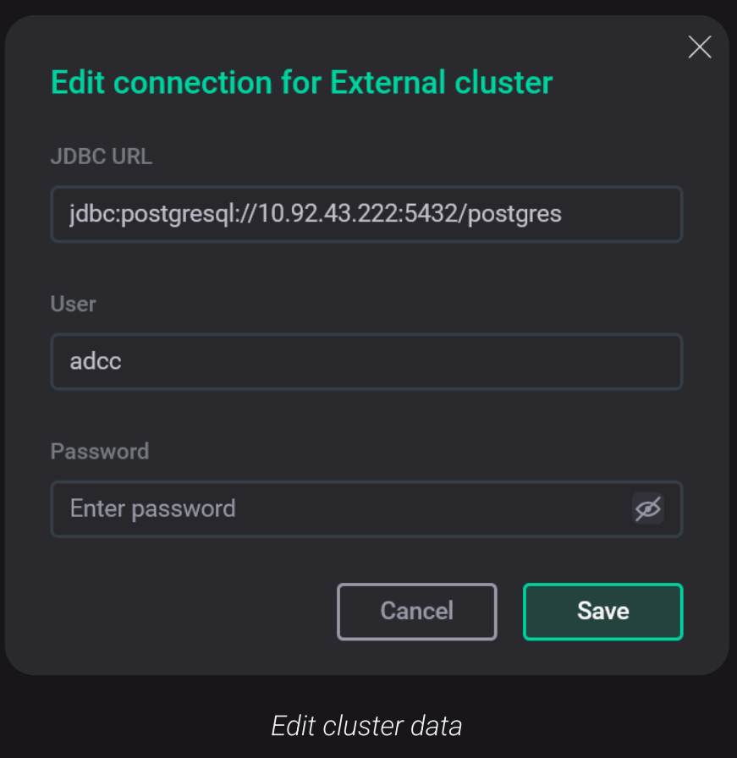
Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Active	No	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Switch to editing cluster data

2. In the window that opens, edit necessary fields (the ***Cluster name*** field value cannot be changed):
   - ***JDBC URL*** — a URL of the JDBC connection to the coordinator host of the ADB cluster. Use the following format:

```
jdbc:postgresql://<ADB coordinator host IP address>:5432/postgres
```

   - ***User*** — a user name. Use `adcc` by default.
   - ***Password*** — a user password. The default user `adcc` does not require a password.


Cancel
Save

Edit connection for External cluster

JDBC URL: jdbc:postgresql://10.92.43.222:5432/postgres

User: adcc

Password: Enter password

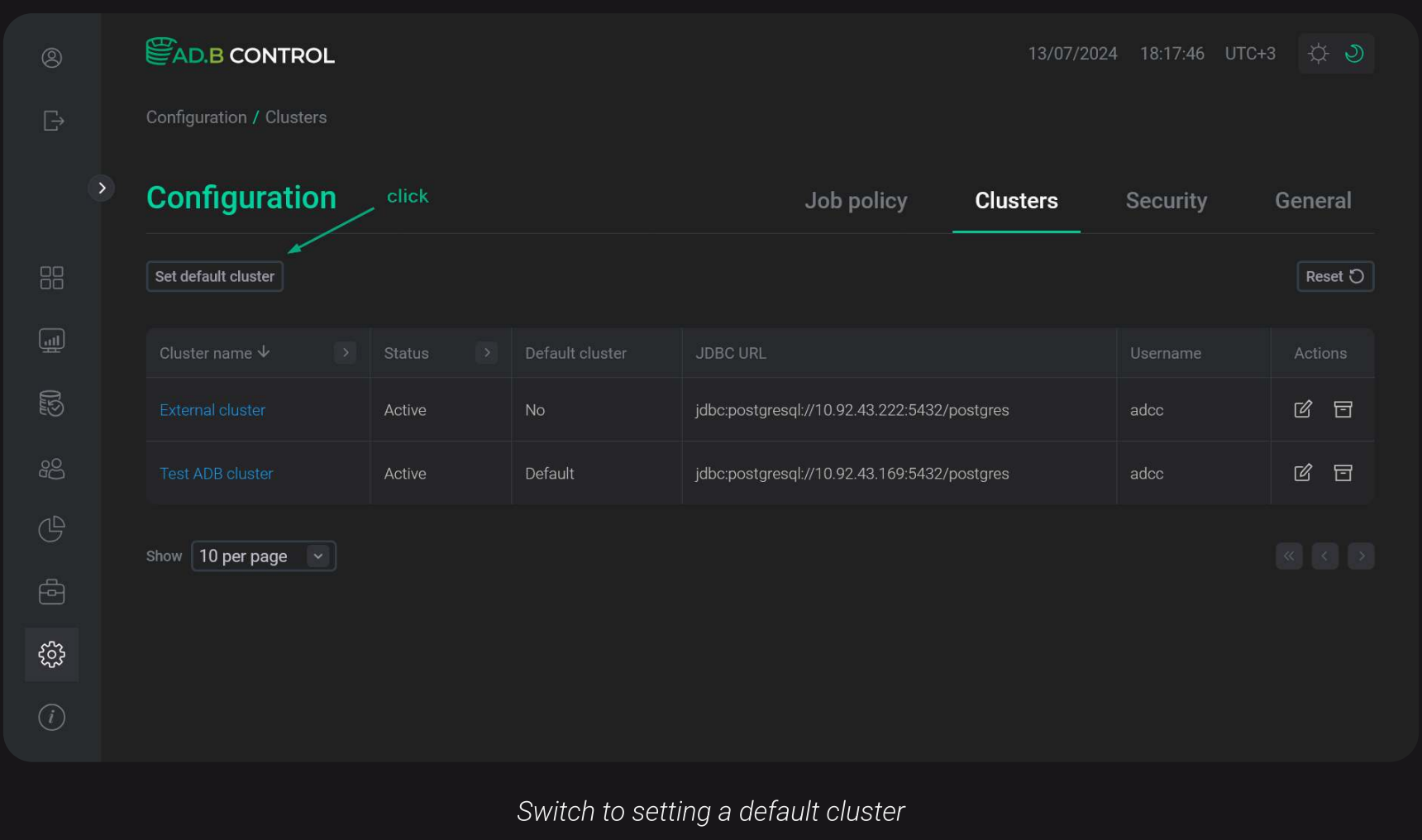
Edit cluster data

3. Click ***Save*** to apply changes.

Set a default cluster

Many pages of the ADB Control web interface have cluster name filters. To assign the cluster to be selected in these filters by default, follow the steps:

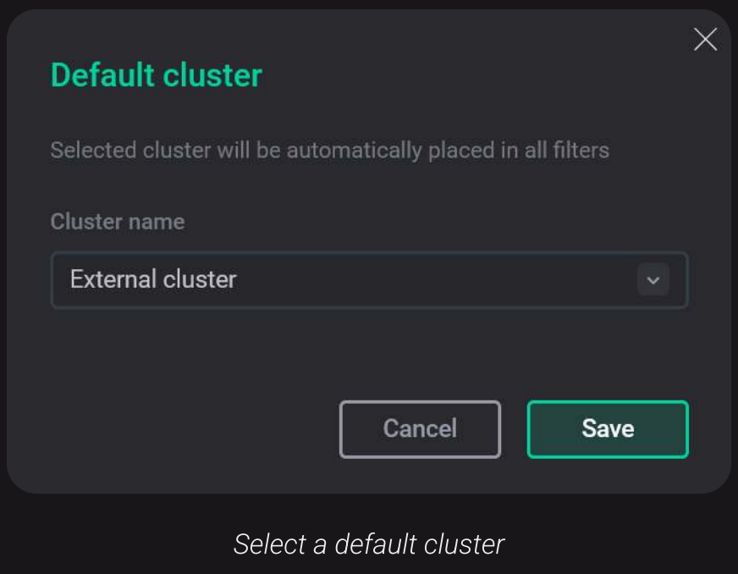
1. Click ***Set default cluster*** on the ***Configuration*** → ***Clusters*** tab.


AD.B CONTROL
13/07/2024 18:17:46 UTC+3
Configuration / Clusters
Configuration
click
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Active	No	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Switch to setting a default cluster

2. In the window that opens, select a default cluster from the ***Cluster name*** drop-down list.


Cancel
Save

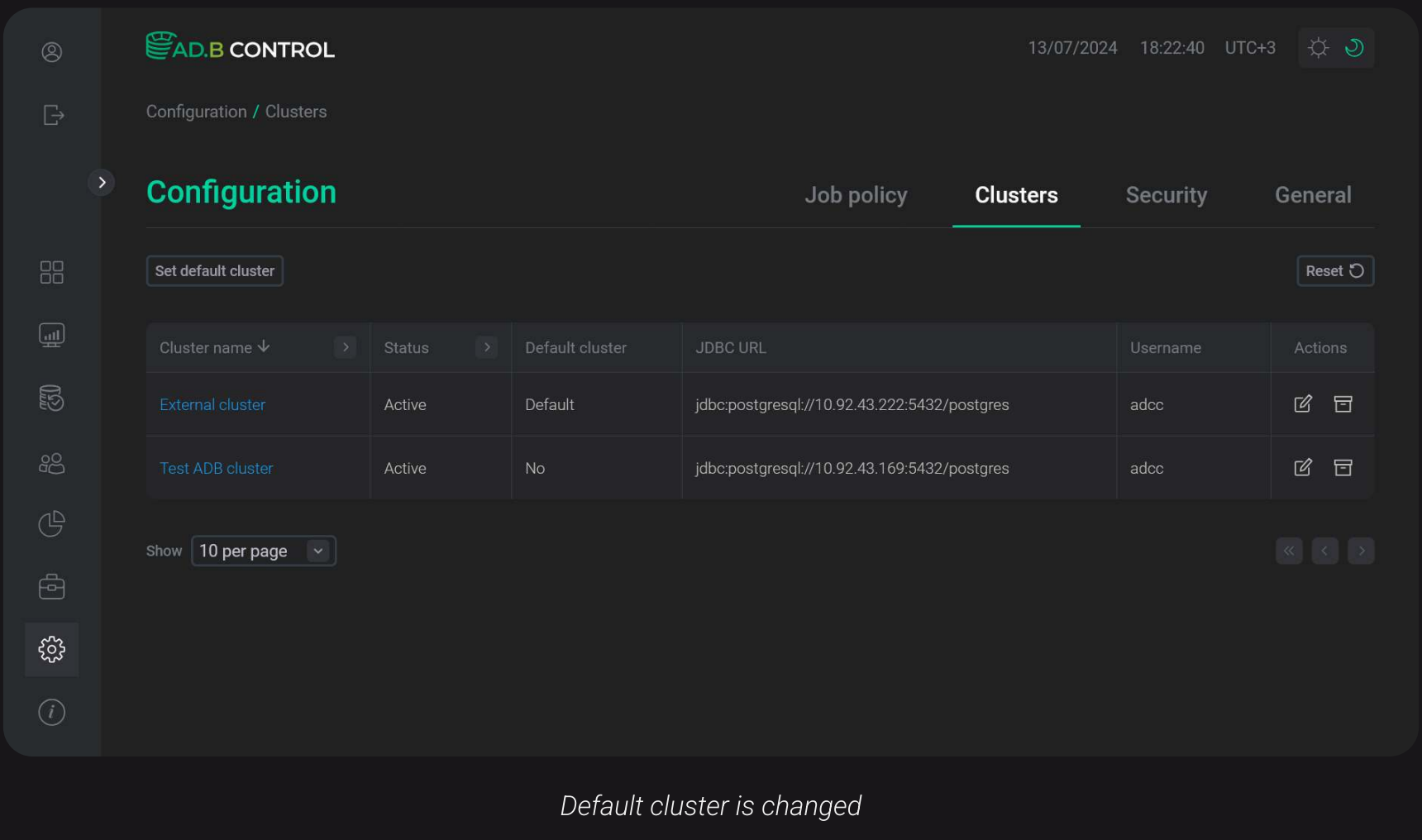
Default cluster

Selected cluster will be automatically placed in all filters

Cluster name: External cluster

Select a default cluster

3. Click ***Save***. As a result, the ***Default cluster*** column in the table of clusters displays `Default` for a new default cluster and `No` — for a previous default cluster.


AD.B CONTROL
13/07/2024 18:22:40 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Active	Default	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	No	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Default cluster is changed

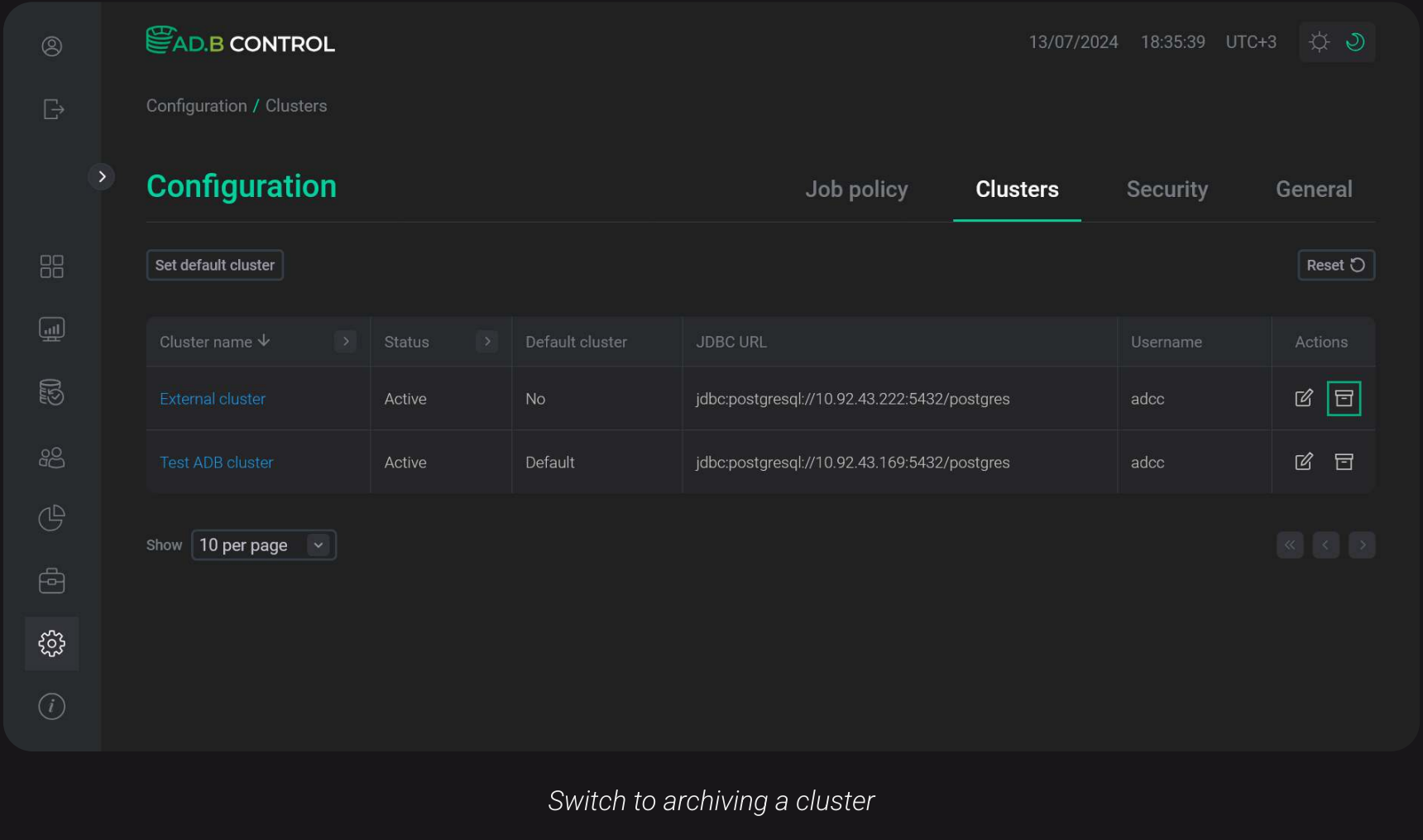
Archive a cluster

In some cases, it is necessary to suspend the processing of monitoring metrics in ADB Control for a specific cluster. For this purpose, you can **archive** a cluster, which means:

- ADB Control does not collect audit events for the cluster.
- The export job skips the data related to the archived cluster.
- Resource group statistics is not collected for the cluster.
- On the ***Information*** page, there are no cluster details.

To archive a cluster, follow the steps:

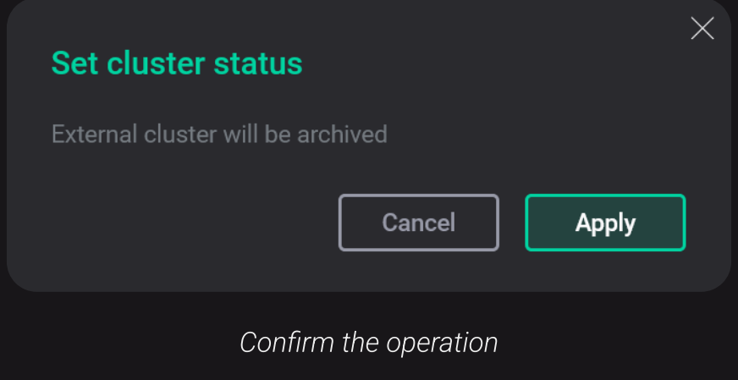
1. Click the 🗄 icon in the ***Actions*** column on the ***Configuration*** → ***Clusters*** tab.


AD.B CONTROL
13/07/2024 18:35:39 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Active	No	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Switch to archiving a cluster

2. In the window that opens, confirm the operation by clicking ***Apply***.

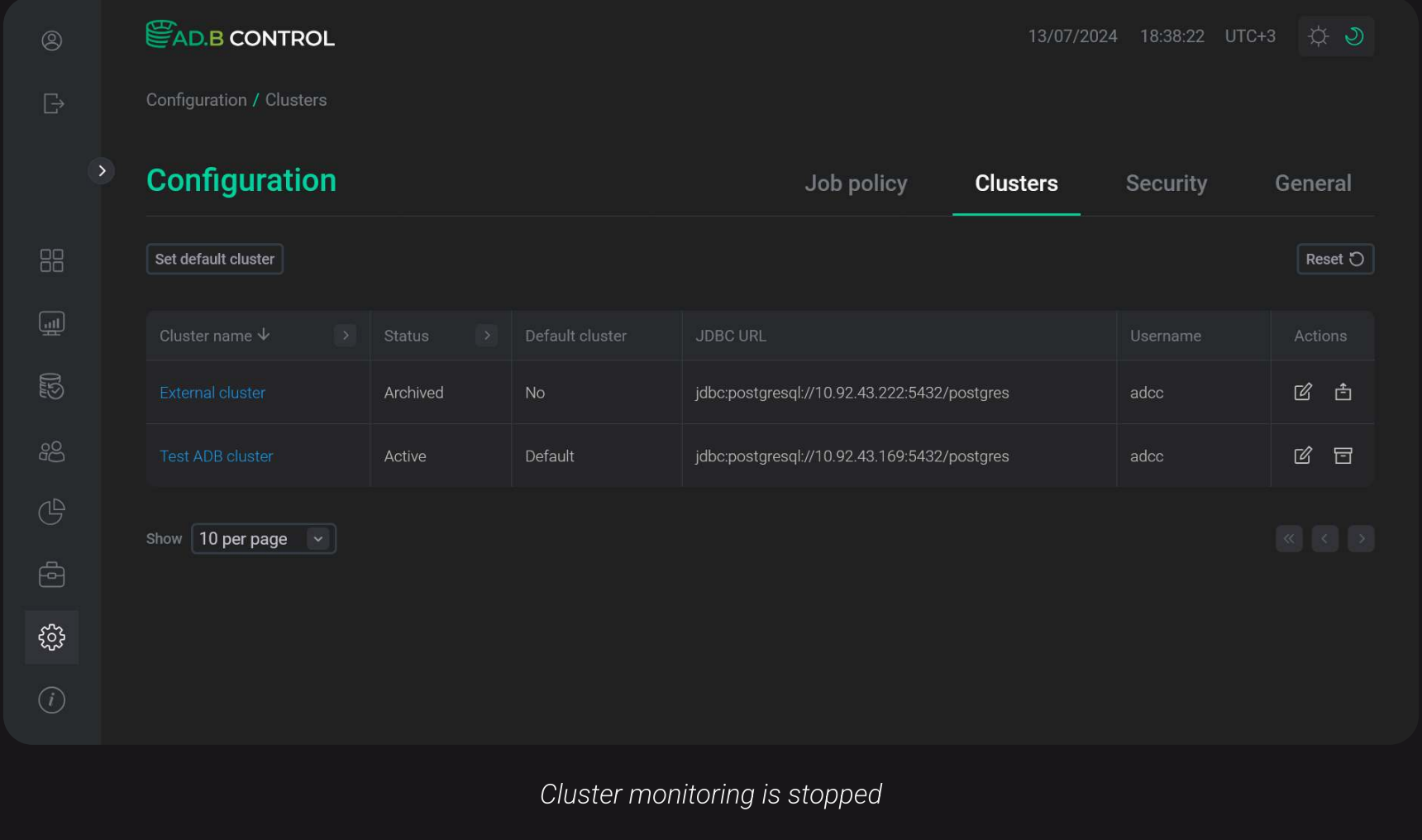

Cancel
Apply

Set cluster status

External cluster will be archived

Confirm the operation

As a result, the cluster status is changed to `Archived` on the ***Configuration*** → ***Clusters*** tab.


AD.B CONTROL
13/07/2024 18:38:22 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
Show 10 per page

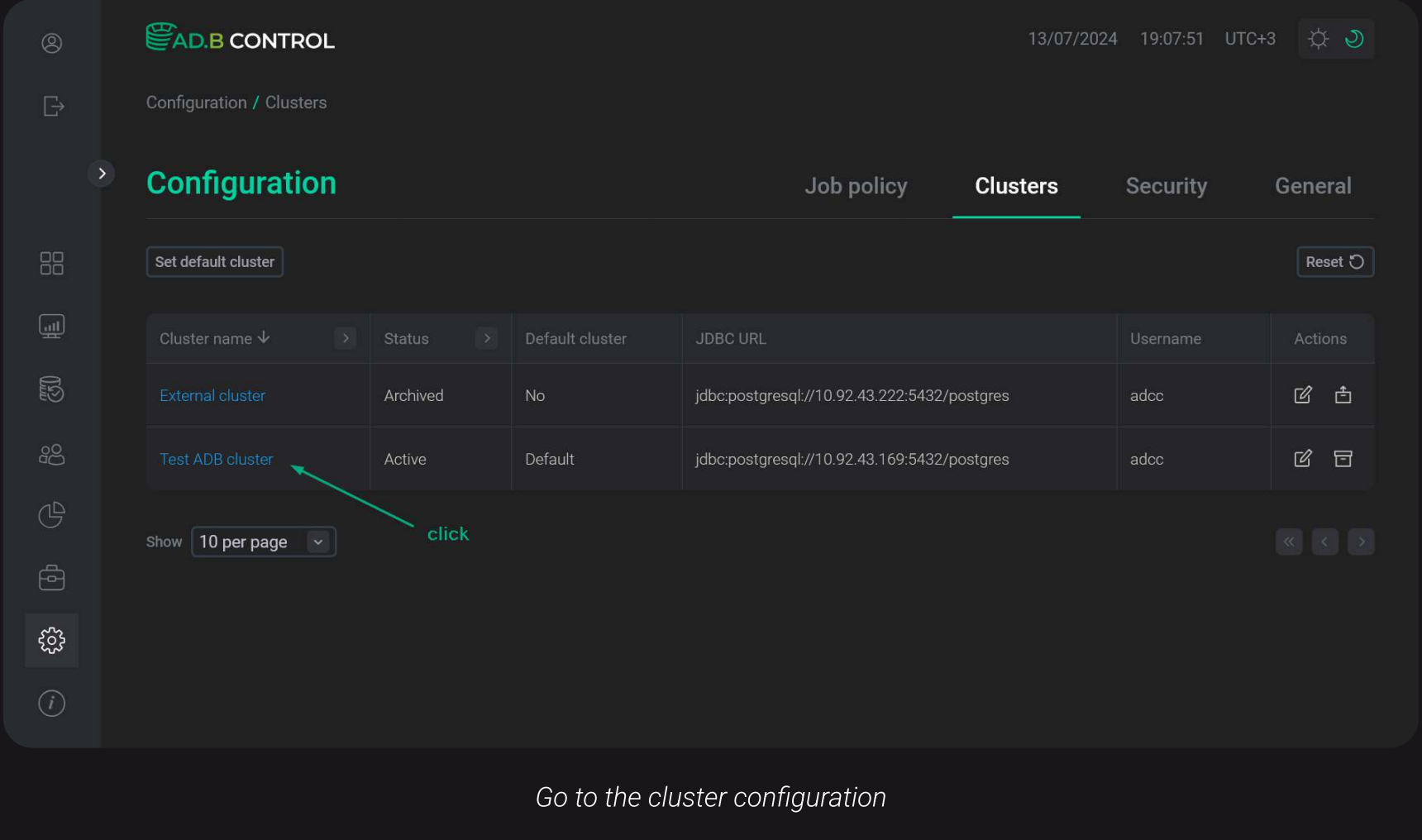
Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Archived	No	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Cluster monitoring is stopped

> **NOTE**
>
> If you later need to resume the cluster monitoring, click the ⤒ icon in the ***Actions*** column for that cluster on the ***Configuration*** → ***Clusters*** tab and confirm the action in the window that opens.

Configure monitoring of the selected cluster

To navigate to the cluster monitoring configuration, click a cluster name on the ***Configuration*** → ***Clusters*** tab.

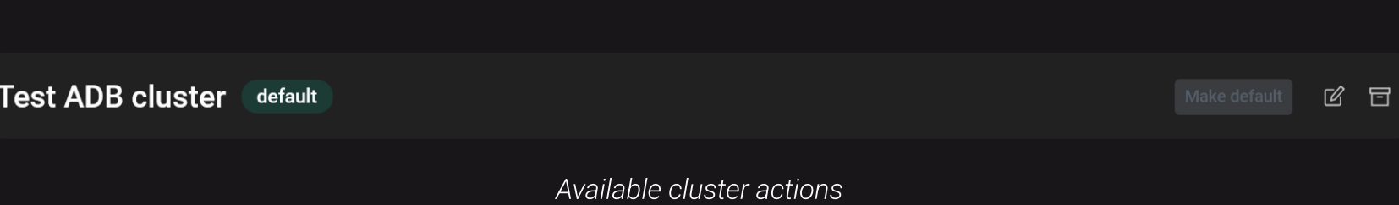

AD.B CONTROL
13/07/2024 19:07:51 UTC+3
Configuration / Clusters
Configuration
Job policy
Clusters
Security
General
Set default cluster
Reset
click
Show 10 per page

Cluster name	Status	Default cluster	JDBC URL	Username	Actions
External cluster	Archived	No	jdbc:postgresql://10.92.43.222:5432/postgres	adcc	
Test ADB cluster	Active	Default	jdbc:postgresql://10.92.43.169:5432/postgres	adcc	

Go to the cluster configuration

The top of the opened page displays a cluster name, the `default` label (if a cluster is selected by default), and a number of control elements for a quick transition to the actions described above:

- ***Make default*** — make a cluster default.
- 🗄 — archive a cluster.
- ⤒ — activate an archived cluster.
- ✎ — edit a cluster connection.

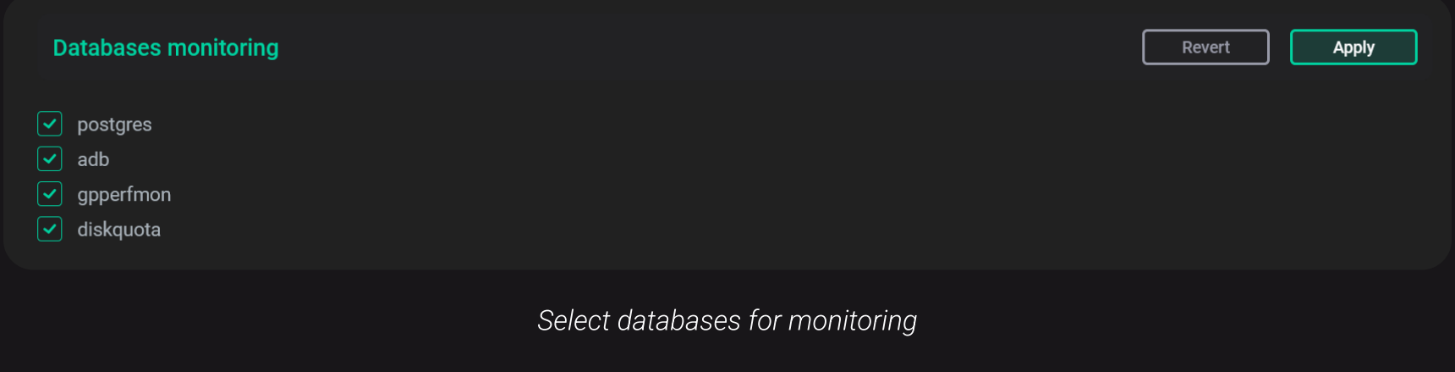

Test ADB cluster
default
Make default

Available cluster actions

The main part of the opened page contains several sections that are described below. To apply any changes to the parameters in these sections, click ***Apply*** in the appropriate section, to undo the changes that have not been yet saved — click ***Revert***.

Databases monitoring

In the ***Databases monitoring*** section on the cluster page, you can select databases that should be monitored in ADB Control. By default, all databases are selected.

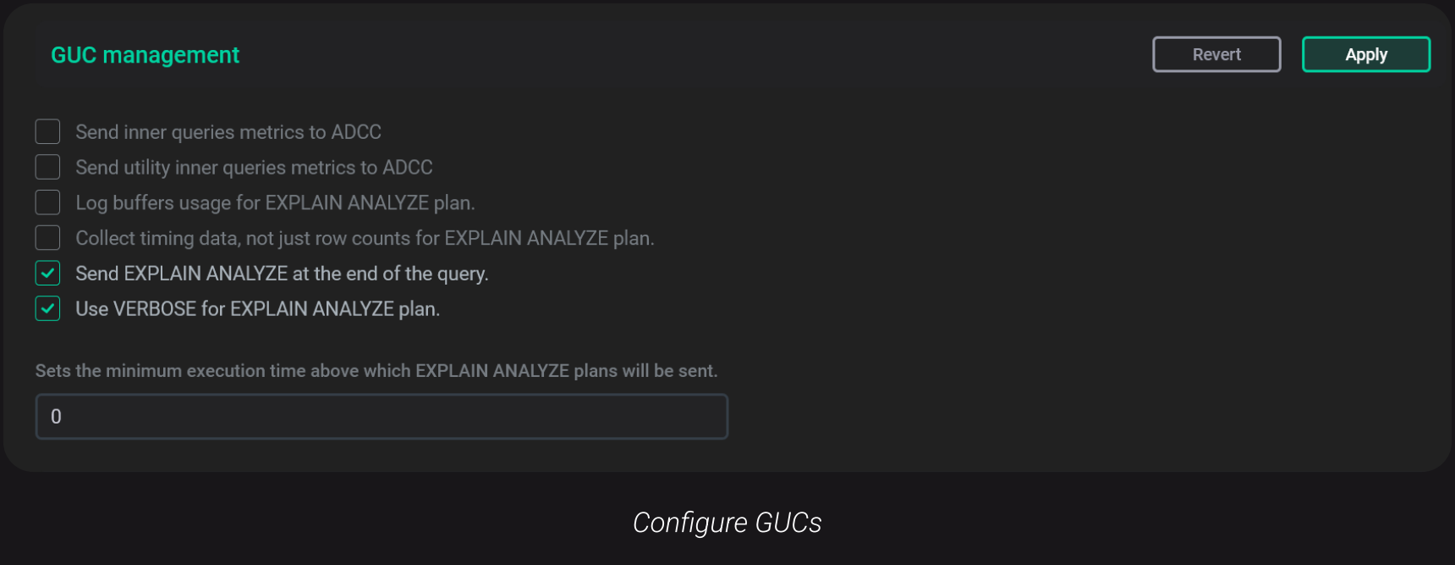

Databases monitoring
Revert
Apply

☑ postgres
☑ adb
☑ gpperfmon
☑ diskquota

Select databases for monitoring

GUC management

In the ***GUC management*** section on the cluster page, you can configure values of the GUCs that control ADB Control behavior.


GUC management
Revert
Apply

☐ Send inner queries metrics to ADCC
☐ Send utility inner queries metrics to ADCC
☐ Log buffers usage for EXPLAIN ANALYZE plan.
☐ Collect timing data, not just row counts for EXPLAIN ANALYZE plan.
☑ Send EXPLAIN ANALYZE at the end of the query.
☑ Use VERBOSE for EXPLAIN ANALYZE plan.

Sets the minimum execution time above which EXPLAIN ANALYZE plans will be sent.
0

Configure GUCs

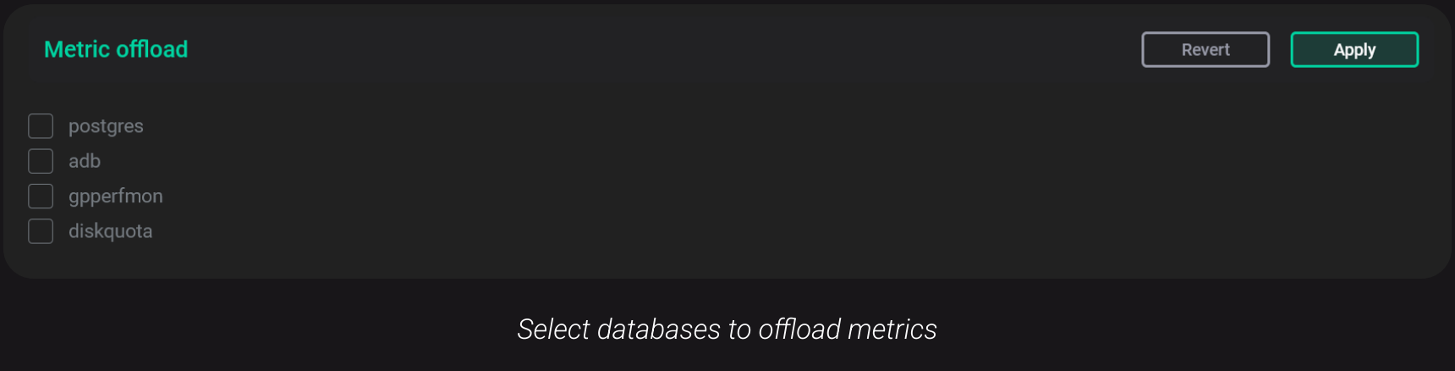
Available GUCs are shown in the table below.

GUCs that manage ADB Control behavior

GUC name	UI description
adcc.monitor_inner_queries	Send inner queries metrics to ADCC
adcc.monitor_utility_inner_queries	Send utility inner queries metrics to ADCC
adcc.explain_log_buffers	Log buffers usage for EXPLAIN ANALYZE plan
adcc.explain_log_timing	Collect timing data, not just row counts for EXPLAIN ANALYZE plan
adcc.explain_log_analyze	Send EXPLAIN ANALYZE at the end of the query
adcc.explain_log_verbose	Use VERBOSE for EXPLAIN ANALYZE plan
adcc.explain_log_min_duration	Sets the minimum execution time above which EXPLAIN ANALYZE plans will be sent (in milliseconds)

Metric offload

The ***Metric offload*** section of the cluster page allows you to select databases, whose metrics will be offloaded to the external analytical database (according to the schedule that is configured on the Configuration → Job policy → Metrics offload page). By default, no database is selected.


Metric offload
Revert
Apply

☐ postgres
☐ adb
☐ gpperfmon
☐ diskquota

Select databases to offload metrics

Configure security policies

Daria Barysheva

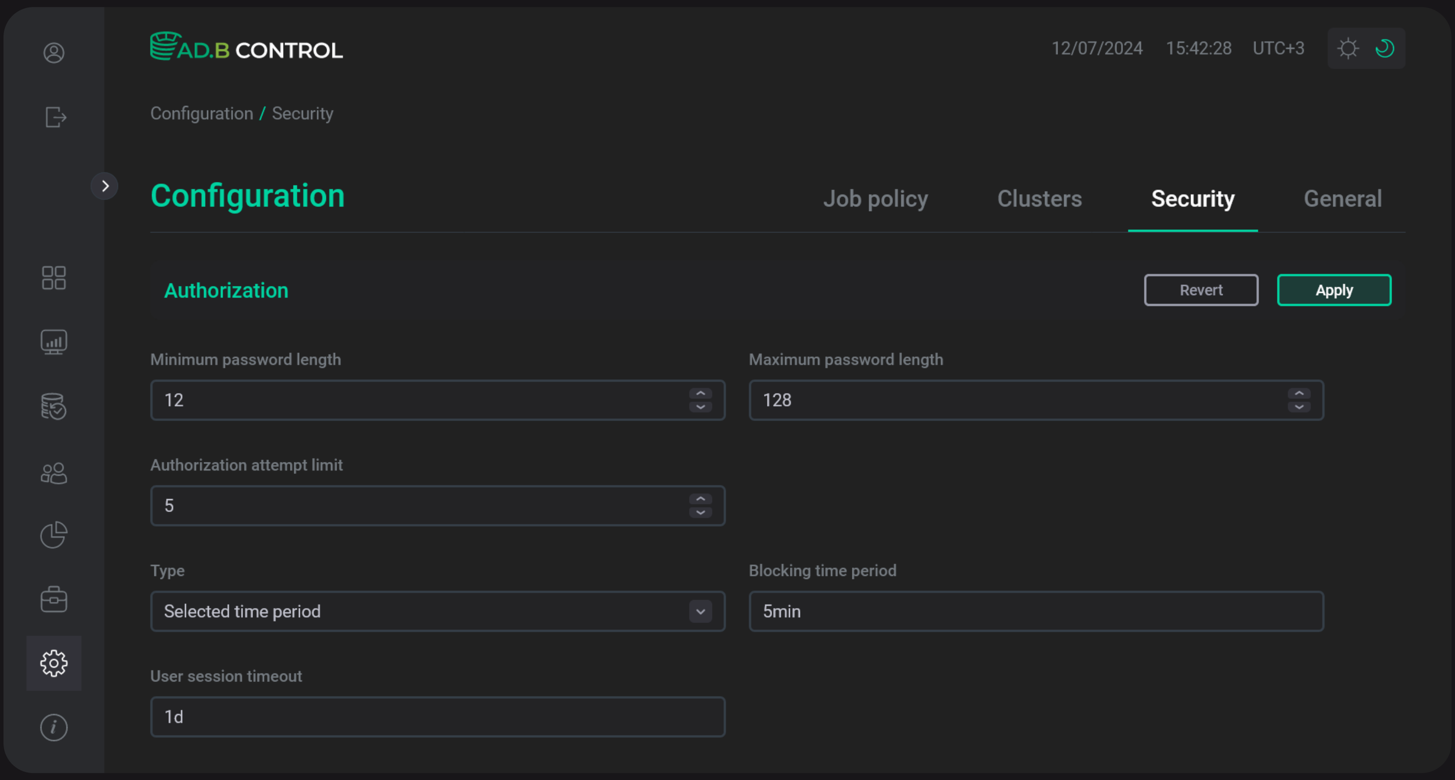
To Table of Contents

The ***Security*** tab on the ***Configuration*** page in the ADB Control web interface allows you to configure security policies. On this tab, you can define the following restrictions, which will be checked during user creation and authorization.

The "Configuration → Security" tab fields

Field	Description	Default value
Minimum password length	The minimal password length for user creation	12
Maximum password length	The maximum password length for user creation	128
Authorization attempt limit	The maximum number of failed login attempts that permanently or temporarily blocks the user (depending on the ***Type*** value)	5
Type	The blocking type that is to be used when the maximum number of failed logon attempts (***Authorization attempt limit***) is reached. Possible values: ■ `Permanently` — block permanently until the system owner performs unblocking. ■ `Selected time period` — block temporarily for the time specified in the ***Blocking time period*** field (or until the system owner performs unblocking).	Selected time period
Blocking time period	The temporary user blocking period in the `<value><unit>` format (without spaces), where `<unit>` has the following possible values: ■ `min` — minutes; ■ `hr` — hours; ■ `d` — days. Examples: `5min`, `1hr`, `1d`	5min
User session timeout	The timeout after which ADB Control user sessions should be terminated. Requires the same format as the ***Blocking time period*** field	1d

To change the default settings, edit the fields and click ***Apply***. Click ***Revert*** if you need to undo the changes that have not been yet saved by clicking ***Apply***.



Minimum password length: 12

Maximum password length: 128

Authorization attempt limit: 5

Type: Selected time period

Blocking time period: 5min

User session timeout: 1d

The Configuration → Security tab

General settings

Daria Barysheva

Contents

To Table of Contents

- General
- Sessions
- Metrics

The ***General*** tab on the ***Configuration*** page in the ADB Control web interface allows you to configure general-purpose settings. This page consists of three tabs, each of which is described in detail below.

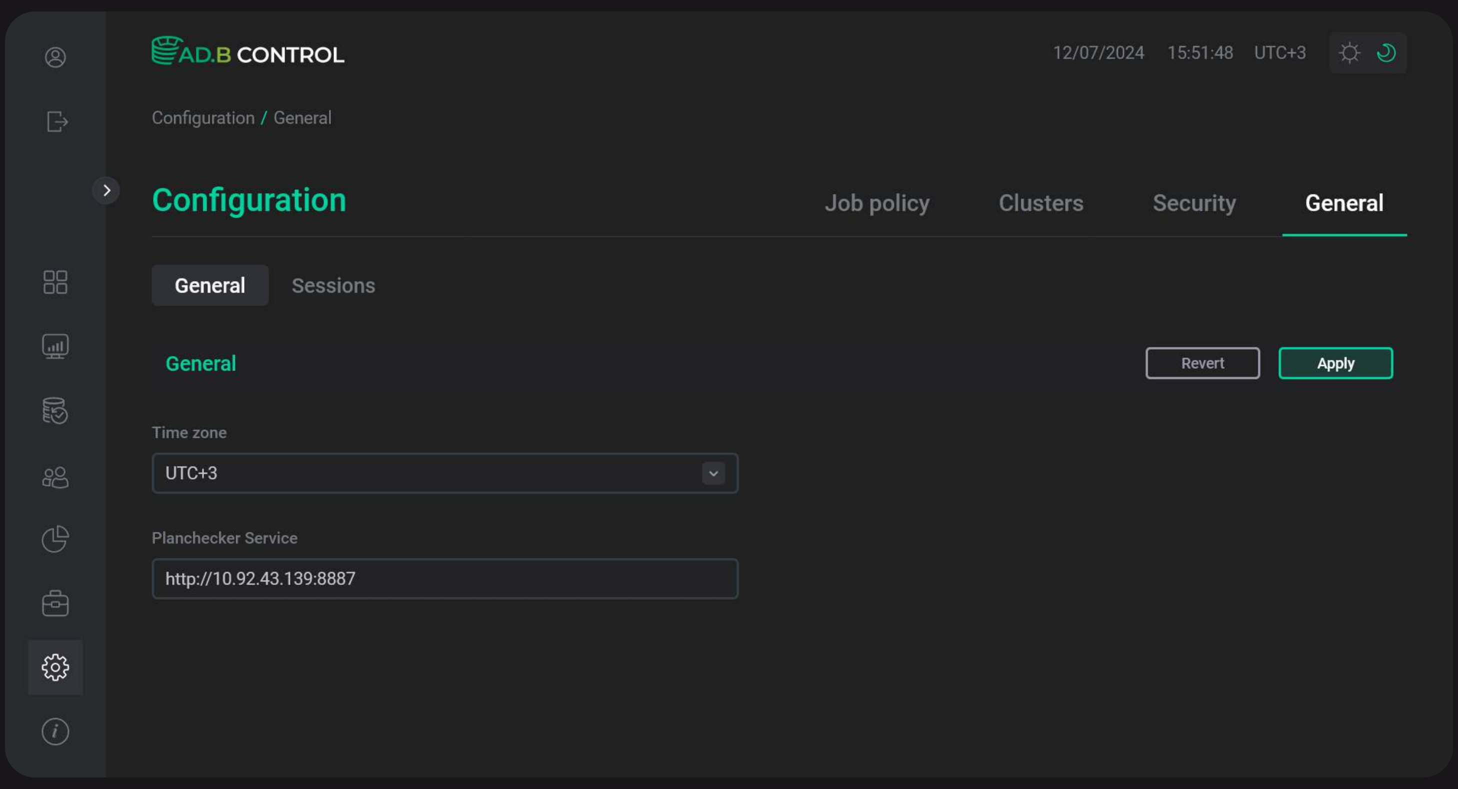
To change the default settings, edit the fields and click ***Apply***. Click ***Revert*** if you need to undo the changes that have not been yet saved by clicking ***Apply***.

General

The ***General*** tab allows you to set the general parameters.

The "Configuration → General → General" tab fields

Field	Description	Default value
Time zone	A user time zone	UTC+3
Planchecker service	Planchecker URL. This parameter is not applicable in ADB 7	—



AD.B CONTROL
12/07/2024 15:51:48 UTC+3
Configuration / General
Configuration
Job policy
Clusters
Security
General
General
Sessions
General
Revert
Apply

Time zone: UTC+3

Planchecker Service: http://10.92.43.139:8887

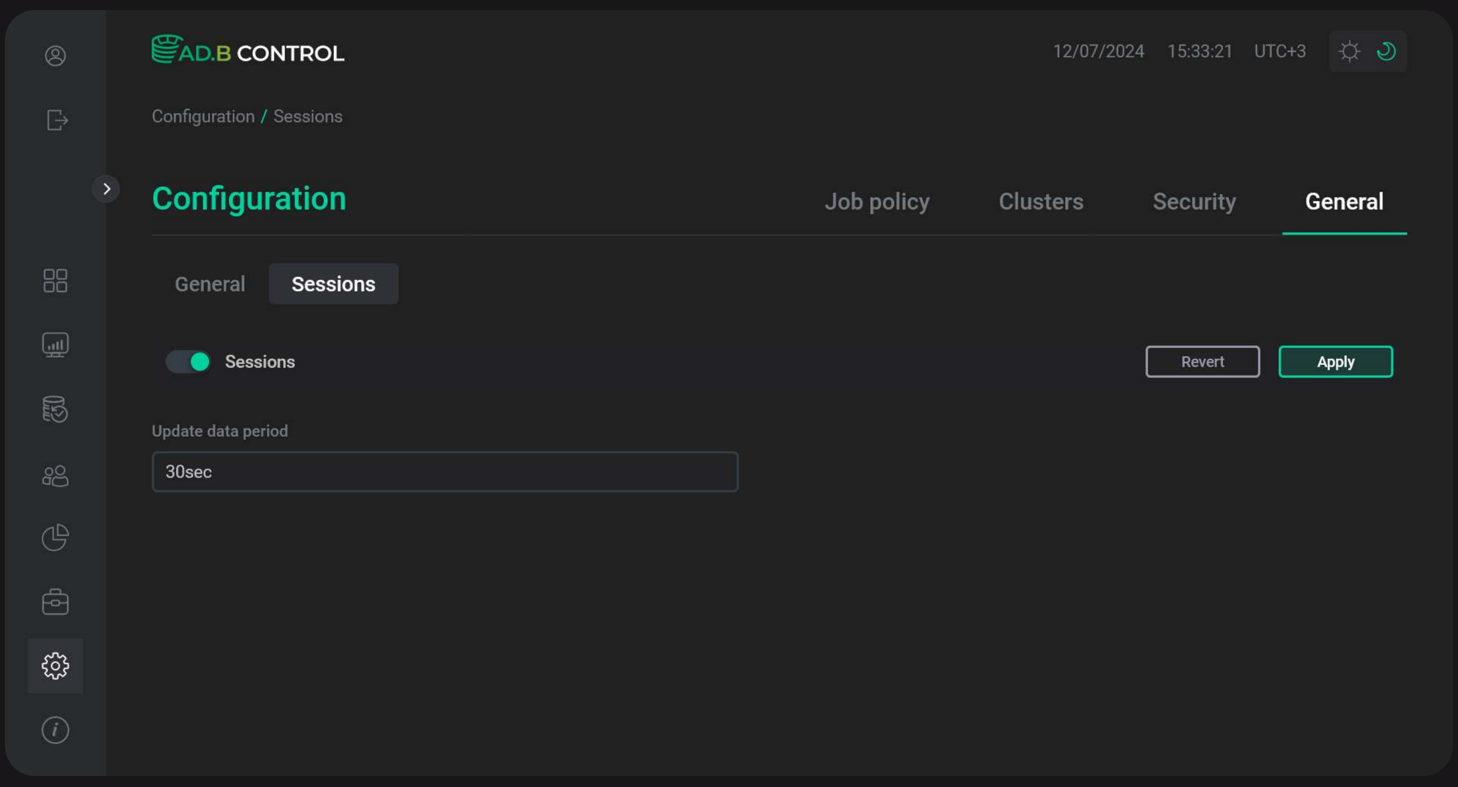
The Configuration → General → General tab

Sessions

The ***Sessions*** tab is designed to configure session monitoring.

The "Configuration → General → Sessions" tab fields

Field	Description	Default value
Sessions	A switcher that indicates whether to monitor sessions in ADB Control	True
Update data period	Indicates how often to receive information on sessions	30sec



AD.B CONTROL
12/07/2024 15:33:21 UTC+3
Configuration / Sessions
Configuration
Job policy
Clusters
Security
General
General
Sessions
Revert
Apply

Sessions: ●

Update data period: 30sec

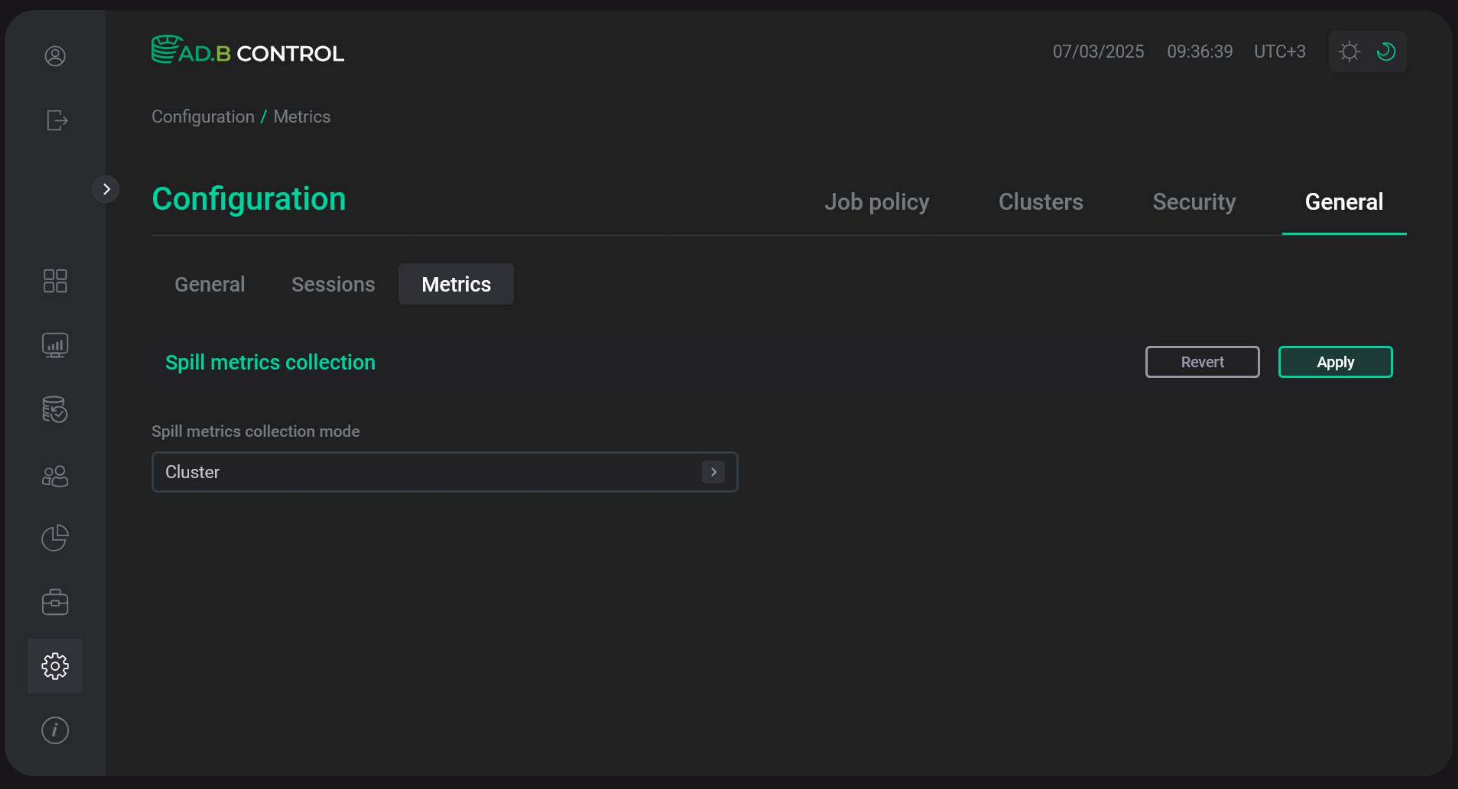
The Configuration → General → Sessions tab

Metrics

The ***Metrics*** tab allows you to configure collecting various metrics. This tab is available starting with ADB Control 4.12.0.

The "Configuration → General → Metrics" tab fields

Field	Description	Default value
Spill metrics collection mode	Mode of collecting spill file metrics: - `Cluster` — only for cluster; - `Cluster and segment` — for whole cluster and each segment. If the `Cluster and segment` value is selected, the ***Spill size*** and ***Spill files number*** metrics will be collected and shown for each cluster segment in the Performance section on the page with command details	Cluster



AD.B CONTROL
07/03/2025 09:36:39 UTC+3
Configuration / Metrics
Configuration
Job policy
Clusters
Security
General
General
Sessions
Metrics
Spill metrics collection
Revert
Apply

Spill metrics collection mode: Cluster

The Configuration → General → Metrics tab

Dashboard

Daria Barysheva

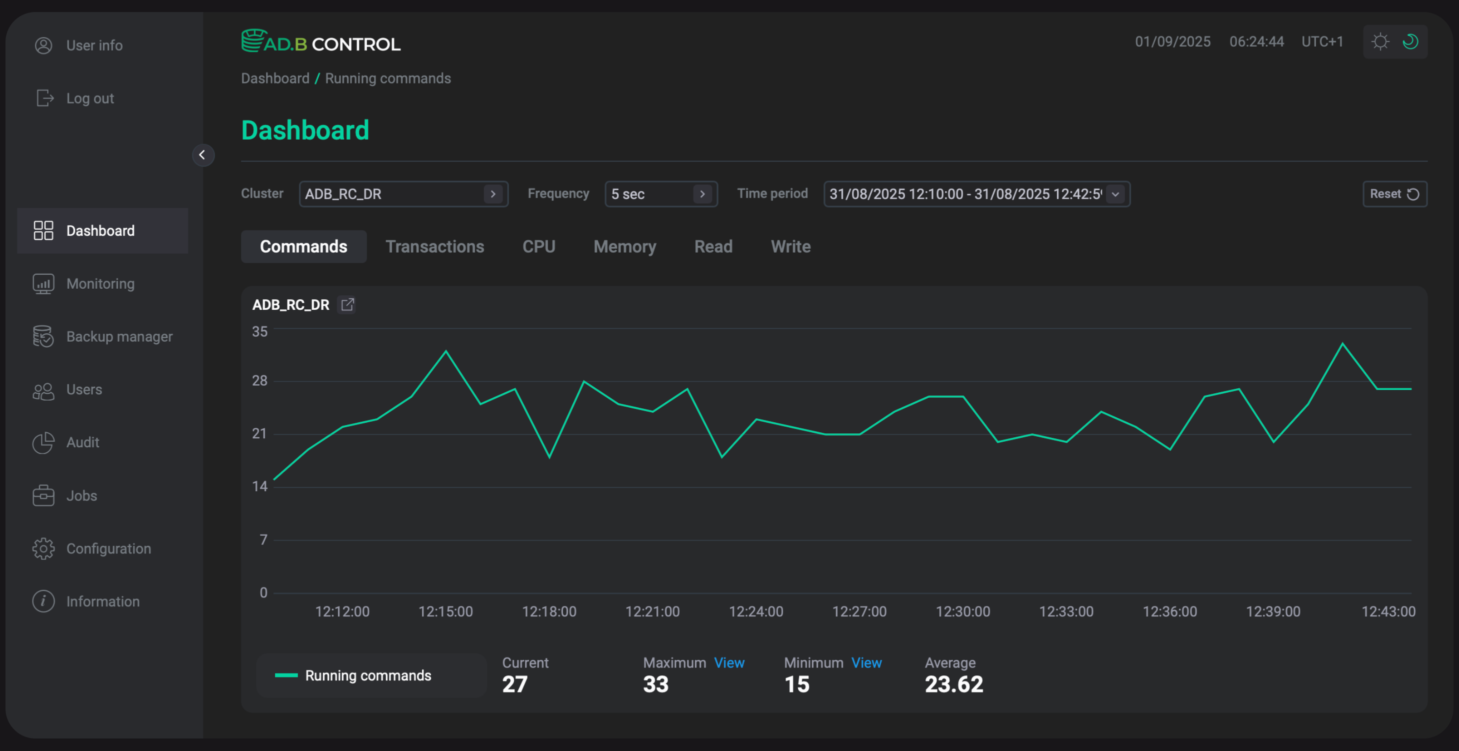
To Table of Contents

The **Dashboard** page in the ADB Control web interface allows you to display charts for tracking the dynamic changes of various metrics over the selected time period in the ADB clusters connected for monitoring.

Each tab on the page corresponds to a separate chart:

- ***Commands*** — a number of SQL commands in the `Running` status.
- ***Transactions*** — a number of transactions in the `Active` status.
- ***CPU*** — a total CPU consumption in the ADB cluster (percentage).
- ***Memory*** — a total virtual memory allocation in the ADB cluster (percentage).
- ***Read*** — a total data read amount in the ADB cluster (in bytes).
- ***Write*** — a total data write amount in the ADB cluster (in bytes).

To open a full screen view of any of the charts, click the ↗ icon to the right of the cluster name in the top part of the chart.

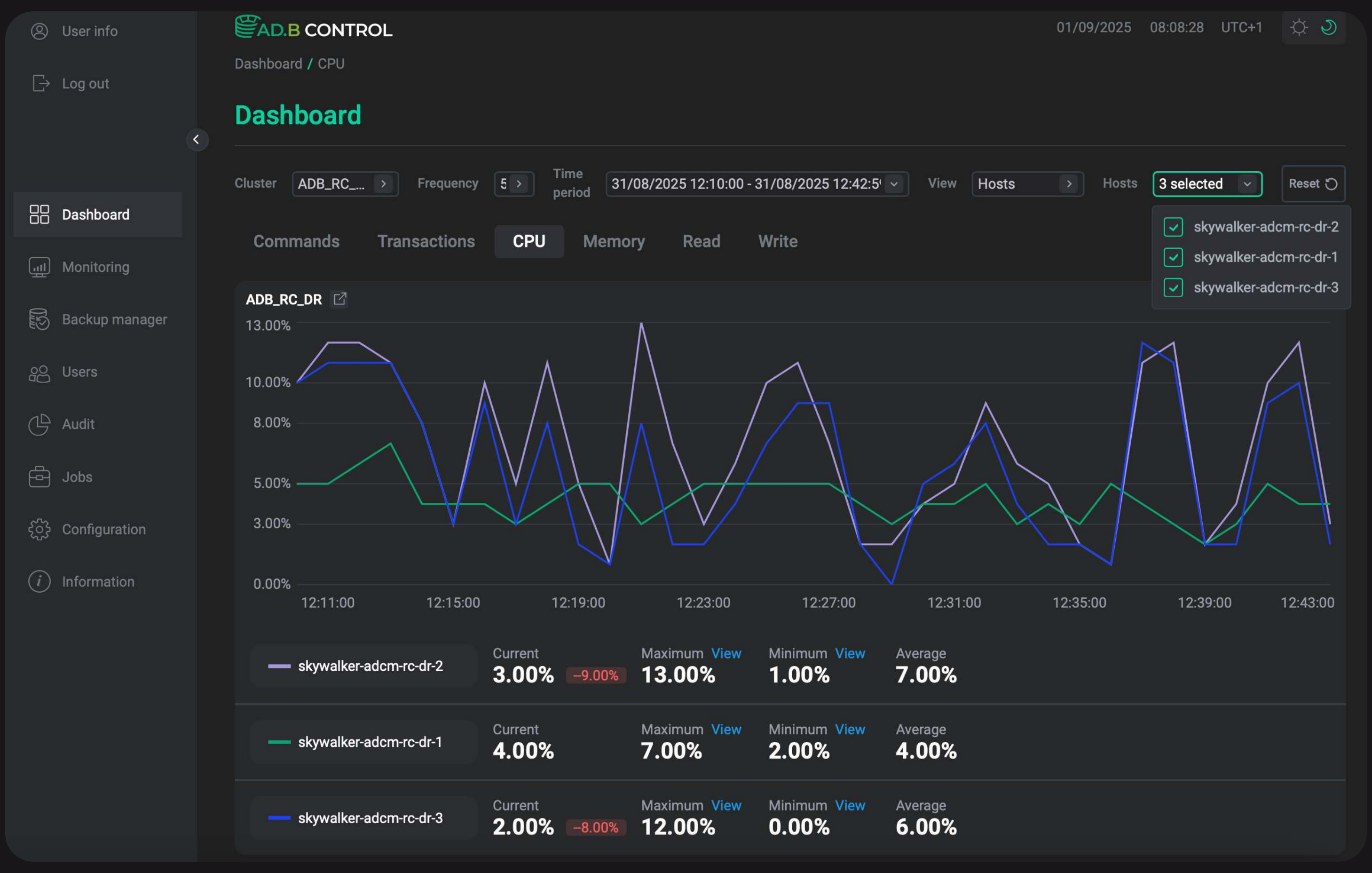

User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
01/09/2025 06:24:44 UTC+1
Dashboard / Running commands
Dashboard
Cluster ADB_RC_DR
Frequency 5 sec
Time period 31/08/2025 12:10:00 - 31/08/2025 12:42:59
Reset
Commands
Transactions
CPU
Memory
Read
Write
ADB_RC_DR
35
28
21
14
7
0
12:12:00
12:15:00
12:18:00
12:21:00
12:24:00
12:27:00
12:30:00
12:33:00
12:36:00
12:39:00
12:43:00
Running commands
Current 27
Maximum View 33
Minimum View 15
Average 23.62

The Dashboard page

To configure charts, you can use the following fields in the top part of the page:

- ***Cluster*** — the ADB cluster for which all charts should be drawn. The default cluster can be selected in the ADB Control settings.
- ***Frequency*** — defines how often to update charts.
- ***Time period*** — the time period for which a chart is required. In the window that opens after clicking the field, you can select an interval from the offered options or set the custom time range.
- ***View*** — defines how to display charts. The field is hidden on the ***Commands*** and ***Transactions*** tabs. Possible values:
  - `Summary` — display a summary chart for the whole ADB cluster.
  - `Hosts` — display information for each cluster host in the form of separate charts.
- ***Hosts*** — the drop-down list that is used to select cluster hosts. Becomes enabled if the `Hosts` value is selected in the ***View*** field.

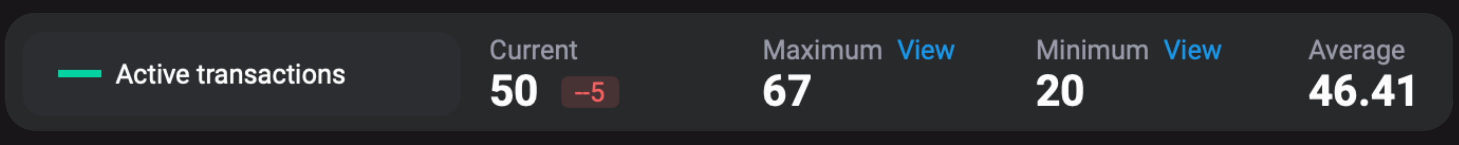
The figure below shows the option of displaying chart information in the context of cluster hosts.


User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
01/09/2025 08:08:28 UTC+1
Dashboard / CPU
Dashboard
Cluster ADB_RC_...
Frequency 1
Time period 31/08/2025 12:10:00 - 31/08/2025 12:42:59
View Hosts
Hosts 3 selected
Reset
☑ skywalker-adcm-rc-dr-2
☑ skywalker-adcm-rc-dr-1
☑ skywalker-adcm-rc-dr-3
Commands
Transactions
CPU
Memory
Read
Write
ADB_RC_DR
13.00%
10.00%
8.00%
5.00%
3.00%
0.00%
12:11:00
12:15:00
12:19:00
12:23:00
12:27:00
12:31:00
12:35:00
12:39:00
12:43:00
skywalker-adcm-rc-dr-2
Current 3.00% -9.00%
Maximum View 13.00%
Minimum View 1.00%
Average 7.00%
skywalker-adcm-rc-dr-1
Current 4.00%
Maximum View 7.00%
Minimum View 2.00%
Average 4.00%
skywalker-adcm-rc-dr-3
Current 2.00% -8.00%
Maximum View 12.00%
Minimum View 0.00%
Average 6.00%

Display a chart in the context of hosts

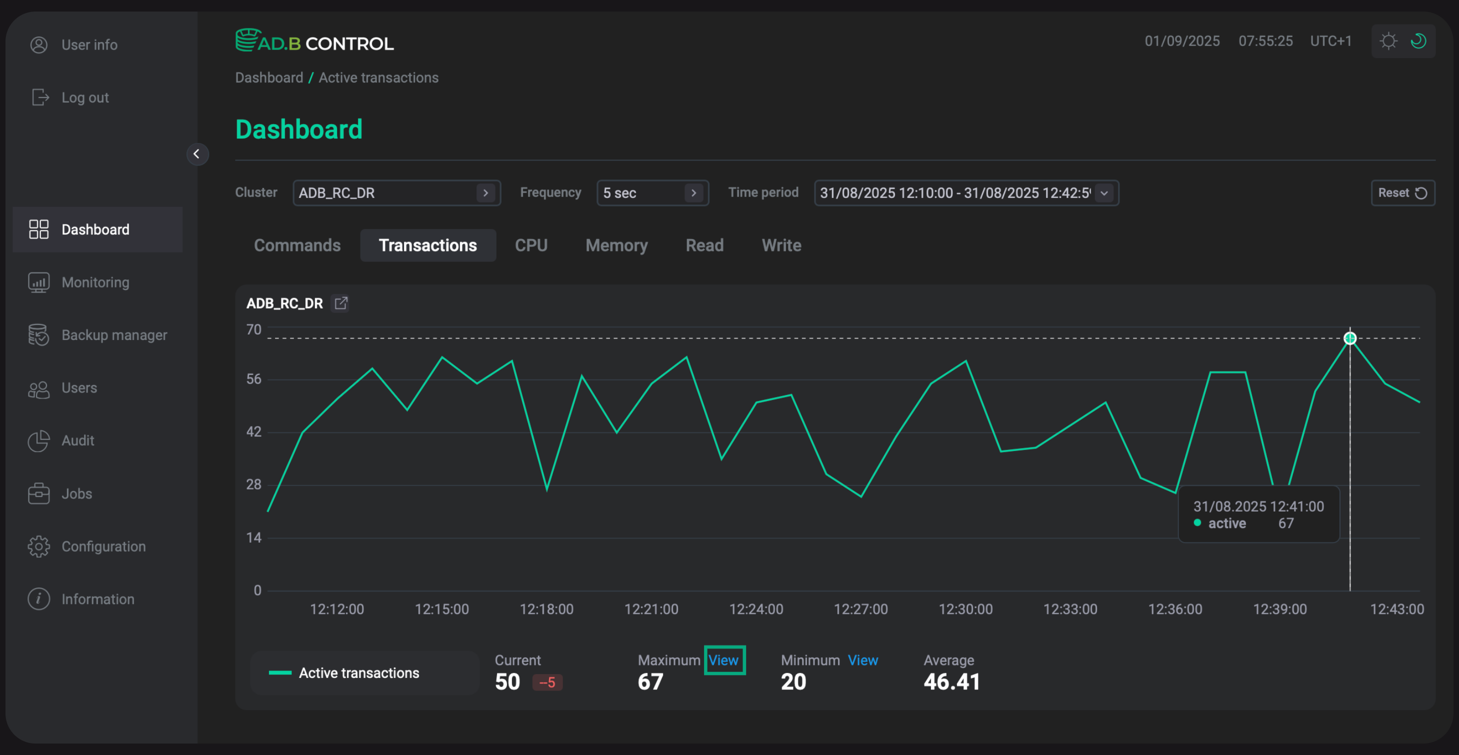
On each tab, you can see the following information under charts:

- ***Current*** — the last metric value received over the selected time period. To the right of the current value, you can see a special indicator that shows the difference between the previous and current metric values (if any).
- ***Maximum*** — the maximum metric value received over the selected time period. By clicking ***View***, you can head to the maximum value on the chart.
- ***Minimum*** — the minimum metric value received over the selected time period. By clicking ***View***, you can head to the minimum value on the chart.
- ***Average*** — the average value, which is calculated based on all metric values received over the selected time period.


Active transactions
Current 50 -5
Maximum View 67
Minimum View 20
Average 46.41

Aggregated metric information

Below, you can see how to highlight the maximum or minimum metric value on the chart by clicking ***View***.


User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
01/09/2025 07:55:25 UTC+1
Dashboard / Active transactions
Dashboard
Cluster ADB_RC_DR
Frequency 5 sec
Time period 31/08/2025 12:10:00 - 31/08/2025 12:42:59
Reset
Commands
Transactions
CPU
Memory
Read
Write
ADB_RC_DR
70
56
42
28
14
0
12:12:00
12:15:00
12:18:00
12:21:00
12:24:00
12:27:00
12:30:00
12:33:00
12:36:00
12:39:00
12:43:00
31/08/2025 12:41:00
active 67
Active transactions
Current 50 -5
Maximum View 67
Minimum View 20
Average 46.41

View the maximum metric value on the chart

You can also see a metric value for any other time point by hovering the mouse over the chart.

Monitoring of commands

Daria Barysheva

ADB CONTROL

Contents

To Table of Contents

- Queries
  - Online
  - History
- Command details
  - Header
  - Overview
  - Performance
  - Command text
  - Query plan sections
- System metrics collected for commands
- Groups
  - Group details
- SQL ID
  - Factors affecting SQL ID
  - Factors not affecting SQL ID

The *Monitoring* → *Commands* page in the ADB Control web interface displays the SQL commands that are performed in the ADB clusters connected to the monitoring system. This page includes two tabs:

- Queries shows individual SQL commands.
- Groups shows SQL commands grouped by SQL ID, node number, and node type. The tab is available starting with ADB Control 4.14.2.

Queries

The ***Queries*** tab displays SQL commands. It includes two tabs, ***Online*** and ***History***, described in detail below.

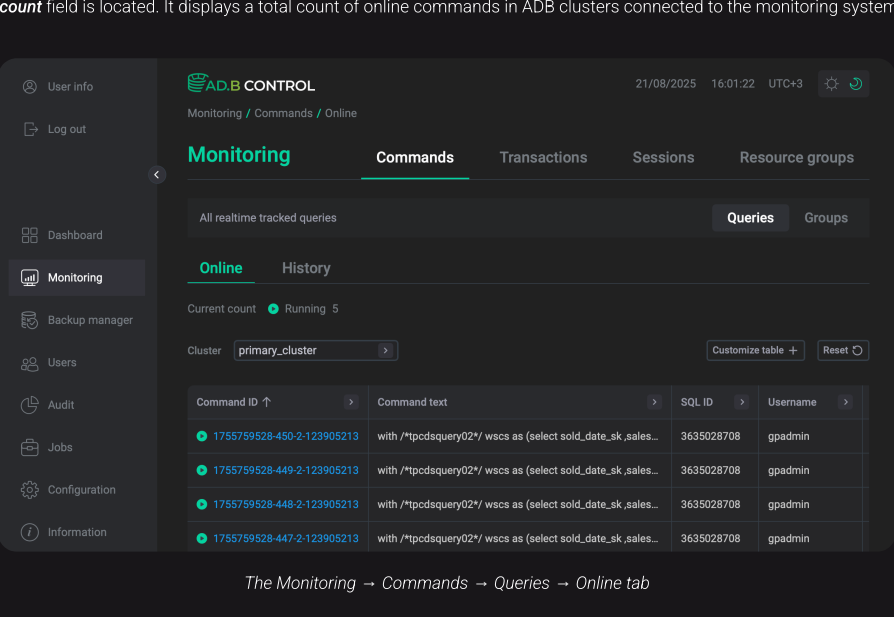
- By default, ADB Control monitors SQL commands longer than 5 seconds and displays them on the ***Monitoring*** → ***Commands*** page.
- Starting with ADB Control 4.1.0, only top-level queries are fixed in the monitoring system by default. This excludes all queries that are called during the processing of the top-level query within functions, SPI (Server Programming Interface), etc. If you need to enable monitoring for inner queries, set the `true` value for the following GUCs (starting with 4.7.5 version, you can edit them in the ADB Control web UI):
  - `adcc.monitor_utility_inner_queries` — defines whether to monitor inner queries for utility commands like `COPY`, `VACUUM`, DDL, and so on.
  - `adcc.monitor_inner_queries` — defines whether to monitor inner queries for other commands.

Online

The ***Monitoring*** → ***Commands*** → ***Queries*** → ***Online*** tab displays commands in the following statuses:

- `Queued` — the commands on which information about the actual start of execution has not yet received.
- `Running` — the commands that are currently running.
- `Cancelling` — the commands that are currently cancelling (as a part of the transaction cancellation or termination).

At the top of the tab, the ***Current count*** field is located. It displays a total count of online commands in ADB clusters connected to the monitoring system.



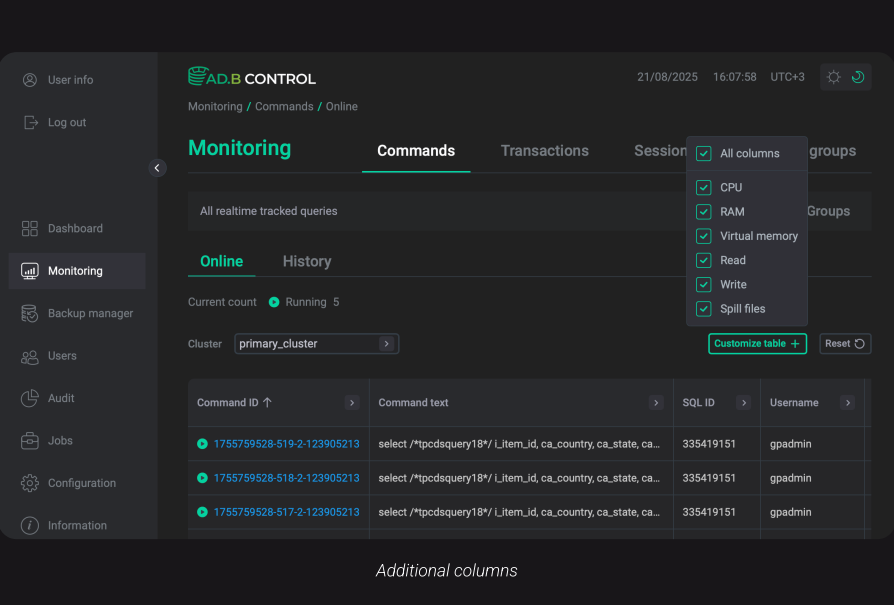
Command ID	Command text	SQL ID	Username
1755759528-450-2-123905213	with /*tpcdsquery02*/ wscs as (select sold_date_sk ,sales...	3635028708	gpadmin
1755759528-449-2-123905213	with /*tpcdsquery02*/ wscs as (select sold_date_sk ,sales...	3635028708	gpadmin
1755759528-448-2-123905213	with /*tpcdsquery02*/ wscs as (select sold_date_sk ,sales...	3635028708	gpadmin
1755759528-447-2-123905213	with /*tpcdsquery02*/ wscs as (select sold_date_sk ,sales...	3635028708	gpadmin

The Monitoring → Commands → Queries → Online tab

On the ***Monitoring*** → ***Commands*** → ***Queries*** → ***Online*** tab, there is a table with the following information on commands.

Field	Description
Command ID	A unique command identifier, which includes: ■ cluster start timestamp; ■ session number; ■ command number within the current session; ■ hash code based on the cluster name (can be negative).
Command text	The first symbols of a command text. To view the full text (in case of long queries), hover the mouse over the field value
SQL ID	A common identifier for SQL commands with the same structure
Username	A name of the user who started the command
Status	A command status. Possible values are listed above
Database	A name of the database where the command is launched
Resource group	A name of the resource group that is used for the command
Cluster	A name of the ADB cluster where the command is launched
Planner	A name of the query optimizer that is used to produce the query execution plan. Possible values: ■ `GPORCA` — GPORCA is used (`optimizer = on`). ■ `Fallbacked` — after the attempt to produce a plan via GPORCA (`optimizer = on`) failed, Postgres query optimizer is used. ■ `Legacy` — Postgres query optimizer is used (`optimizer = off`).
Submitted	The timestamp when the user submitted the command (in the `DD/MM/YYYY HH:mm:ss` format)
Tags	The command tags
Run time	A total command duration in hours, minutes, seconds

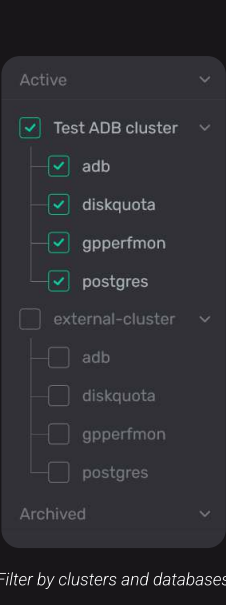
If necessary, you can add columns to the table. To do this, click ***Customize table*** and select column names in the section that opens. Currently you can add system metrics that are described below.



Command ID	Command text	SQL ID	Username
1755759528-519-2-123905213	select /*tpcdsquery18*/ i_item_id, ca_country, ca_state, ca...	335419151	gpadmin
1755759528-518-2-123905213	select /*tpcdsquery18*/ i_item_id, ca_country, ca_state, ca...	335419151	gpadmin
1755759528-517-2-123905213	select /*tpcdsquery18*/ i_item_id, ca_country, ca_state, ca...	335419151	gpadmin

Additional columns

Above the table with a list of commands, the ***Cluster*** filter is located. You can use this filter to select the ADB cluster and its databases for which you want to display data in the table. Initially all databases of the default cluster are selected.



Filter by clusters and databases

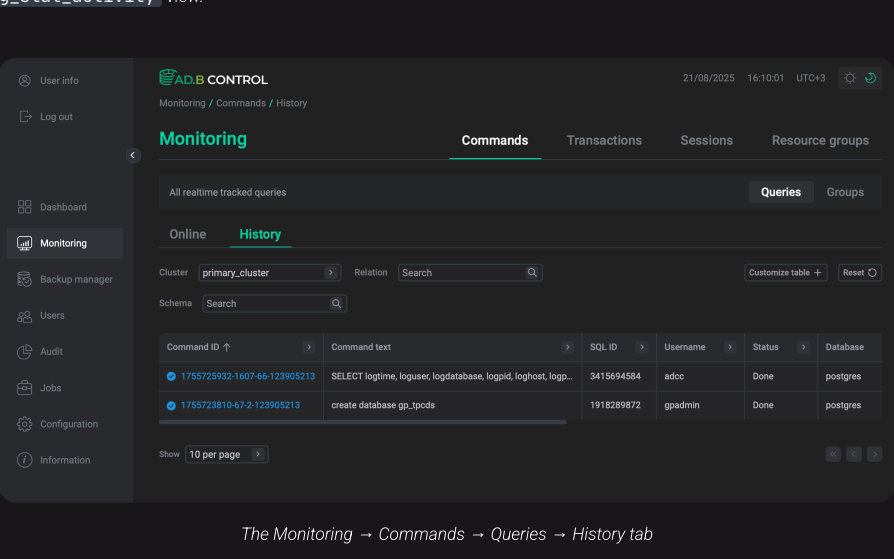
In the column headers of the table with a list of commands, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Username***), the search value should be entered. For columns that show date and time (e.g. ***Submitted***), the time range can be selected from the calendar.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

History

The ***Monitoring*** → ***Commands*** → ***Queries*** → ***History*** tab displays commands in the following statuses:

- `Done` — successfully completed commands.
- `Cancelled` — cancelled commands (as a part of the transaction cancellation or termination).
- `Error` — the commands during which some errors occurred.
- `Unknown` — the commands, the final status of which is undefined. It is set on a timeout for queries in the `Queued`, `Running`, `Cancelling` statuses, for which there are no corresponding rows in the `pg_stat_activity` view.



Command ID	Command text	SQL ID	Username	Status	Database
1755759392-1607-66-123905213	SELECT logtime, loguser, logdatabase, logpid, loghost, logs...	3415694084	adcc	Done	postgres
1755759816-67-3-123905213	create database gp_tpcds	1918289672	gpadmin	Done	postgres

The Monitoring → Commands → Queries → History tab

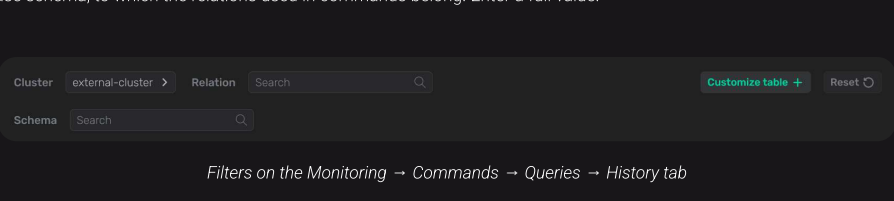
Most of the fields in the table with the list of commands on the ***History*** tab match the fields that are described above for the ***Online*** tab. Additionally, the following field is available:

- ***Ended*** — a command end timestamp in the `DD/MM/YYYY HH:mm:ss` format. The field is filled in both as a result of successful execution and in case of error or cancellation.

Like on the ***Online*** tab, you have the ability to add additional columns to the table by clicking ***Customize table***.

Above the table with a list of commands, there are filters that you can use to select specific data. In addition to the ***Cluster*** filter described above, the ***History*** tab contains the following filters:

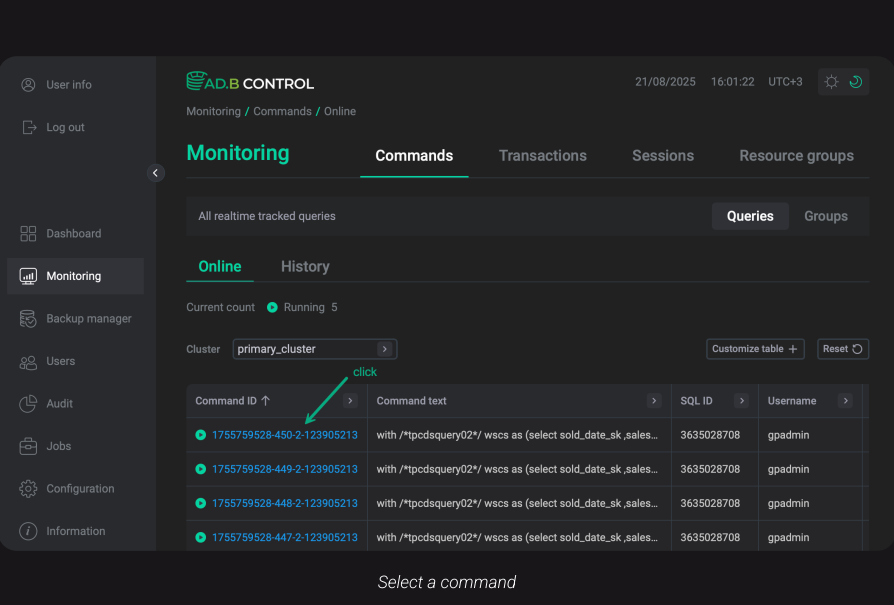
- ***Relation*** — filter by the relations that are used in commands. Enter a full value.
- ***Schema*** — filter by the database schema, to which the relations used in commands belong. Enter a full value.



Filters on the Monitoring → Commands → Queries → History tab

Command details

To view the command details, click the command identifier (***Command ID***) in the table on the Monitoring → Commands → Queries → Online or Monitoring → Commands → Queries → History tab.



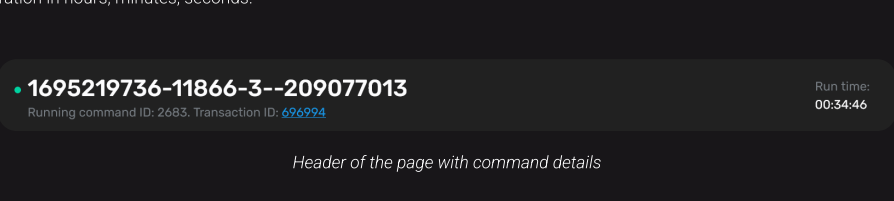
Select a command

The following page contains multiple sections that are described below.

Header

At the top of the page, the following information is displayed:

- The command unique identifier (see ***Command ID*** above). In the example below — `1695219736-11866-3-209077013`.
- The command status (see ***Status*** above). In the example below — `Running`.
- The command sequence number in ADB Control (starting with `1`). The continuous numbering is used across all clusters. In the example below — `2683`.
- ***Transaction ID*** — the transaction identifier. To view transaction details, click the identifier.
- ***Run time*** — the command duration in hours, minutes, seconds.



Header of the page with command details

Overview

The ***Overview*** section displays the general command information. In addition to the fields mentioned above, the following information is available:

- ***Run time*** — the command duration in hours, minutes, seconds.
- ***Est. progress*** — the progress of the command execution, which is calculated as a percentage of the time predicted by the query planner. The following formula is used:

$$Est.progress = \frac{\sum(NodeProgress(x) \cdot Est.Cost(x))}{\sum Est.Cost(x)}$$

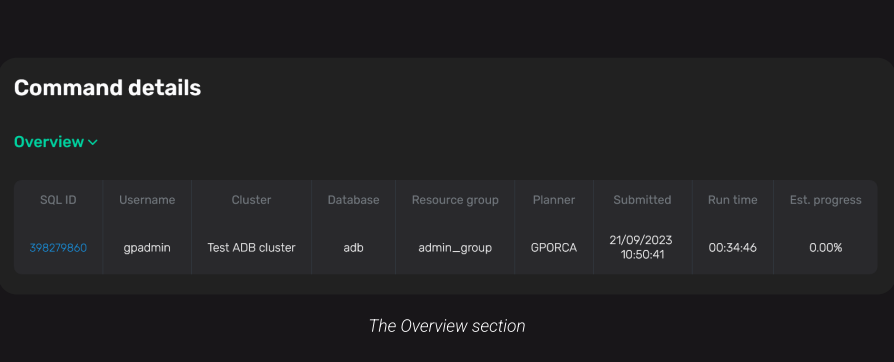
where:

- `Est.Cost(x)` — the estimate of the plan node cost made by the planner.
- `NodeProgress(x)` — the progress for each node of the query plan, which is calculated by the following formula:

$$NodeProgress = \frac{ActualRows}{EstimatedRows}$$

where:

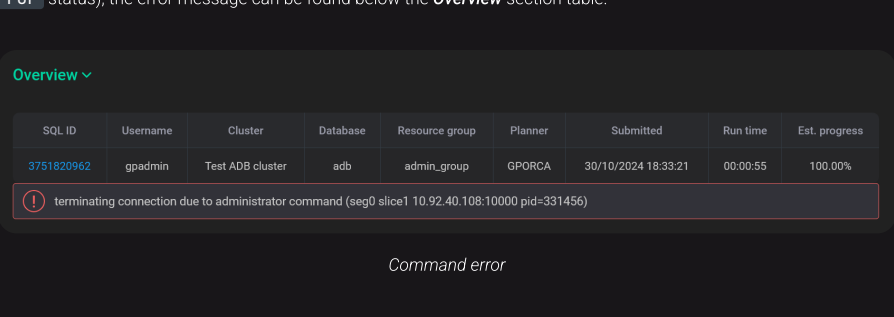
  - `ActualRows` — an actual number of processed table rows (tuples).
  - `EstimatedRows` — the planner's assumption of the number of rows (tuples) that will be processed for the current plan node.



SQL ID	Username	Cluster	Database	Resource group	Planner	Submitted	Run time	Est. progress
596279860	gpadmin	Test ADB cluster	adb	admin_group	GPORCA	21/09/2023 10:50:41	00:34:46	0.00%

The Overview section

If the command fails (has the `Error` status), the error message can be found below the ***Overview*** section table.



SQL ID	Username	Cluster	Database	Resource group	Planner	Submitted	Run time	Est. progress
3751820962	gpadmin	Test ADB cluster	adb	admin_group	GPORCA	30/10/2024 18:33:21	00:00:55	100.00%

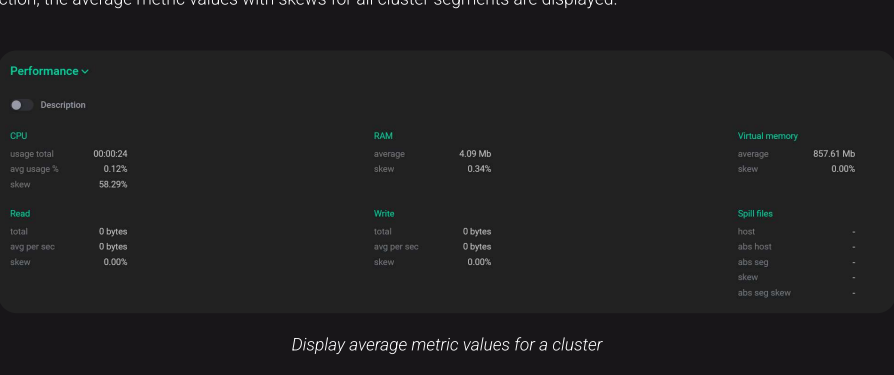
terminating connection due to administrator command (seg0 slice1 10.92.40.108:10000 pid=331456)

Command error

Performance

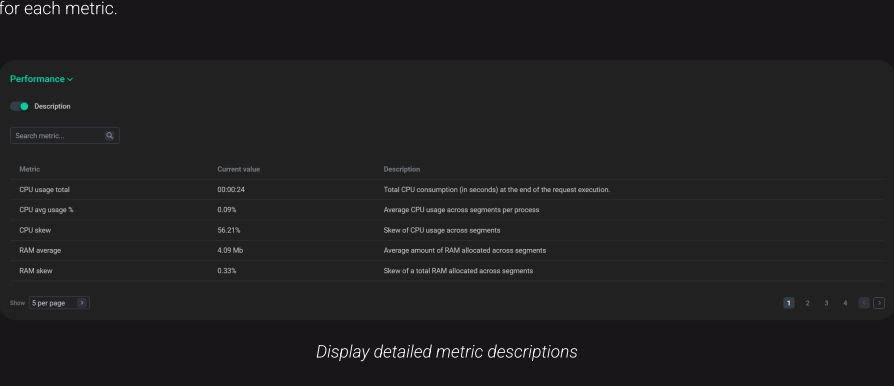
The ***Performance*** section displays the statistics of system resource consumption by the selected command. The metrics available for monitoring are listed in System metrics collected for commands.

At the top of the ***Performance*** section, the average metric values with skews for all cluster segments are displayed.



Display average metric values for a cluster

You can select an alternative display of metrics by activating the switcher that is located under the ***Performance*** section. The result is a table where not only the current value but also a detailed description is displayed for each metric.

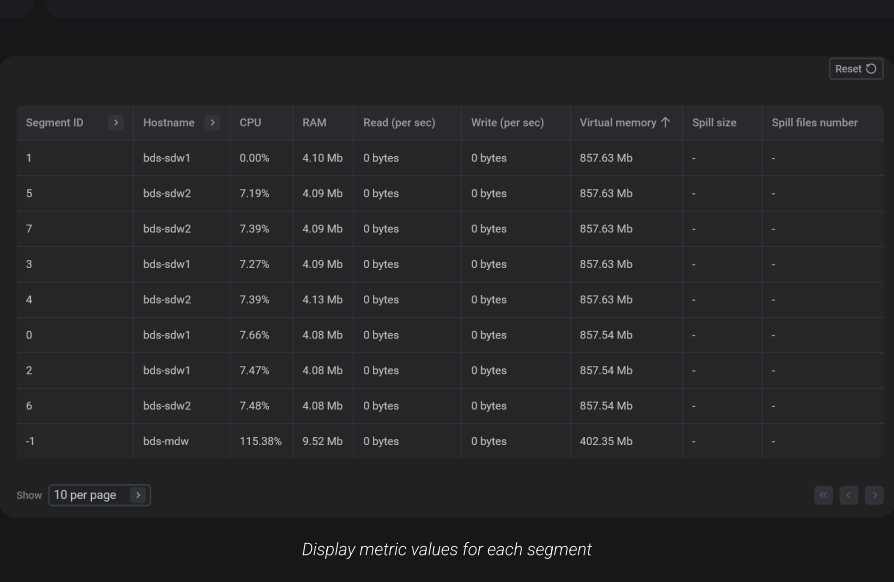


Display detailed metric descriptions

In the bottom part of the ***Performance*** section, the metric values are shown separately for each cluster segment, including the coordinator segment `seg-1` (where ***Segment ID*** equals to `-1`).

Fields of the table with segment metrics

Field	Description
Segment ID	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables*** → ***gp_segment_configuration*** section of the Tables overview article
Hostname	A host name
CPU	The amount of CPU consumed by the segment (percentage). Based on the metric values for all segments, the average value CPU avg usage % is calculated
RAM	The amount of RAM consumed by the segment (in bytes). Based on the metric values for all segments, the average value RAM average is calculated
Read (per sec)	The amount of data read per second by the segment (in bytes). Based on the metric values for all segments, the average value Read avg per sec is calculated
Write (per sec)	The amount of data written per second by the segment (in bytes). Based on the metric values for all segments, the average value Write avg per sec is calculated
Virtual memory	The amount of virtual memory allocated by the segment (in bytes). Based on the metric values for all segments, the average value Virtual memory average is calculated
Spill size	Size of spill files (in bytes). Filled in if the ***Spill metrics collection mode*** value is set to `Cluster and segment` on the Configuration → General → Metrics tab
Spill files number	Number of spill files. Filled in if the ***Spill metrics collection mode*** value is set to `Cluster and segment` on the Configuration → General → Metrics tab



Segment ID	Hostname	CPU	RAM	Read (per sec)	Write (per sec)	Virtual memory	Spill size	Spill files number
1	bds-sdw1	0.00%	4.10 Mb	0 bytes	0 bytes	857.63 Mb	-	-
5	bds-sdw2	7.19%	4.09 Mb	0 bytes	0 bytes	857.63 Mb	-	-
7	bds-sdw2	7.39%	4.09 Mb	0 bytes	0 bytes	857.63 Mb	-	-
3	bds-sdw1	7.27%	4.09 Mb	0 bytes	0 bytes	857.63 Mb	-	-
4	bds-sdw2	7.39%	4.13 Mb	0 bytes	0 bytes	857.63 Mb	-	-
0	bds-sdw1	7.66%	4.08 Mb	0 bytes	0 bytes	857.54 Mb	-	-
2	bds-sdw1	7.47%	4.08 Mb	0 bytes	0 bytes	857.54 Mb	-	-
6	bds-sdw2	7.48%	4.09 Mb	0 bytes	0 bytes	857.54 Mb	-	-
-1	bds-mdw	115.38%	9.52 Mb	0 bytes	0 bytes	402.35 Mb	-	-

Display metric values for each segment

Starting with ADBM 2.1.2 you can filter and sort data in the table displayed above as follows:

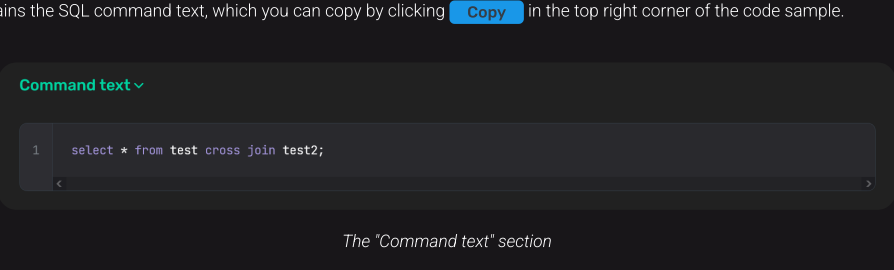
- Filters are available for the ***Segment ID*** and ***Hostname*** columns. To apply a filter, click the > icon in the column header and enter the value to search. The ▼ icon means that the filter is defined for the column. To reset all filters, click ***Reset***.
- Data sorting is available for all columns except ***Hostname***. To change a sort order for a column, click the ↑ or ↓ icon in the column header.

IMPORTANT

System metrics are collected from ADB segments by ADB Control agents only for commands longer than 15 seconds. Due to this fact, the section with metrics will be empty for the commands that complete faster.

Command text

The ***Command text*** section contains the SQL command text, which you can copy by clicking Copy in the top right corner of the code sample.



```
select * from test cross join test2;
```

The "Command text" section

Query plan sections

Below the ***Command text*** section, there are several tabs that display query execution details:

- ***Plan & progress***
- ***Text***
- ***Relations***

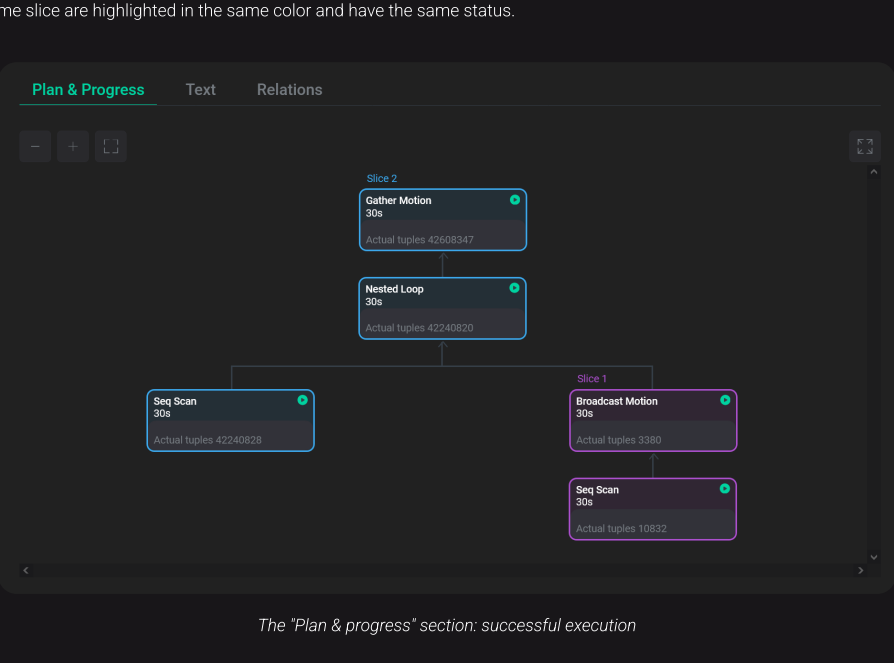
The first tab ***Plan & progress*** contains a graphical view of the query plan in the form of a tree. Execution is done from child nodes to the root. Each plan node displays a type of the performed operation and the following information:

- **For DB queries in progress**:
  - The actual number of processed tuples at the current moment. This value is being refreshed.
  - The actual node execution time of a slice (group of nodes) at the current moment. This value is being refreshed and displayed on all nodes that belong to the same slice.
- **For finished DB queries**:
  - The total node execution time.
  - The total number of processed tuples.
  - The ratio of the node execution time to the total query execution time (in percentage).

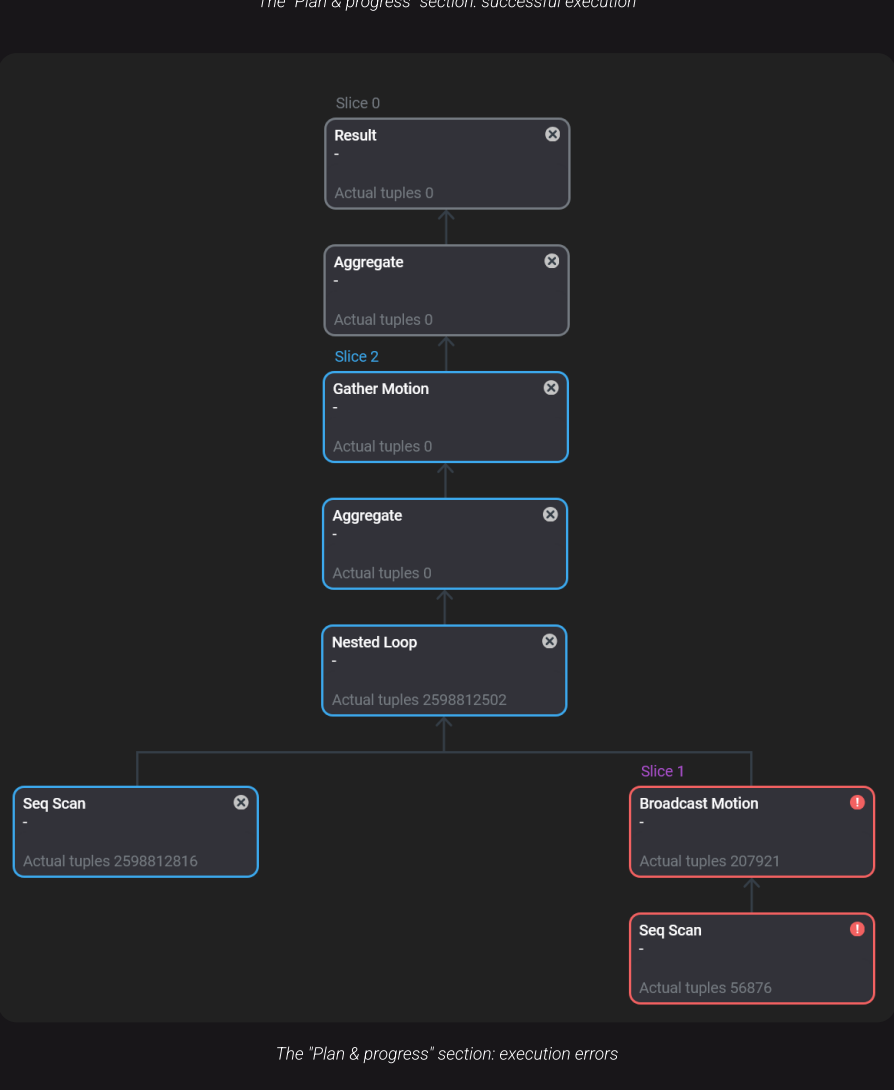
In the top right corner of each plan node, there is an icon corresponding to its status:

- ● — the plan node is being executed.
- ✓ — the plan node is finished.
- ! — some error occurred when running the slice that contains the current plan node. The error status is available starting with ADB Control 4.10.3. Plan nodes in the error status are highlighted in red.
- ⊗ — the plan node is unreachable. This status is used for plan nodes that are above the error slice in the query plan tree, and also for those nodes whose parent node appeared in the error slice in the initialization status (before execution). The unreachable status is available starting with ADB Control 4.10.3.

Plan nodes that belong to the same slice are highlighted in the same color and have the same status.



The "Plan & progress" section: successful execution

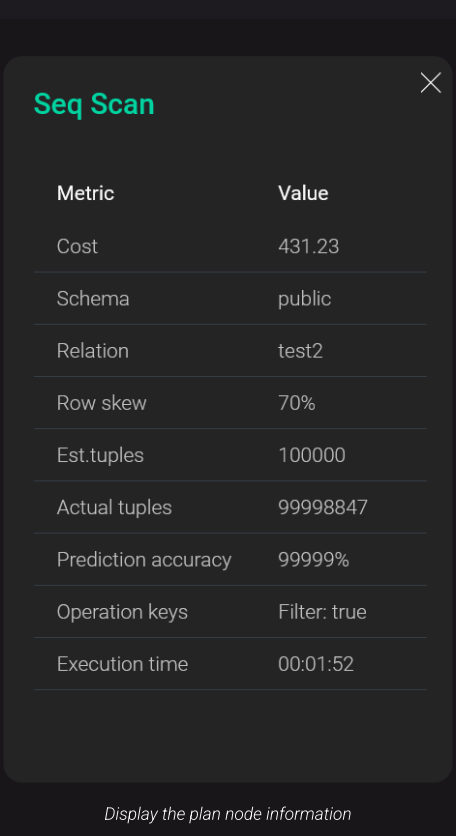


The "Plan & progress" section: execution errors

When you click on a query plan node, a panel with additional information appears on the right side of the screen.

Plan node information

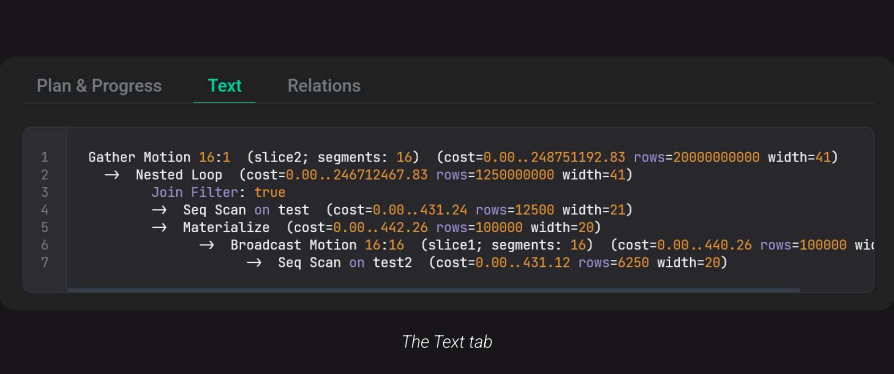
Field	Description
Cost	A plan node cost
Schema	A schema name. Can be used to distinguish eponymous tables that belong to different schemas
Relation	A name of the relation that is used in the current plan node
Row skew	A data skew indicator. The difference between `1` and the ratio of the average number of tuples obtained from the segments to the maximum: $RowSkew = 1 - \frac{avg(SegmentRows)}{max(SegmentRows)}$
Est.tuples	An estimated number of table rows (tuples) that should be processed or scanned by the plan node
Actual tuples	An actual number of processed table rows (tuples)
Prediction accuracy	A ratio of the actual number of processed tuples to the estimated number (percentage). The value that exceeds 100% indicates that the planner has made a wrong assumption about the number of tuples that should be processed/scanned when executing the plan node
Operation keys	Additional conditions that are used to perform specific operations. For example, `Hash Key`, `Join Filter`, `Merge Key`, `Filter`
Execution time	A plan node execution time in hours, minutes, seconds



Metric	Value
Cost	431.23
Schema	public
Relation	test2
Row skew	70%
Est.tuples	100000
Actual tuples	99998847
Prediction accuracy	99999%
Operation keys	Filter: true
Execution time	00:01:52

Display the plan node information

The next tab ***Text*** outputs a text presentation of the query plan, which is the result of the `EXPLAIN` command. You can copy the text by clicking the Copy icon in the top right corner of the code sample.



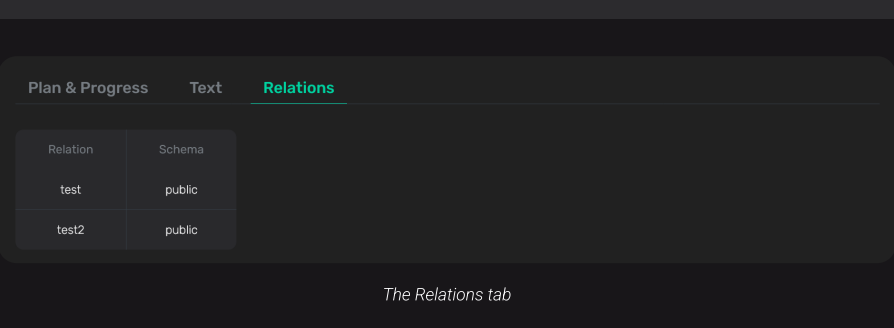
```
Gather Motion 16:1  (slice2; segments: 16)  (cost=0.00..248751192.83 rows=20000000000 width=41)
  ->  Nested Loop  (cost=0.00..246712467.83 rows=1250000000 width=41)
        Join Filter: true
        ->  Seq Scan on test  (cost=0.00..431.24 rows=12500 width=21)
        ->  Materialize  (cost=0.00..442.26 rows=100000 width=20)
              ->  Broadcast Motion 16:16  (slice1; segments: 16)  (cost=0.00..440.26 rows=100000 wi
                    ->  Seq Scan on test2  (cost=0.00..431.12 rows=6250 width=20)
```

The Text tab

The last tab ***Relations*** lists the database relations that are used in the query. For each relation, the name of the corresponding schema is displayed.

NOTE

To view audit details for the specific relation, you can click the row corresponding to that relation in the table above. This functionality is available for ADB Control users with appropriate permissions (see `View secured relation audit` in the Authorization article).



Relation	Schema
test	public
test2	public

The Relations tab

System metrics collected for commands

Both Online and History tabs allow you to view the statistics of the system resource consumption by commands (on the Performance tab in the command details). The metrics available for commands are listed below.

NOTE

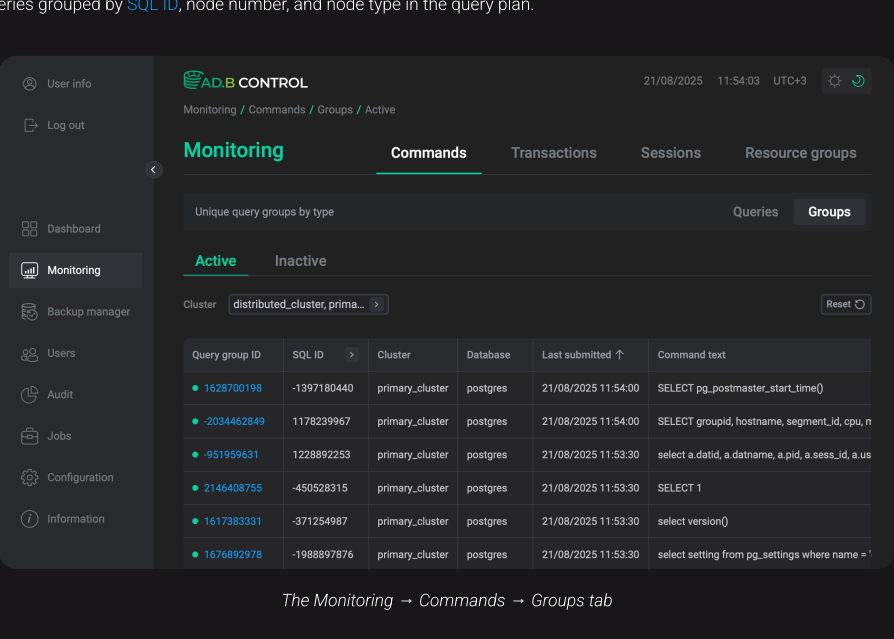
- The ***Online*** tab displays current metric values, whereas the ***History*** tab shows metric values at the end of the command execution.
- The following metrics are not calculated on the ***Online*** tab: ***Spill files max*** and ***Spill files skew***.
- The calculation of average and skew values does not take into account the coordinator segment (where ***Segment ID*** is equal to `-1`).

Command metrics

Group	Metric	Description
CPU	CPU usage total	The total CPU consumption across cluster segments (in seconds)
	CPU avg usage %	The CPU percent average for all segment processes executing the query. Calculated as the result of dividing the total CPU consumption on all cluster segments (excluding the coordinator and standby) by the total number of processes processing the query on segment hosts (excluding the coordinator and standby)
	CPU skew	The indicator that displays the CPU consumption skew across cluster segments (percentage). A value other than zero indicates that one of the segments uses more CPU than others
RAM	RAM average	The average amount of RAM consumed across cluster segments (in bytes). The RAM consumption at any given moment is calculated based on the operating system metric **rss (Resident Set Size)**
	RAM skew	The indicator that displays the RAM consumption skew across cluster segments (percentage). A value other than zero indicates that one of the segments uses more RAM than others
Virtual memory	Virtual memory average	The average amount of virtual memory allocated across cluster segments (in bytes). The virtual memory consumption at any given moment is calculated based on the operating system metric **vsize (Virtual Memory Size)**
	Virtual memory skew	The indicator that displays the virtual memory skew across cluster segments (percentage). A value other than zero indicates that one of the segments uses more virtual memory than others
Read	Read total	The total amount of data read across cluster segments (in bytes). The following formula is used for calculation: $ReadTotal = TotalReadSpeed \cdot QueryTime$
	Read avg per sec	The average amount of data read across cluster segments per second (in bytes). The data read rate at any given moment is calculated based on the I/O Read information from */proc/[pid]/stat* as a delta between the current and previous values
	Read skew	The indicator that displays the data reading skew across cluster segments (percentage). A value other than zero indicates that one of the segments reads more data from a disk than others
Write	Write total	The total amount of data written across cluster segments (in bytes). The following formula is used for calculation: $WriteTotal = TotalWriteSpeed \cdot QueryTime$
	Write avg per sec	The average amount of data written across cluster segments per second (in bytes). The data write rate at any given moment is calculated based on the I/O Write information from */proc/[pid]/stat* as a delta between the current and previous values
	Write skew	The indicator that displays the data writing skew across cluster segments (percentage). A value other than zero indicates that one of the segments writes more data to a disk than others
Spill files	Spill files host	The total amount of spill files on cluster hosts at the current moment (in bytes)
	Spill files abs host	The maximum value of ***Spill files host*** detected on cluster hosts during the command execution (in bytes)
	Spill files abs seg	The maximum amount of spill files produced by one cluster segment during the command execution (in bytes)
	Spill files skew	The indicator that displays the spill file amount skew (in percents) at the current moment. A value other than zero indicates that one of the segments uses more spill files than others
	Spill files abs seg skew	The indicator that displays the spill file amount skew (in percents) at the moment when the ***Spill files abs seg*** value was fixed. The following formula is used for calculation: $SpillFilesSkew = 1 - \frac{avg(SegmentSpillFiles)}{max(SegmentSpillFiles)}$

Groups

A **query group** is a set of SQL queries grouped by SQL ID, node number, and node type in the query plan.



AD.B CONTROL
21/08/2025 11:54:03 UTC+3
Monitoring / Commands / Groups / Active
Monitoring
Commands
Transactions
Sessions
Resource groups
Unique query groups by type
Queries
Groups
Active
Inactive
Cluster
distributed_cluster, prima...
Reset

Query group ID	SQL ID	Cluster	Database	Last submitted	Command text
1628700198	-1397180440	primary_cluster	postgres	21/08/2025 11:54:00	SELECT pg_postmaster_start_time()
-2054462849	1178239967	primary_cluster	postgres	21/08/2025 11:54:00	SELECT groupid, hostname, segment_id, cpu, n
-931959631	1228892253	primary_cluster	postgres	21/08/2025 11:53:30	select a.datid, a.datname, a.pid, a.sess_id, a.us
2146408755	-450528315	primary_cluster	postgres	21/08/2025 11:53:30	SELECT 1
1617383231	-371254987	primary_cluster	postgres	21/08/2025 11:53:30	select version()
1676892978	-1988897876	primary_cluster	postgres	21/08/2025 11:53:30	select setting from pg_settings where name = '

The Monitoring → Commands → Groups tab

When a query is received, ADB Control creates a group for every new combination of SQL ID, node number, and node type. If the next query matches these parameters of an existing group, it belongs to this group, and the ***Last submitted*** parameter of the group is updated. Otherwise, a new group is created.

NOTE

Currently, the duration of an SQL command determines on which tab the command is shown:

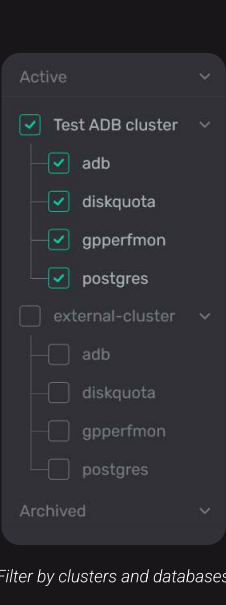
- By default, the Queries tab only shows SQL commands that run for 5 seconds or longer.
- The ***Groups*** tab shows query groups of the `Short` type, consisting only of SQL commands that are shorter than 5 seconds.

Query groups can be **active** and **inactive** depending on the time of the last query in the group. If the time of the last query is within the last 5 minutes, the group is considered active and is displayed on the ***Active*** tab. If more than 5 minutes have passed after the last query in the group, it is considered inactive and is displayed on the ***Inactive*** tab.

The ***Monitoring*** → ***Commands*** → ***Groups*** tab consists of a table with the following information.

Field	Description
Query group ID	An identifier of the query group calculated as a hash based on the SQL ID, node number, and node type
SQL ID	A common identifier for SQL commands with the same structure
Cluster	A name of the ADB cluster where the group queries are launched
Database	A name of the database where the group queries are launched
Last submitted	The time of the last query in the group
Command text	The text of the first SQL query in the group
Type	The type of the group. Currently, only the `Short` type is implemented

Above the table with a list of groups, the ***Cluster*** filter is located. You can use this filter to select the ADB cluster and its databases for which you want to display data in the table. Initially all databases of the default cluster are selected.



Filter by clusters and databases

Group details

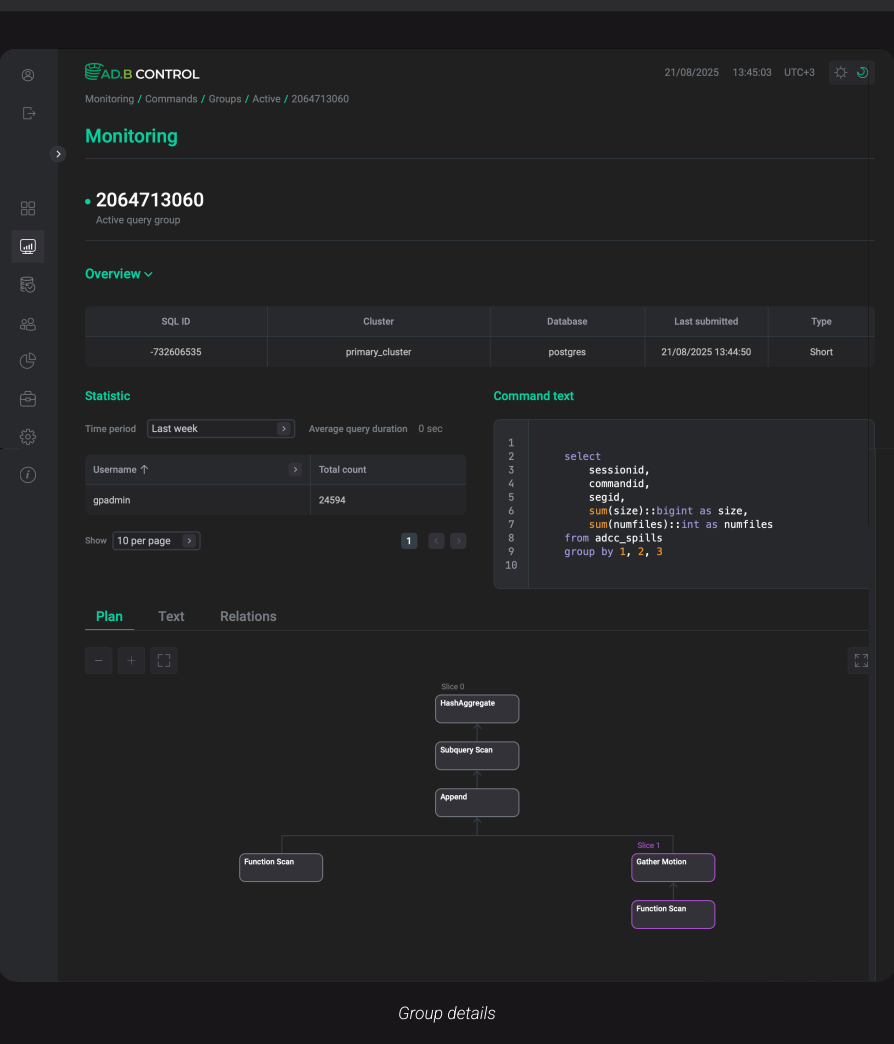
To view the group details, click the group identifier in the ***Query group ID*** column.

The page that opens contains the following sections:

- ***Overview*** displays the main parameters of the group described in the table above.
- ***Statistic*** shows the average query duration and the number of queries per user in the group. Use the ***Time period*** filter to show data for the last hour, last day, or last week.
- ***Command text*** shows the text of the first SQL query in the group.
- ***Plan***, ***Text***, and ***Relations*** — refer to Command details → Query plan sections for description of these tabs.

NOTE

Unlike the ***Queries*** tab, query groups don't show the progress and the statuses of nodes.



SQL ID	Cluster	Database	Last submitted	Type
-732906635	primary_cluster	postgres	21/08/2025 13:44:50	Short

Username	Total count
gpadmin	24594

```

select
    sessionid,
    commandid,
    segid,
    sum(size)::bigint as size,
    sum(numfiles)::int as numfiles
from adcc_spills
group by 1, 2, 3
```

Group details

SQL ID

An SQL ID is an identifier of an SQL command structure.

ADB Control uses SQL ID to identify commands with the same structure and to categorize them into groups. In ADB Control, SQL ID corresponds to the `queryid` value from the `pg_stat_statements` table in ADB.

Semantically equivalent SQL commands share the same SQL ID. In practice, this means the command properties described below will match.

Subqueries are analyzed recursively according to these same principles.

Factors affecting SQL ID

When the following SQL command properties differ, the commands will have separate SQL IDs.

Property | **Examples of commands with different SQL IDs**

Operation type: `SELECT`, `INSERT`, `UPDATE`, and `DELETE`

```
INSERT INTO employees (id, name) VALUES (1, 'John Doe');
SELECT id, name FROM employees;
```

Target relations

```
SELECT id, name FROM employees;
SELECT id, name FROM users;
```

```
INSERT INTO employees (id, name) VALUES (1, 'John Doe');
INSERT INTO people (id, name) VALUES (1, 'John Doe');
```

Affected columns including their order

```
SELECT id,name FROM users;
SELECT name,id FROM users;
```

Functions. Different functions result in different SQL IDs regardless of their arguments

```
SELECT DISTINCT UPPER (country) FROM users;
SELECT DISTINCT LOWER (country) FROM users;
```

Common table expressions. The structure and the order of CTEs if there are many. Different names result in different SQL IDs even if the content is identical

```
WITH my_filter AS (SELECT * FROM users WHERE active = true)
    SELECT * FROM my_filter;
WITH my_cte AS (SELECT * FROM users WHERE active = true)
    SELECT * FROM my_cte;
```

The `WHERE` clause. Separate SQL IDs are assigned when the columns and operators differ. The constants are ignored

```
SELECT * FROM users WHERE id > 5;
SELECT * FROM users WHERE id < 5;
SELECT * FROM users WHERE id BETWEEN 5 AND 10;
SELECT * FROM users WHERE age > 5;
```

`ORDER BY`, `GROUP BY`, `DISTINCT`, `DISTINCT ON`. Separate SQL IDs are assigned when the columns and comparison and sorting operators differ. The constants are ignored

```
SELECT * FROM users ORDER BY age;
SELECT * FROM users ORDER BY name;
SELECT * FROM users ORDER BY name DESC;
```

`JOIN` — the presence and the type of `JOIN` and the structure of the `JOIN` queries

```
SELECT * FROM orders o LEFT JOIN users u ON o.user_id = u.id;
SELECT * FROM orders o INNER JOIN users u ON o.user_id = u.id;
SELECT * FROM orders NATURAL JOIN users;
```

`HAVING` — operators, functions

```
SELECT user_id, COUNT(*) as order_count
FROM orders
GROUP BY user_id
HAVING COUNT(*) > 2;

SELECT user_id, SUM(amount) as total
FROM orders
GROUP BY user_id
HAVING SUM(amount) > 200;
```

Window functions — columns, functions, operators

```
SELECT id, user_id, amount,
       ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY amount ASC) as rn_asc
FROM orders;
SELECT id, user_id, amount,
       ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY amount DESC) as
rn_desc
FROM orders;
```

`UNION`, `INTERSECT`, and `EXCEPT` and the presence of `ALL` lead to separate SQL IDs, but `DISTINCT` is ignored

```
SELECT user_id FROM orders WHERE amount > 50
UNION
SELECT user_id FROM orders WHERE status = 'completed';

SELECT user_id FROM orders WHERE amount > 50
INTERSECT
SELECT user_id FROM orders WHERE status = 'completed';

SELECT user_id FROM orders WHERE amount > 50
EXCEPT
SELECT user_id FROM orders WHERE status = 'completed';
```

Factors not affecting SQL ID

If only the following properties differ in SQL command text, the commands will have the same SQL ID.

Property | **Examples of commands having the same SQL ID**

Aliases

```
SELECT id AS user_id, name AS full_name FROM users;
SELECT id AS client_id, name AS person_name FROM users;
```

Constants (literal values) in operators and functions

```
SELECT * FROM users WHERE id > 5;
SELECT * FROM users WHERE id > 10;
```

```
SELECT * FROM users WHERE active = true;
SELECT * FROM users WHERE active = false;
```

```
SELECT * FROM users LIMIT 5;
SELECT * FROM users LIMIT 10;
```

Line breaks, spaces, and tabs

```
SELECT id, name FROM users WHERE active = true;
SELECT id, name
FROM users
WHERE active = true;
```

`FOR [KEY] UPDATE/SHARE` locks

```
SELECT * FROM orders WHERE amount > 100 FOR UPDATE;
SELECT * FROM orders WHERE amount > 100 FOR SHARE;
SELECT * FROM orders WHERE amount > 100 FOR KEY SHARE;
```

All utility commands (those other than `SELECT`, `INSERT`, `UPDATE`, and `DELETE`) have the SQL ID of `0`.

If a utility command has a subquery that is not a utility command, that subquery will have a separate SQL ID

```
CREATE TABLE employees () INHERITS (users);
ALTER TABLE employees DROP COLUMN name;
VACUUM employees;
DROP TABLE employees;
```

Monitoring of transactions

Daria Barysheva

Contents

To Table of Contents

- List of transactions
  - Online
  - History
- Transaction details
  - Header
  - Overview
  - Performance
  - Commands
- Cancel or terminate the transaction
- System metrics collected for transactions

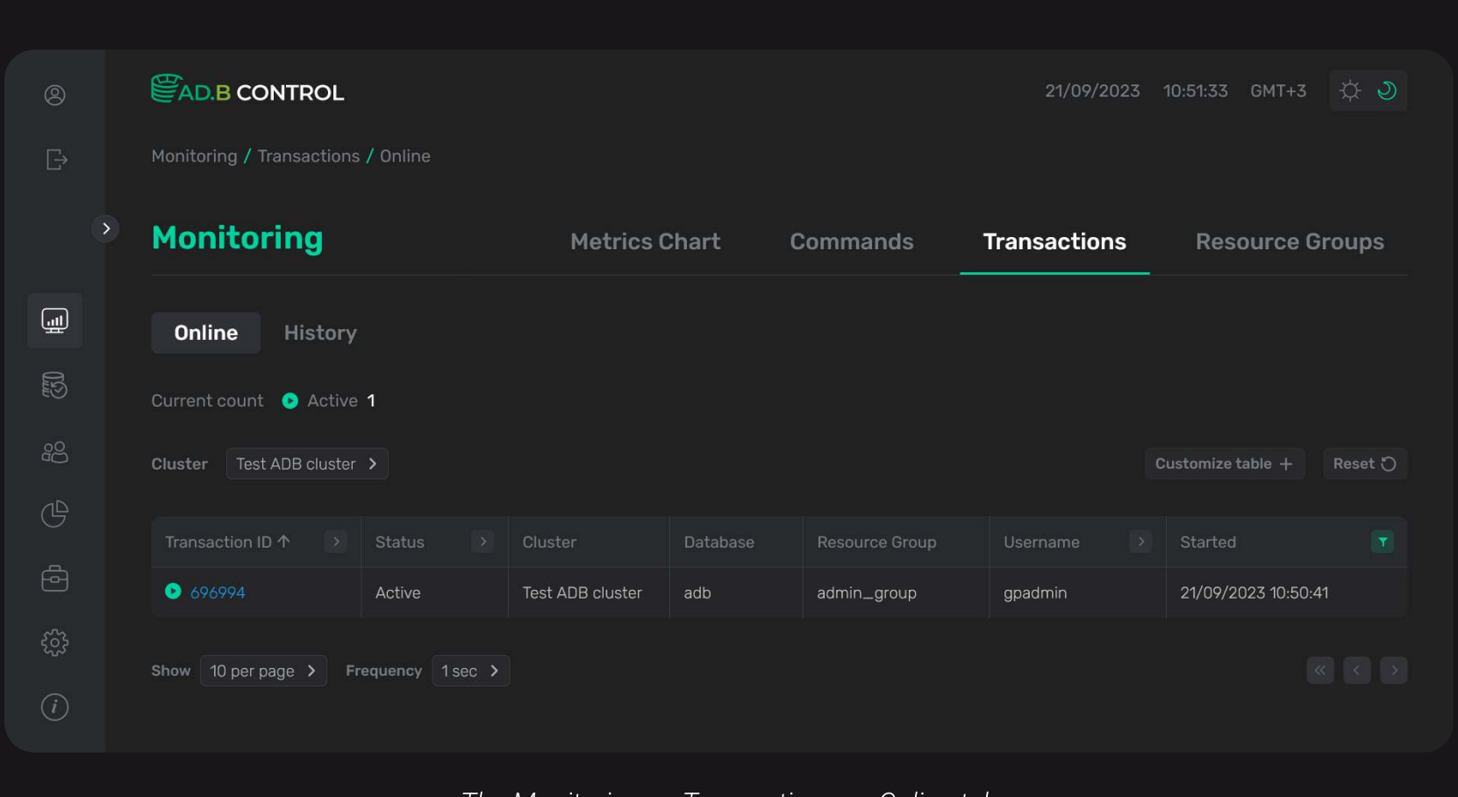
The ***Monitoring*** → ***Transactions*** page in the ADB Control web interface displays the transactions that are performed in the ADB clusters connected to the monitoring system. This page includes two tabs ***Online*** and ***History***, each of which is described in detail below.

List of transactions

Online

The ***Monitoring*** → ***Transactions*** → ***Online*** tab displays the currently running transactions (in the `Active` status).

At the top of the tab, the ***Current count*** field is located. It displays a total count of active transactions in ADB clusters connected to the monitoring system.



Current count: Active 1

Cluster: Test ADB cluster

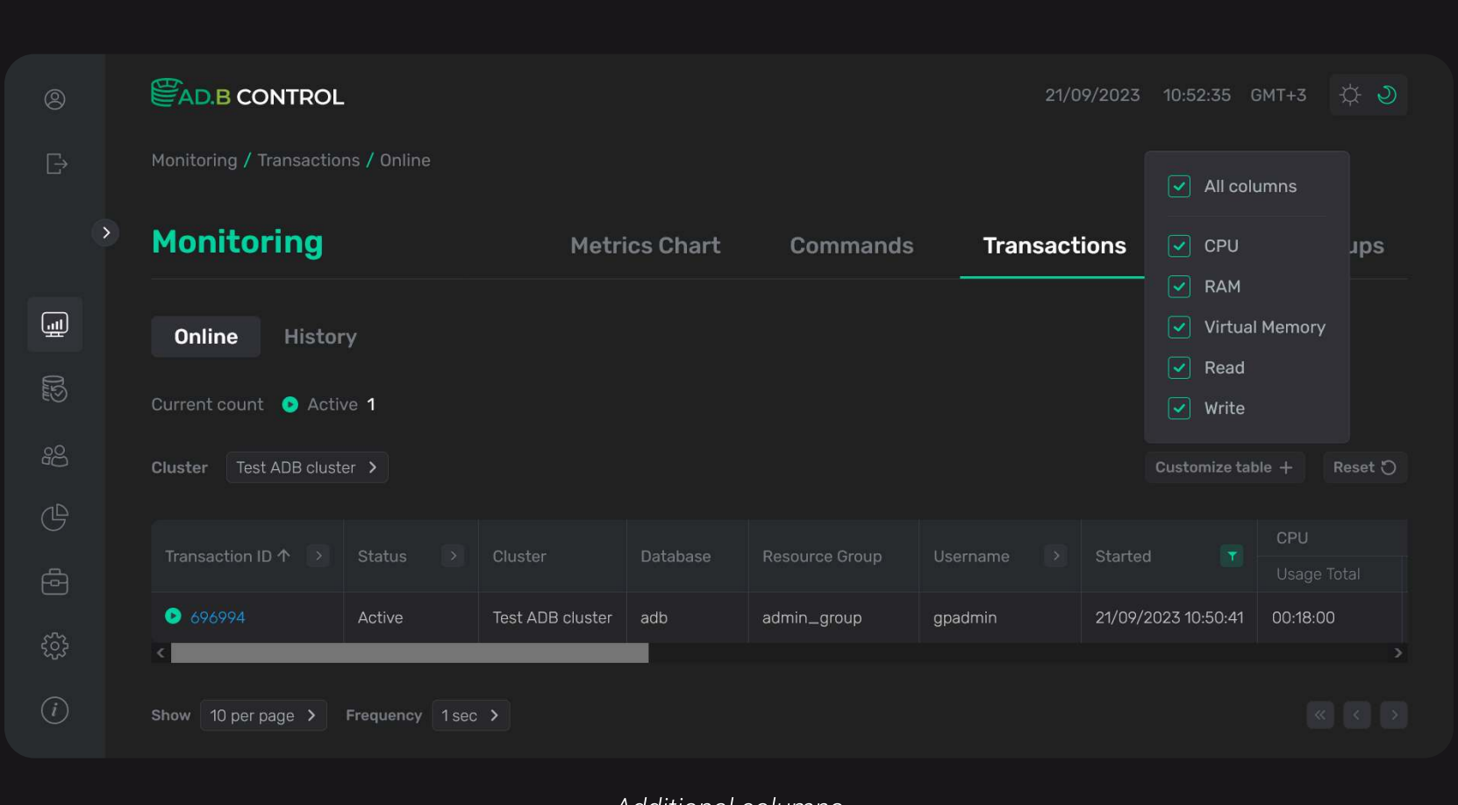
Transaction ID	Status	Cluster	Database	Resource Group	Username	Started
696994	Active	Test ADB cluster	adb	admin_group	gpadmin	21/09/2023 10:50:41

The Monitoring → Transactions → Online tab

On the ***Monitoring*** → ***Transactions*** → ***Online*** tab, there is a table with the following information on transactions.

Field	Description
Transaction ID	A unique transaction identifier
Status	A transaction status. Possible values are listed above
Cluster	A name of the ADB cluster where the transaction is launched
Database	A name of the database where the transaction is launched
Resource group	A name of the resource group that is used for the transaction
Username	A name of the user who started the transaction
Started	A transaction start timestamp in the `DD/MM/YYYY HH:mm:ss` format
Tags	The transaction tags

If necessary, you can add columns to the table. To do this, click ***Customize table*** and select column names in the section that opens. Currently you can add system metrics that are described below.



☑ All columns
☑ CPU
☑ RAM
☑ Virtual Memory
☑ Read
☑ Write

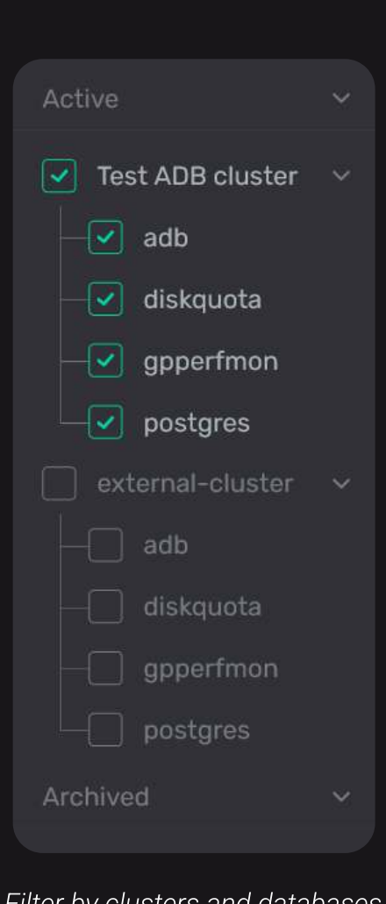
Current count: Active 1

Cluster: Test ADB cluster

Transaction ID	Status	Cluster	Database	Resource Group	Username	Started	CPU Usage Total
696994	Active	Test ADB cluster	adb	admin_group	gpadmin	21/09/2023 10:50:41	00:18:00

Additional columns

Above the table with a list of transactions, the ***Cluster*** filter is located. You can use this filter to select the ADB cluster and its databases for which you want to display data in the table. Initially all databases of the default cluster are selected.


Active
- ☑ Test ADB cluster
  - ☑ adb
  - ☑ diskquota
  - ☑ gpperfmon
  - ☑ postgres
- ☐ external-cluster
  - ☐ adb
  - ☐ diskquota
  - ☐ gpperfmon
  - ☐ postgres

Archived

Filter by clusters and databases

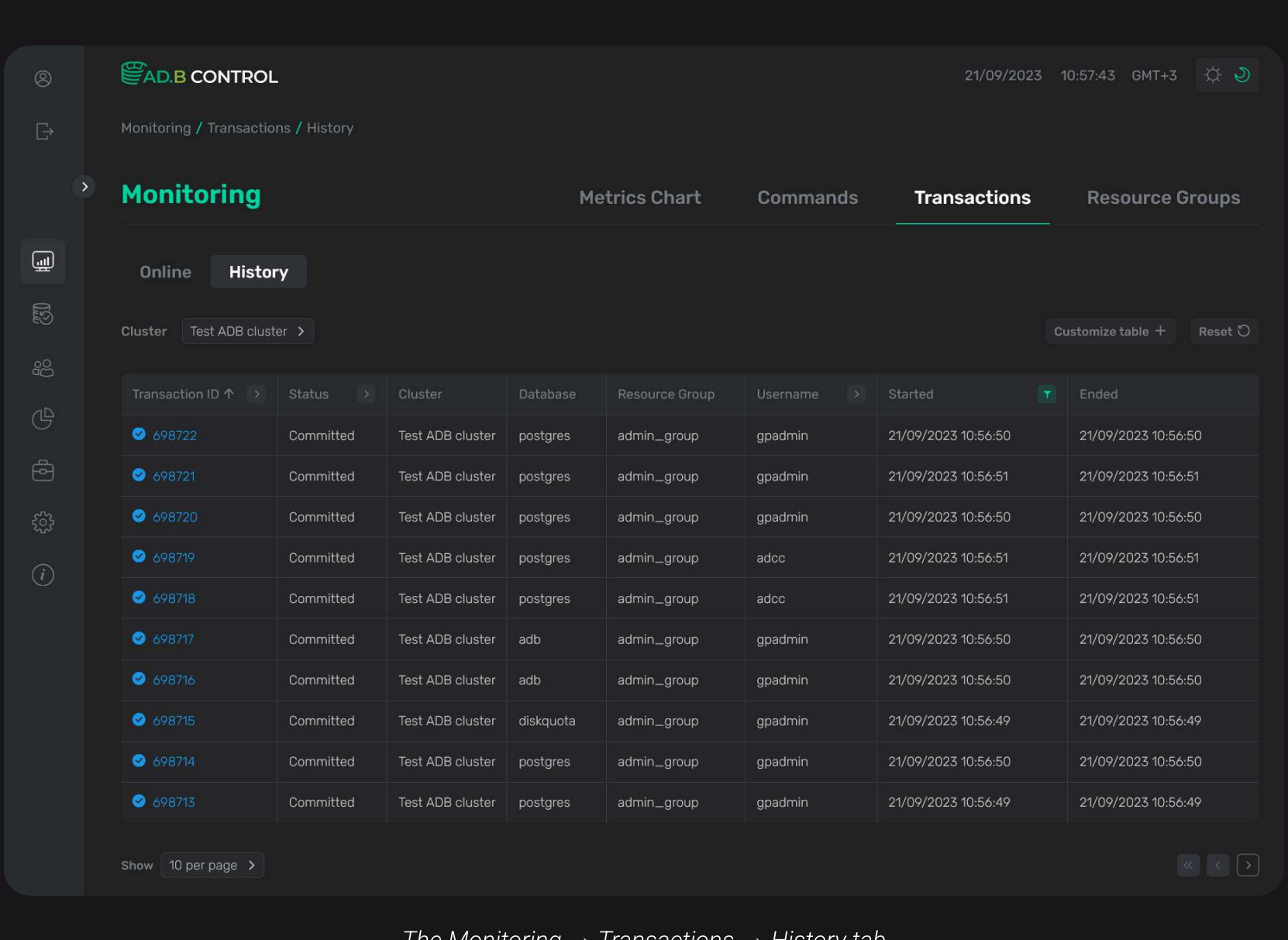
In the column headers of the table with a list of transactions, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Username***), the search value should be entered. For columns that show date and time (e.g. ***Started***), the time range can be selected from the calendar.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

History

The ***Monitoring*** → ***Transactions*** → ***History*** tab displays transactions in the following statuses:

- `Committed` — successfully completed transactions.
- `Aborted` — cancelled/terminated transactions.
- `Unknown` — the transactions, the final status of which is undefined. It is set on a timeout for transactions in the `Active` status, for which there are no corresponding rows in the `pg_stat_activity` view.



Cluster: Test ADB cluster

Transaction ID	Status	Cluster	Database	Resource Group	Username	Started	Ended
698722	Committed	Test ADB cluster	postgres	admin_group	gpadmin	21/09/2023 10:56:50	21/09/2023 10:56:50
698721	Committed	Test ADB cluster	postgres	admin_group	gpadmin	21/09/2023 10:56:51	21/09/2023 10:56:51
698720	Committed	Test ADB cluster	postgres	admin_group	gpadmin	21/09/2023 10:56:50	21/09/2023 10:56:50
698719	Committed	Test ADB cluster	postgres	admin_group	adcc	21/09/2023 10:56:51	21/09/2023 10:56:51
698718	Committed	Test ADB cluster	postgres	admin_group	adcc	21/09/2023 10:56:51	21/09/2023 10:56:51
698717	Committed	Test ADB cluster	adb	admin_group	gpadmin	21/09/2023 10:56:50	21/09/2023 10:56:50
698716	Committed	Test ADB cluster	adb	admin_group	gpadmin	21/09/2023 10:56:50	21/09/2023 10:56:50
698715	Committed	Test ADB cluster	diskquota	admin_group	gpadmin	21/09/2023 10:56:49	21/09/2023 10:56:49
698714	Committed	Test ADB cluster	postgres	admin_group	gpadmin	21/09/2023 10:56:50	21/09/2023 10:56:50
698713	Committed	Test ADB cluster	postgres	admin_group	gpadmin	21/09/2023 10:56:49	21/09/2023 10:56:49

The Monitoring → Transactions → History tab

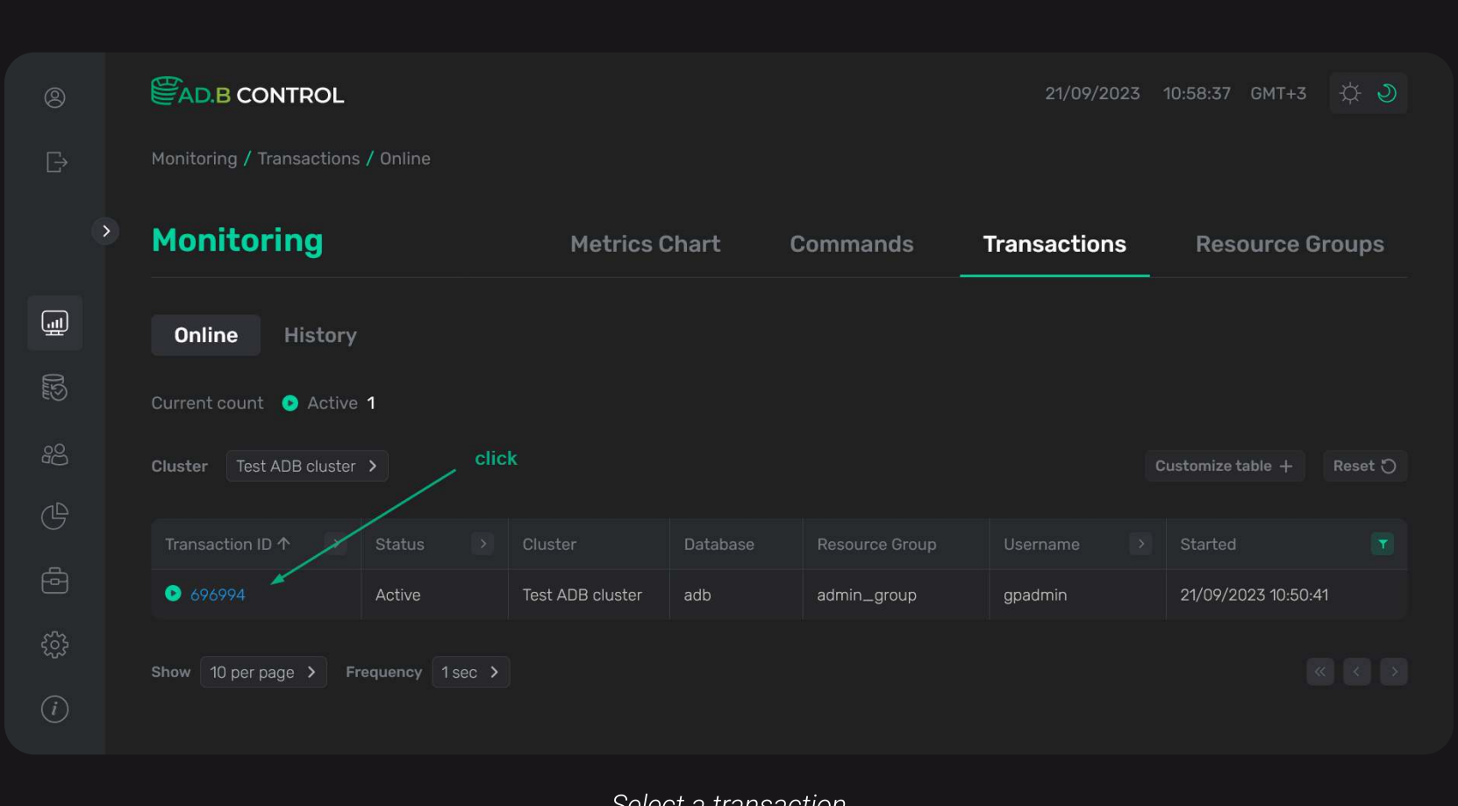
Most of the fields in the table with the list of transactions on the ***History*** tab match the fields that are described above for the ***Online*** tab. Only one new field is enabled — ***Ended***, which displays a transaction end timestamp in the `DD/MM/YYYY HH:mm:ss` format. The field is filled in both as a result of successful execution and in case of error or cancellation.

Like on the ***Online*** tab, you have the ability to add additional columns to the table by clicking ***Customize table***.

Above the table with a list of transactions, the ***Cluster*** filter is located (see the description above).

Transaction details

To view the transaction details, click the transaction identifier (***Transaction ID***) in the table on the Monitoring → Transactions → Online or Monitoring → Transactions → History tab.



Current count: Active 1

Cluster: Test ADB cluster

Transaction ID	Status	Cluster	Database	Resource Group	Username	Started
696994	Active	Test ADB cluster	adb	admin_group	gpadmin	21/09/2023 10:50:41

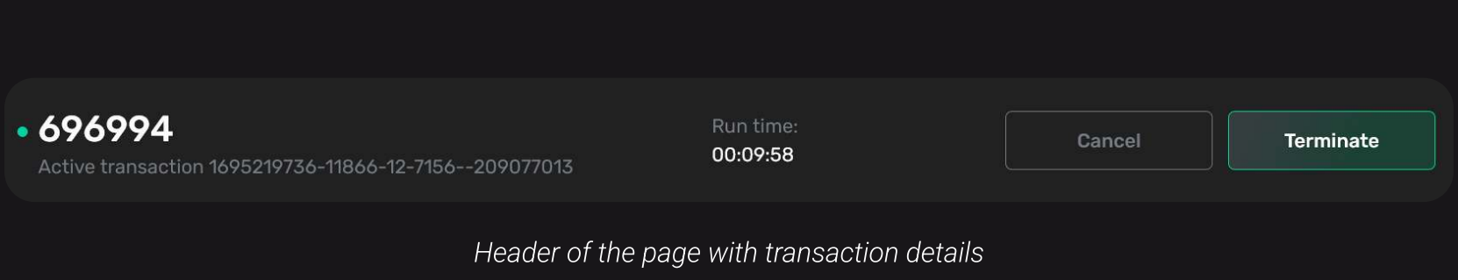
Select a transaction

The following page contains multiple sections that are described below.

Header

At the top of the page, the following information is displayed:

- The transaction unique identifier (see ***Transaction ID*** above). In the example below — `696994`.
- The transaction status (see ***Status*** above). In the example below — `Active`.
- The complex transaction identifier (in the example below — `1695219736-11866-12-7156-209077013`), which includes:
  - cluster start timestamp;
  - session number;
  - transaction number within the current session;
  - the `lxid` transaction identifier on coordinator;
  - hash code based on the cluster name (can be negative).
- ***Run time*** — the transaction duration in hours, minutes, seconds.
- The ***Cancel*** and ***Terminate*** buttons, which you can use to cancel/terminate transactions. For more information, see Cancel or terminate a transaction.



696994

Active transaction 1695219736-11866-12-7156--209077013

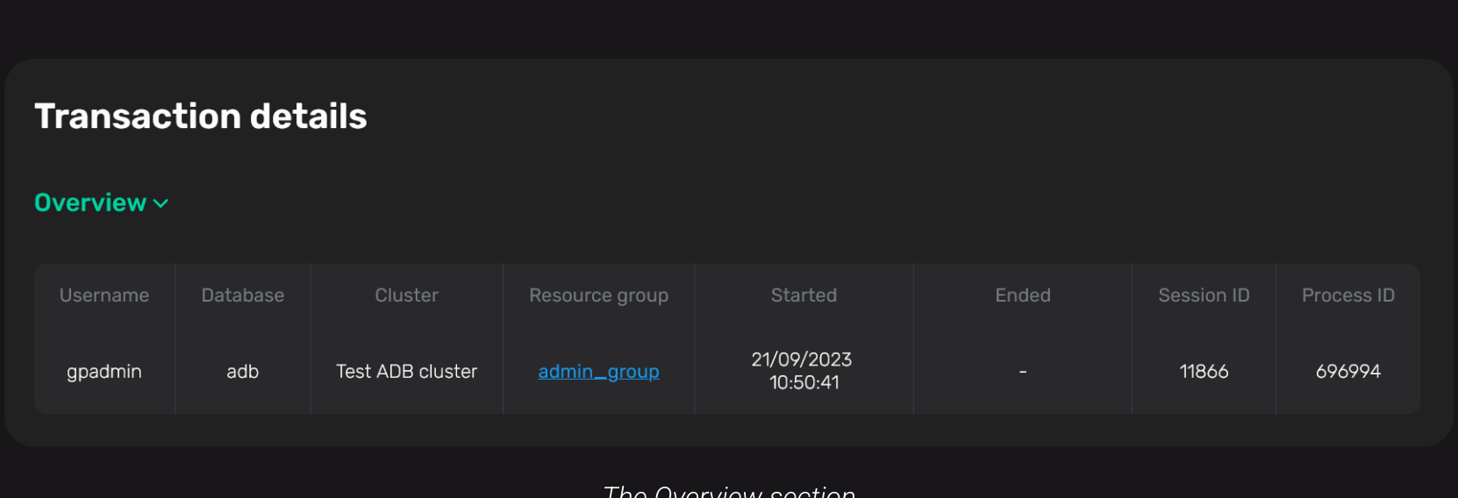
Run time: 00:09:58

Header of the page with transaction details

Overview

The ***Overview*** section displays the general transaction information. In addition to the fields mentioned above, the following information is available:

- ***Ended*** — the transaction end timestamp in the `DD/MM/YYYY HH:mm:ss` format. For active transactions, the field contains `-`.
- ***Session ID*** — an identifier of the session, within which the transaction was started.
- ***Process ID*** — an identifier of the process, within which the transaction was started.


Transaction details

Overview

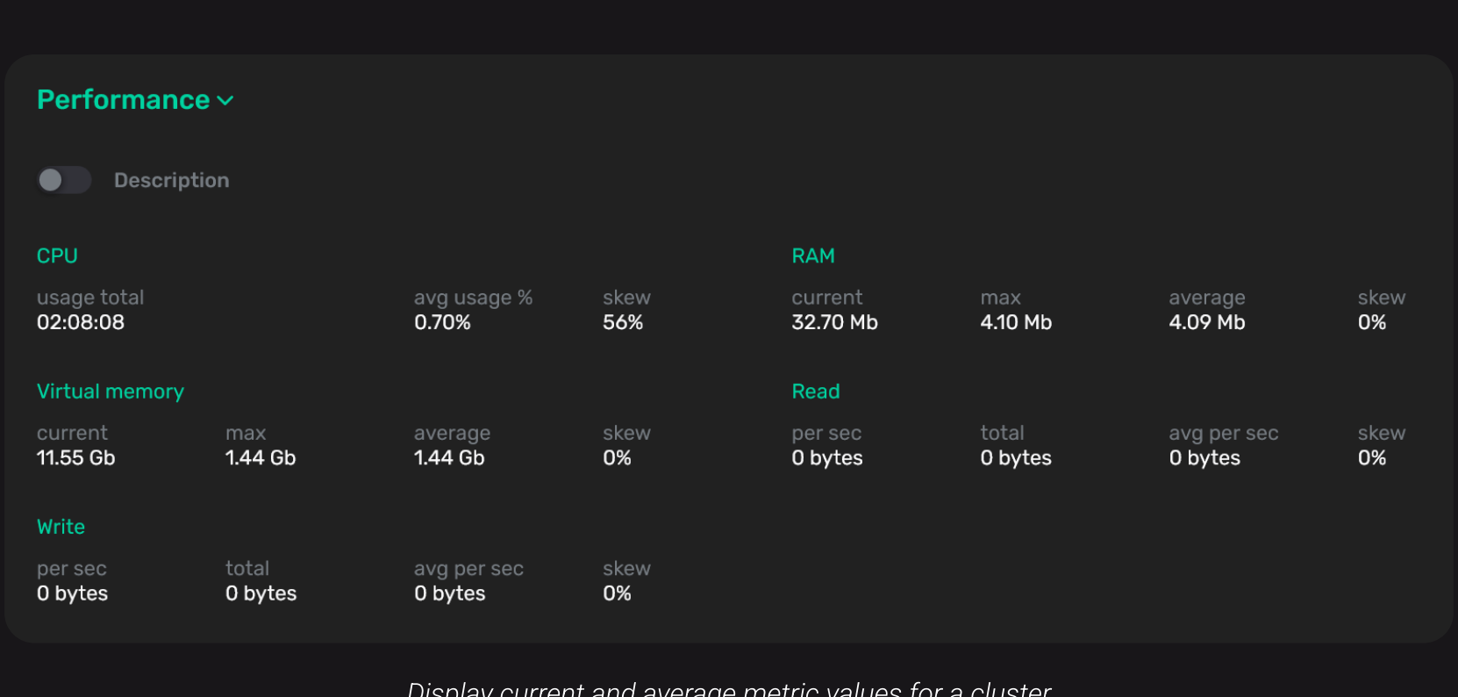
Username	Database	Cluster	Resource group	Started	Ended	Session ID	Process ID
gpadmin	adb	Test ADB cluster	admin_group	21/09/2023 10:50:41	-	11866	696994

The Overview section

Performance

The ***Performance*** section displays the statistics of system resource consumption by the selected transaction. The metrics available for monitoring are listed in System metrics collected for transactions.

At the top of the ***Performance*** section, the current and average metric values with skews for all cluster segments are displayed.



CPU: usage total 02:08:08; avg usage % 0.70%; skew 56%

RAM: current 32.70 Mb; max 4.10 Mb; average 4.09 Mb; skew 0%

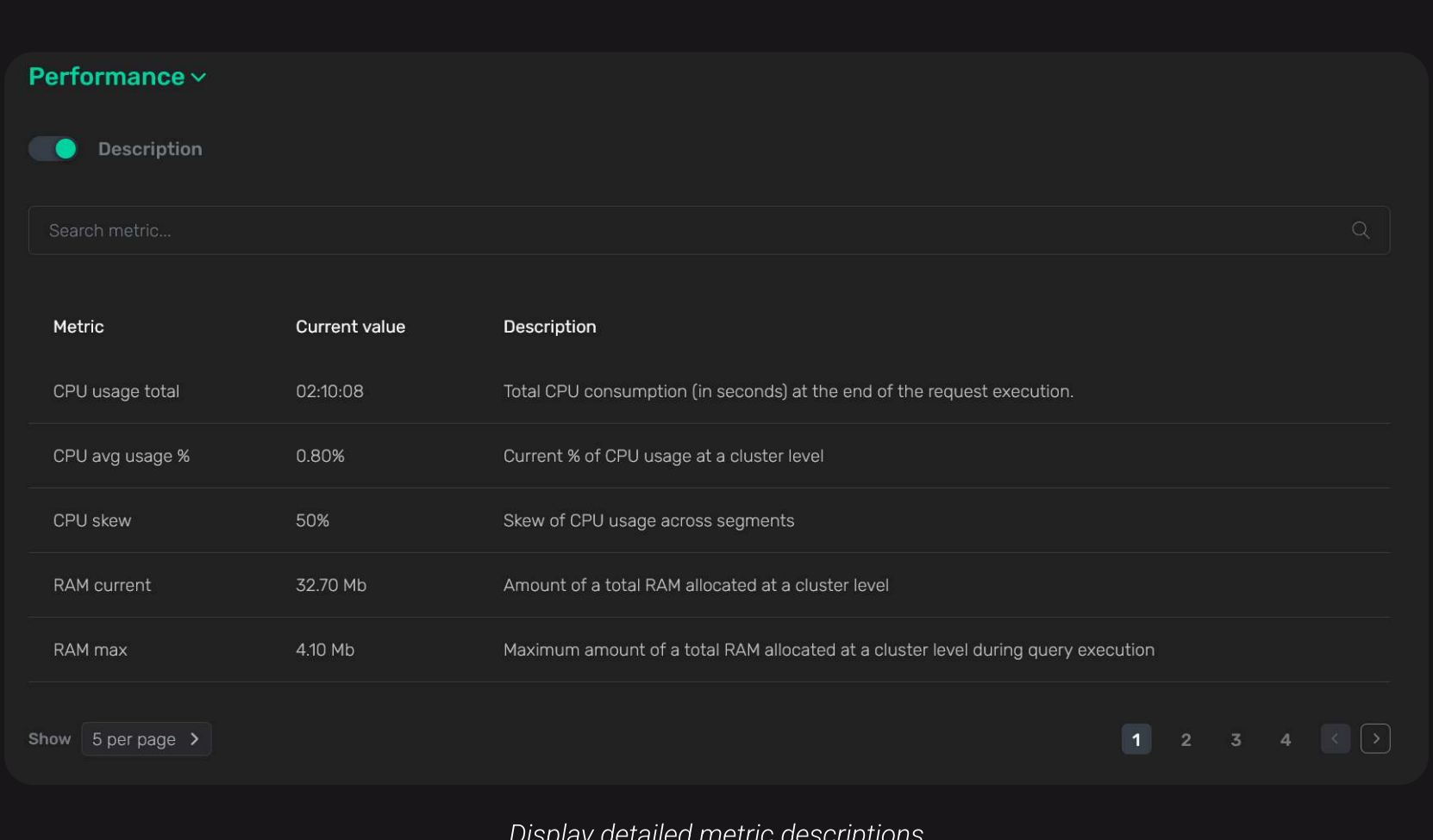
Virtual memory: current 11.55 Gb; max 1.44 Gb; average 1.44 Gb; skew 0%

Read: per sec 0 bytes; total 0 bytes; avg per sec 0 bytes; skew 0%

Write: per sec 0 bytes; total 0 bytes; avg per sec 0 bytes; skew 0%

Display current and average metric values for a cluster

You can select an alternative display of metrics by activating the switcher that is located under the ***Performance*** section. The result is a table where not only the current value but also a detailed description is displayed for each metric.



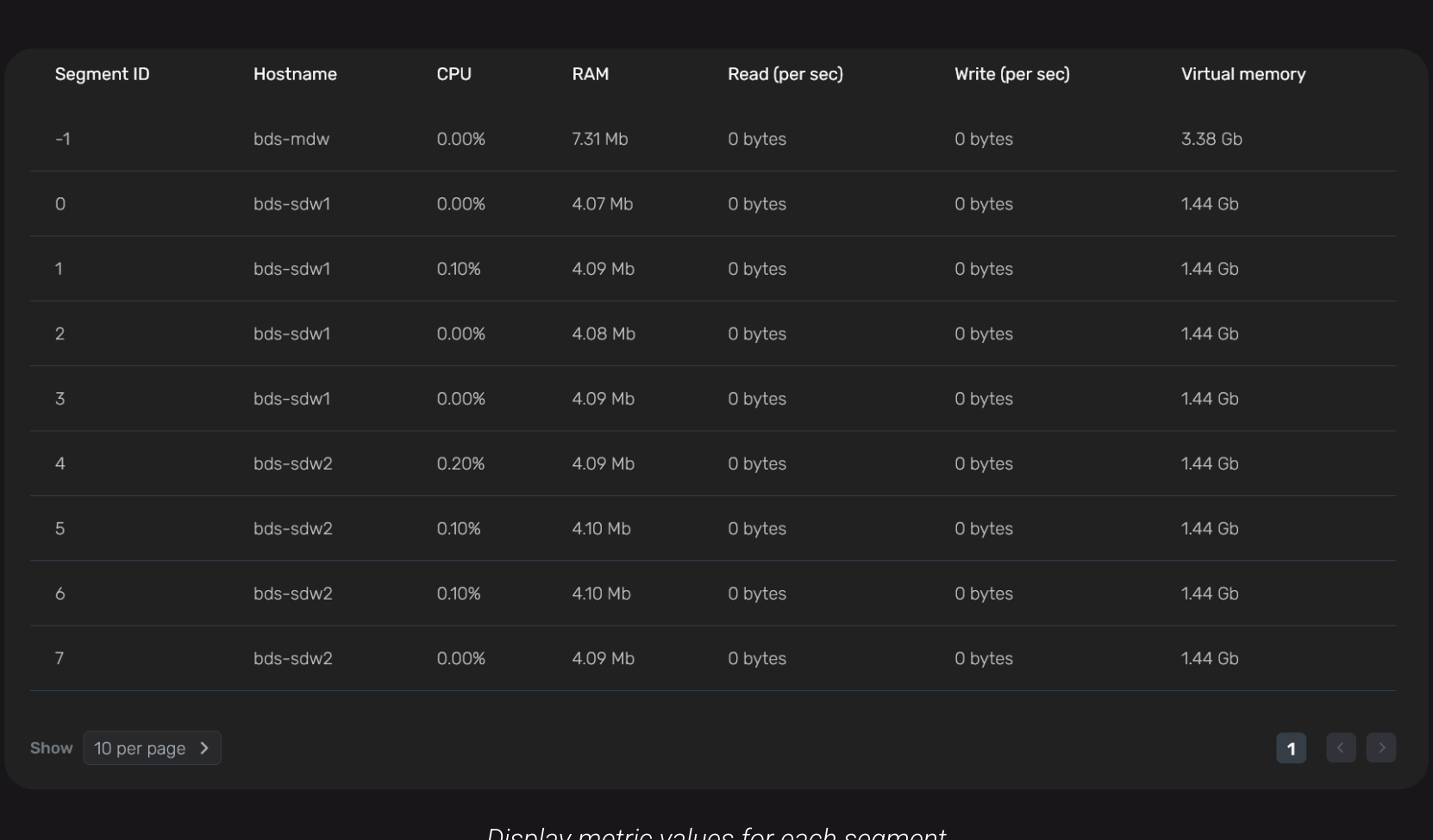
Metric	Current value	Description
CPU usage total	02:10:08	Total CPU consumption (in seconds) at the end of the request execution.
CPU avg usage %	0.80%	Current % of CPU usage at a cluster level
CPU skew	50%	Skew of CPU usage across segments
RAM current	32.70 Mb	Amount of a total RAM allocated at a cluster level
RAM max	4.10 Mb	Maximum amount of a total RAM allocated at a cluster level during query execution

Display detailed metric descriptions

In the bottom part of the ***Performance*** section, the metric values are shown separately for each cluster segment, including the coordinator segment `seg-1` (where ***Segment ID*** equals to `-1`).

Fields of the table with segment metrics

Field	Description
Segment ID	A content identifier for a segment instance. Corresponds to `gp_segment_configuration.content`. For more information, see the ***System catalog tables → gp_segment_configuration*** section of the Tables overview article
Hostname	A host name
CPU	The amount of CPU consumed by the segment (percentage). Based on the metric values for all segments, the average value CPU avg usage % is calculated
RAM	The amount of RAM consumed by the segment (in bytes). Based on the metric values for all segments, the average value RAM average is calculated
Read (per sec)	The amount of data read per second by the segment (in bytes). Based on the metric values for all segments, the average value Read avg per sec is calculated
Write (per sec)	The amount of data written per second by the segment (in bytes). Based on the metric values for all segments, the average value Write avg per sec is calculated
Virtual memory	The amount of virtual memory allocated by the segment (in bytes). Based on the metric values for all segments, the average value Virtual memory average is calculated



Segment ID	Hostname	CPU	RAM	Read (per sec)	Write (per sec)	Virtual memory
-1	bds-mdw	0.00%	7.31 Mb	0 bytes	0 bytes	3.38 Gb
0	bds-sdw1	0.00%	4.07 Mb	0 bytes	0 bytes	1.44 Gb
1	bds-sdw1	0.10%	4.09 Mb	0 bytes	0 bytes	1.44 Gb
2	bds-sdw1	0.00%	4.08 Mb	0 bytes	0 bytes	1.44 Gb
3	bds-sdw1	0.00%	4.09 Mb	0 bytes	0 bytes	1.44 Gb
4	bds-sdw2	0.20%	4.09 Mb	0 bytes	0 bytes	1.44 Gb
5	bds-sdw2	0.10%	4.10 Mb	0 bytes	0 bytes	1.44 Gb
6	bds-sdw2	0.10%	4.10 Mb	0 bytes	0 bytes	1.44 Gb
7	bds-sdw2	0.00%	4.09 Mb	0 bytes	0 bytes	1.44 Gb

Display metric values for each segment

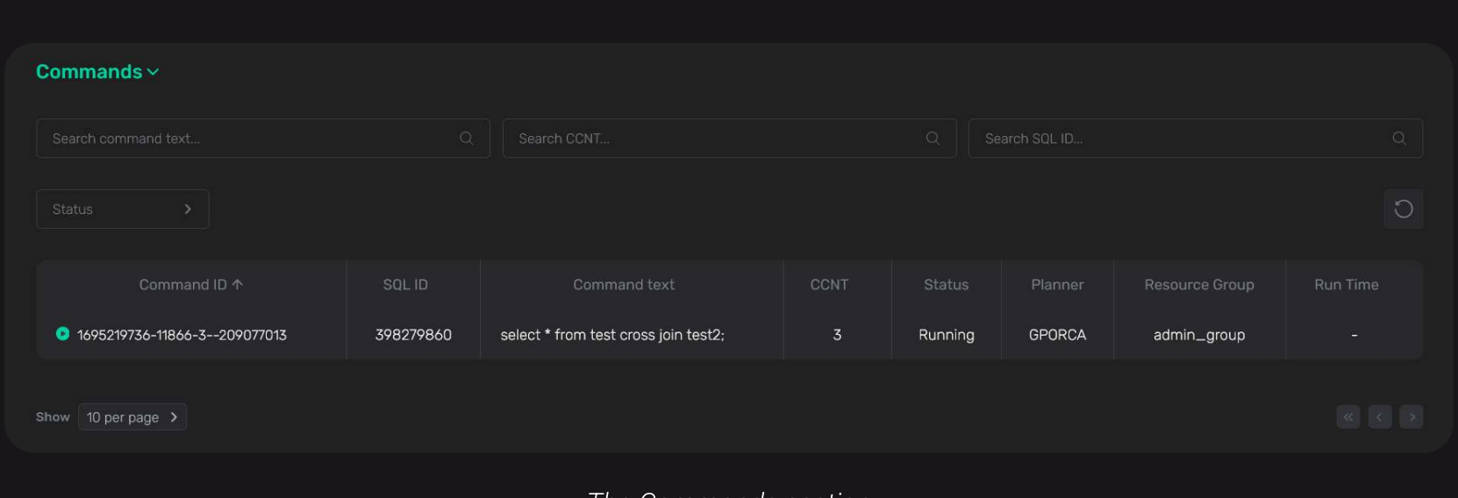
> **IMPORTANT**
>
> System metrics are collected from ADB segments by ADB Control agents only for transactions longer than 15 seconds. Due to this fact, the section with metrics will be empty for the transactions that complete faster.

Commands

The ***Commands*** section lists the commands that are performed within the current transaction.

The "Commands" tab fields

Field	Description
Command ID	A unique command identifier, which includes: - cluster start timestamp; - session number; - command number within the current session; - hash code based on the cluster name (can be negative). To open the page with command details, click the command identifier
SQL ID	A common identifier for SQL commands with the same structure
Command text	The first symbols of a command text. To view the full text (in case of long queries), hover the mouse over the field value
CCNT	A command number within the current session
Status	A command status. Possible values: - `Queued` — the commands on which information about the actual start of execution has not yet received. - `Running` — the commands that are currently running. - `Cancelling` — the commands that are currently cancelling (as a part of the transaction cancellation or termination). - `Done` — successfully completed commands. - `Cancelled` — cancelled commands (as a part of the transaction cancellation or termination). - `Error` — the commands during which some errors occurred. - `Unknown` — the commands, the final status of which is undefined. It is set on a timeout for queries in the `Queued`, `Running`, `Cancelling` statuses, for which there are no corresponding rows in the `pg_stat_activity` view.
Planner	A name of the query optimizer that is used to produce the query execution plan. Possible values: - `GPORCA` — GPORCA is used (`optimizer = on`). - `Fallbacked` — after the attempt to produce a plan via GPORCA (`optimizer = on`) failed, Postgres query optimizer is used. - `Legacy` — Postgres query optimizer is used (`optimizer = off`).
Resource group	A name of the resource group that is used for the command
Run time	A command duration in hours, minutes, seconds



Command ID	SQL ID	Command text	CCNT	Status	Planner	Resource Group	Run Time
1695219736-11866-3--209077013	398279860	select * from test cross join test2;	3	Running	GPORCA	admin_group	-

The Commands section

Above the table with commands, there are filters that you can use to select specific data:

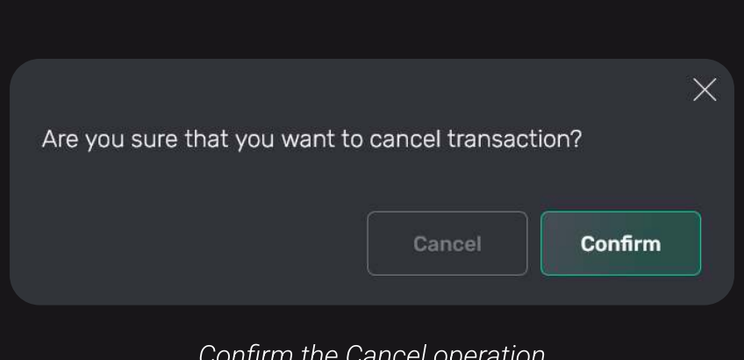
- ***Search command text...*** — filter by the command text (see ***Command text*** above). You can search by any substring of the SQL command.
- ***Search CCNT...*** — filter by the command number within a session (see ***CCNT*** above). Enter a full value.
- ***Search SQL ID...*** — filter by the identifier that is common for SQL commands with the same structure (see ***SQL ID*** above). Enter a full value.
- ***Status*** — filter by the command status (see ***Status*** above). Select a value from the drop-down list.

Cancel or terminate the transaction

On the page with transaction details, you can cancel a transaction by clicking one of the following buttons:

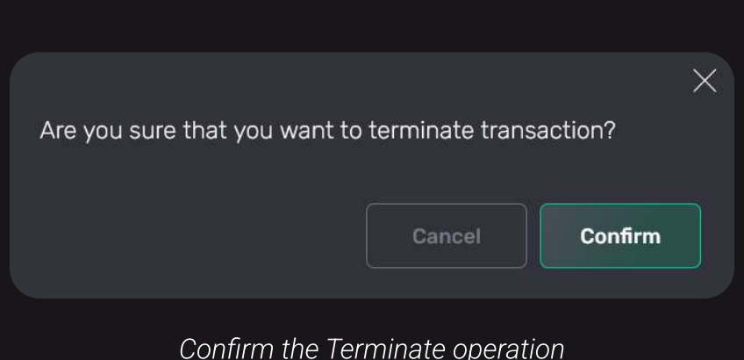
- ***Cancel*** — calls the `pg_cancel_backend` function, which cancels a transaction in the process.
- ***Terminate*** — calls the `pg_terminate_backend` function, which terminates the process within which the transaction is running.

After clicking the button, confirm the operation.



Are you sure that you want to cancel transaction?

Confirm the Cancel operation



Are you sure that you want to terminate transaction?

Confirm the Terminate operation

After a transaction is cancelled or terminated, it acquires the `Aborted` status, whereas all transaction commands get the `Cancelled` status.

> **IMPORTANT**
>
> - Cancellation and termination of transactions is available for ADB Control users with appropriate permissions (see `Kill self queries`, `Kill some queries`, `Kill all queries` in the Authorization article).
> - You can cancel and terminate only active transactions. For transactions that are opened on the History tab, ***Cancel*** and ***Terminate*** buttons are disabled.
> - Cancellation and termination of transactions are recorded as a part of the operation audit under the operation names `Cancel` and `Terminate`, respectively (for the `Transaction` object type). You can view audit events on the Audit → Operations tab.

System metrics collected for transactions

Both Online and History tabs allow you to view the statistics of the system resource consumption by transactions (on the Performance tab in the transaction details). The metrics available for transactions are listed below.

> **NOTE**
>
> The ***Online*** tab displays current metric values, whereas the ***History*** tab shows metric values at the end of the transaction execution.

Transaction metrics

Group	Metric	Description
CPU	CPU usage total	The total CPU consumption across cluster segments (in seconds)
	CPU avg usage %	The CPU percent average for all segment processes executing the query. Calculated as the result of dividing the total CPU consumption on all cluster segments (excluding the coordinator and standby) by the total number of processes processing the query on segment hosts (excluding the coordinator and standby)
	CPU skew	The indicator that displays the CPU consumption skew across cluster segments (percentage). A value other than zero indicates that one of the segments uses more CPU than others
RAM	RAM current	The current amount of RAM consumed at the cluster level (in bytes)
	RAM max	The maximum amount of RAM consumed at the cluster level during the transaction execution (in bytes)
	RAM average	The average amount of RAM consumed across cluster segments (in bytes). The RAM consumption at any given moment is calculated based on the operating system metric **rss (Resident Set Size)**
	RAM skew	The indicator that displays the RAM consumption skew across cluster segments (percentage). A value other than zero indicates that one of the segments uses more RAM than others
Virtual memory	Virtual memory current	The current amount of virtual memory allocated at the cluster level (in bytes)
	Virtual memory max	The maximum amount of virtual memory allocated at the cluster level during the transaction execution (in bytes)
	Virtual memory average	The average amount of virtual memory allocated across cluster segments (in bytes). The virtual memory consumption at any given moment is calculated based on the operating system metric **vsize (Virtual Memory Size)**
	Virtual memory skew	The indicator that displays the virtual memory skew across cluster segments (percentage). A value other than zero indicates that one of the segments uses more virtual memory than others
Read	Read per sec	The current amount of data read per second at the cluster level (in bytes)
	Read total	The total amount of data read across cluster segments (in bytes). The following formula is used for calculation: $$ReadTotal = TotalReadSpeed \cdot QueryTime$$
	Read avg per sec	The average amount of data read across cluster segments per second (in bytes). The data read rate at any given moment is calculated based on the I/O Read information from */proc/[pid]/stat* as a delta between the current and previous values
	Read skew	The indicator that displays the data reading skew across cluster segments (percentage). A value other than zero indicates that one of the segments reads more data from a disk than others
Write	Write per sec	The current amount of data written per second at the cluster level (in bytes)
	Write total	The total amount of data written across cluster segments (in bytes). The following formula is used for calculation: $$WriteTotal = TotalWriteSpeed \cdot QueryTime$$
	Write avg per sec	The average amount of data written across cluster segments per second (in bytes). The data write rate at any given moment is calculated based on the I/O Write information from */proc/[pid]/stat* as a delta between the current and previous values
	Write skew	The indicator that displays the data writing skew across cluster segments (percentage). A value other than zero indicates that one of the segments writes more data to a disk than others

Monitoring of sessions

Daria Barysheva

Contents

To Table of Contents

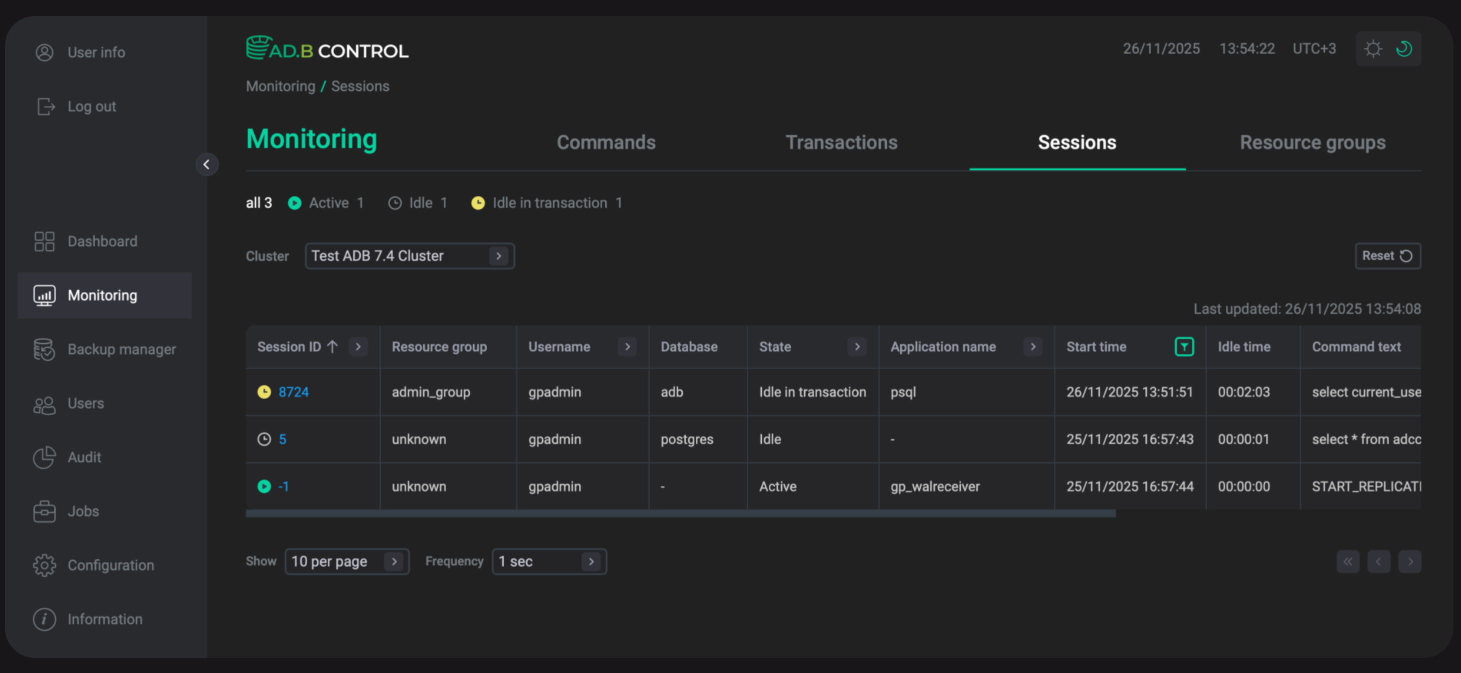
View a list of sessions

Terminate a session

The *Monitoring* → *Sessions* tab is designed to monitor current ADB sessions, both active and idle. This page also allows you to terminate specified sessions in order to release the connection pool for further connections.

View a list of sessions

On the *Monitoring* → *Sessions* tab, there is a table with the following information on sessions.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/11/2025 13:54:22 UTC+3
Monitoring / Sessions
Monitoring
Commands
Transactions
Sessions
Resource groups
all 3 Active 1 Idle 1 Idle in transaction 1
Cluster Test ADB 7.4 Cluster
Reset
Last updated: 26/11/2025 13:54:08
Show 10 per page Frequency 1 sec

Session ID	Resource group	Username	Database	State	Application name	Start time	Idle time	Command text
8724	admin_group	gpadmin	adb	Idle in transaction	psql	26/11/2025 13:51:51	00:02:03	select current_use
5	unknown	gpadmin	postgres	Idle	-	25/11/2025 16:57:43	00:00:01	select * from adcc
-1	unknown	gpadmin	-	Active	gp_walreceiver	25/11/2025 16:57:44	00:00:00	START_REPLICATI

List of sessions

The "Monitoring → Sessions" tab fields

Field	Description
Session ID	A session identifier
Resource group	A resource group name
Username	A name of the user who connected to the backend
Database	A database name
State	A session state. Possible values: ■ `Active` — the backend is running a query. ■ `Idle` — the backend is waiting for a new client command. ■ `Idle in transaction` — the backend is in a transaction, but there are no currently running queries.
Application name	A name of the application that started a session
Start time	The time when the backend process was started
Idle time	An idle time (in seconds)
Command text	A text of the most recent SQL query that was run at the backend (within the current session). If `state` is `active`, the field shows the currently running query
Tags	The session tags

NOTE

To view the details of the most recent command within the current session, click the session identifier (*Session ID*). Note that you can view the command details if this command is registered in ADB Control. For more information on necessary requirements, see Monitoring of commands.

At the top of the tab, the *Active*, *Idle*, and *Idle in transaction* fields are located. They display a total count of sessions in the specified statuses.

Above the table with a list of sessions, the *Cluster* filter is located. You can use this filter to select the ADB cluster for which you want to display data in the table. The default cluster for all such filters is defined in the ADB Control settings.

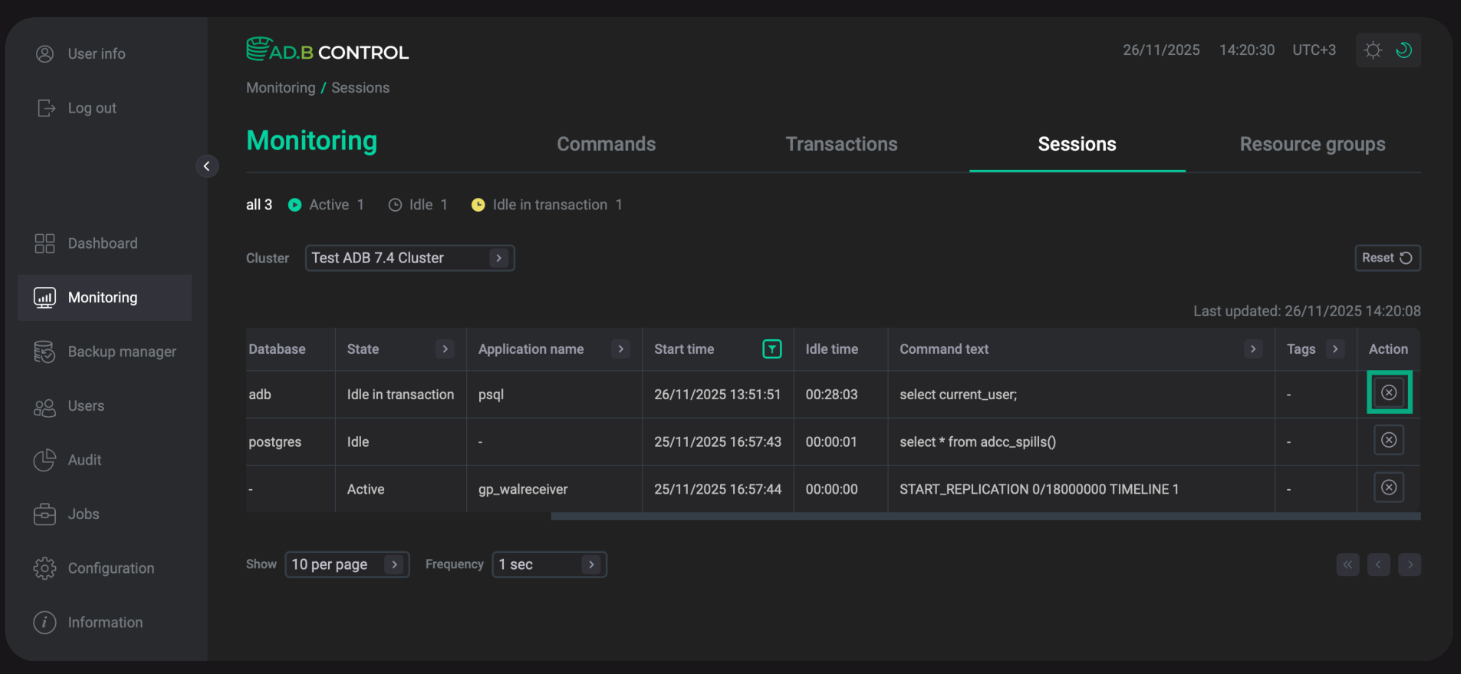
In the column headers of the table with a list of sessions, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. *State*), you can select a value from the drop-down list. For some columns (e.g. *Session ID*), the search value should be entered. For columns that show date and time (e.g. *Start time*), the time range can be selected from the calendar.

The ⏷ icon means that a filter is defined for the column. To reset all filters, click *Reset*.

Terminate a session

To terminate a session, follow the steps:

1. Click the ⊗ icon in the *Actions* column on the *Monitoring* → *Sessions* tab.

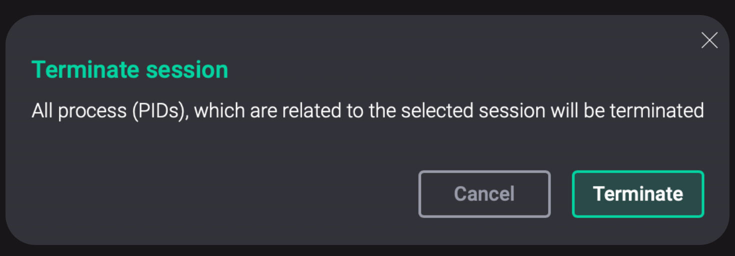


User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/11/2025 14:20:30 UTC+3
Monitoring / Sessions
Monitoring
Commands
Transactions
Sessions
Resource groups
all 3 Active 1 Idle 1 Idle in transaction 1
Cluster Test ADB 7.4 Cluster
Reset
Last updated: 26/11/2025 14:20:08
Show 10 per page Frequency 1 sec

Database	State	Application name	Start time	Idle time	Command text	Tags	Action
adb	Idle in transaction	psql	26/11/2025 13:51:51	00:28:03	select current_user;	-	⊗
postgres	Idle	-	25/11/2025 16:57:43	00:00:01	select * from adcc_spills()	-	⊗
-	Active	gp_walreceiver	25/11/2025 16:57:44	00:00:00	START_REPLICATION 0/18000000 TIMELINE 1	-	⊗

Switch to terminating a session

2. In the window that opens, confirm the operation by clicking *Terminate*.



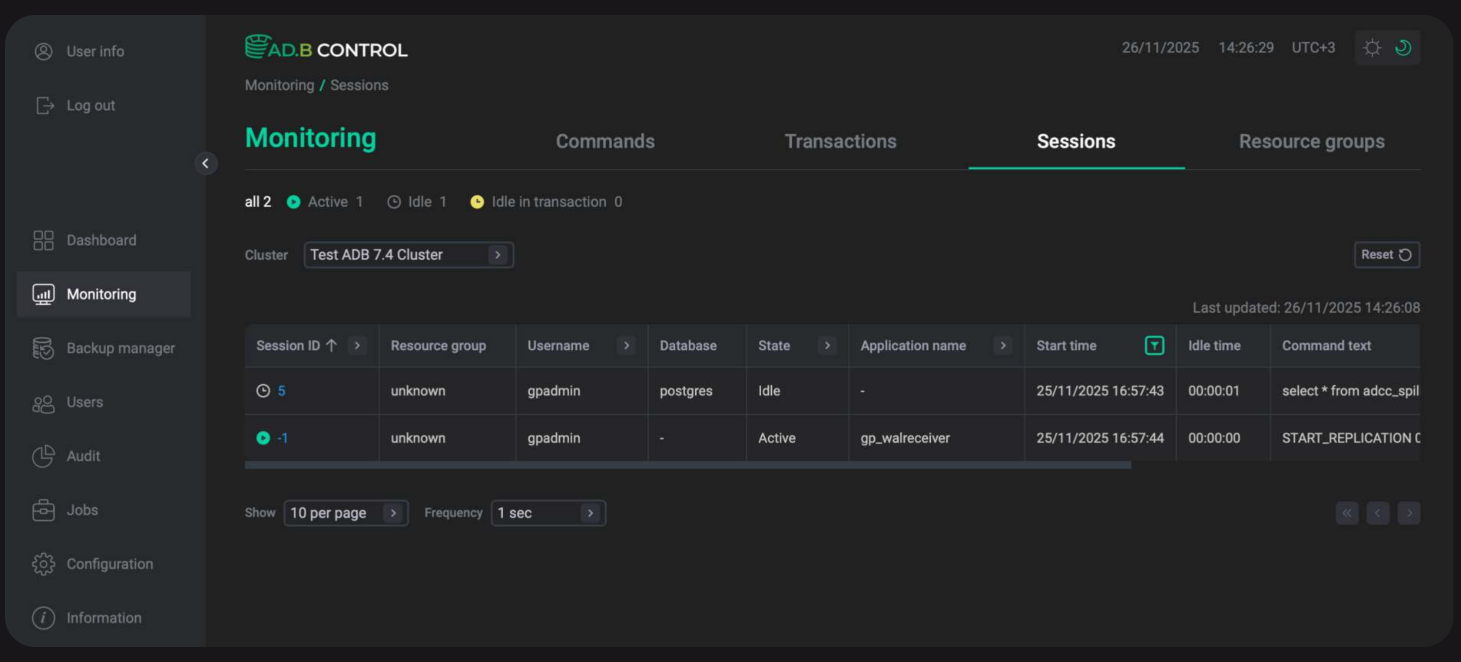
Cancel
Terminate

Terminate session

All process (PIDs), which are related to the selected session will be terminated

Confirm the operation

As a result, the session is terminated and no longer displayed on the *Monitoring* → *Sessions* tab.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/11/2025 14:26:29 UTC+3
Monitoring / Sessions
Monitoring
Commands
Transactions
Sessions
Resource groups
all 2 Active 1 Idle 1 Idle in transaction 0
Cluster Test ADB 7.4 Cluster
Reset
Last updated: 26/11/2025 14:26:08
Show 10 per page Frequency 1 sec

Session ID	Resource group	Username	Database	State	Application name	Start time	Idle time	Command text
5	unknown	gpadmin	postgres	Idle	-	25/11/2025 16:57:43	00:00:01	select * from adcc_spil
-1	unknown	gpadmin	-	Active	gp_walreceiver	25/11/2025 16:57:44	00:00:00	START_REPLICATION (

Session data is removed

Work with query tags

Daria Barysheva

To Table of Contents

Starting with 4.7.5, ADB Control supports **query tagging** — assigning custom metadata to SQL queries in the form of `<key>: <values>` pairs. You can set tags for a session or transaction in ADB and then view and filter these tags on the ***Monitoring*** → ***Commands***, ***Monitoring*** → ***Transactions***, and ***Monitoring*** → ***Sessions*** pages in ADB Control. Tags are assigned to ADB queries via the `query_band` GUC. This functionality can be useful for monitoring query groups, for example, as part of the common ETL process.

Below is an example of working with query tags:

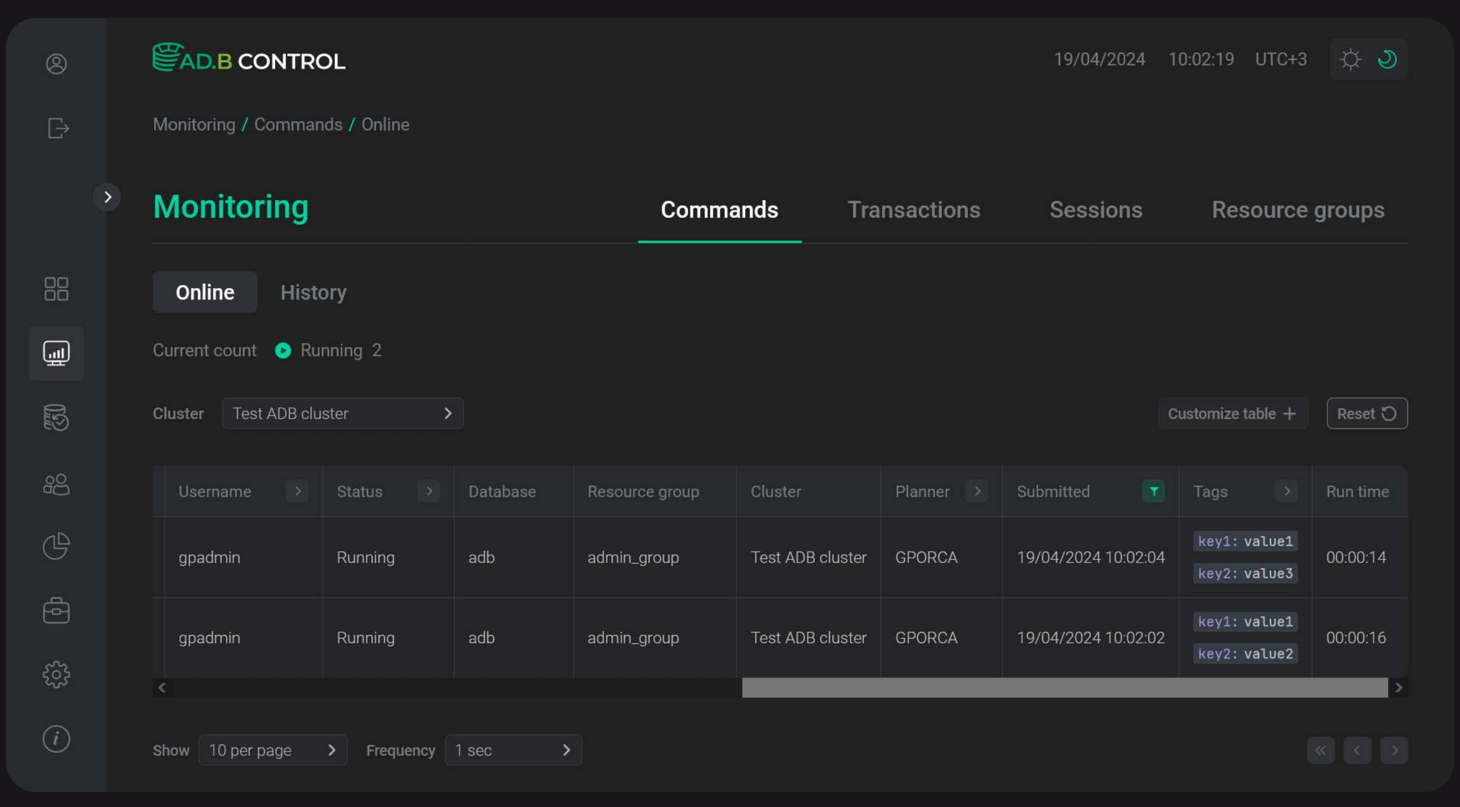
1. Open the first session in ADB (e.g. by connecting to the `adb` database on the coordinator host via `psql`).
2. Set values for two tags at the session level:

```sql
SET query_band = 'key1: value1,key2: value2';
```

3. Run any SQL query that lasts long enough to appear in the ADB Control web interface.
4. Within the second ADB session, set a different value for the `key2` tag:

```sql
SET query_band = 'key1: value1,key2: value3';
```

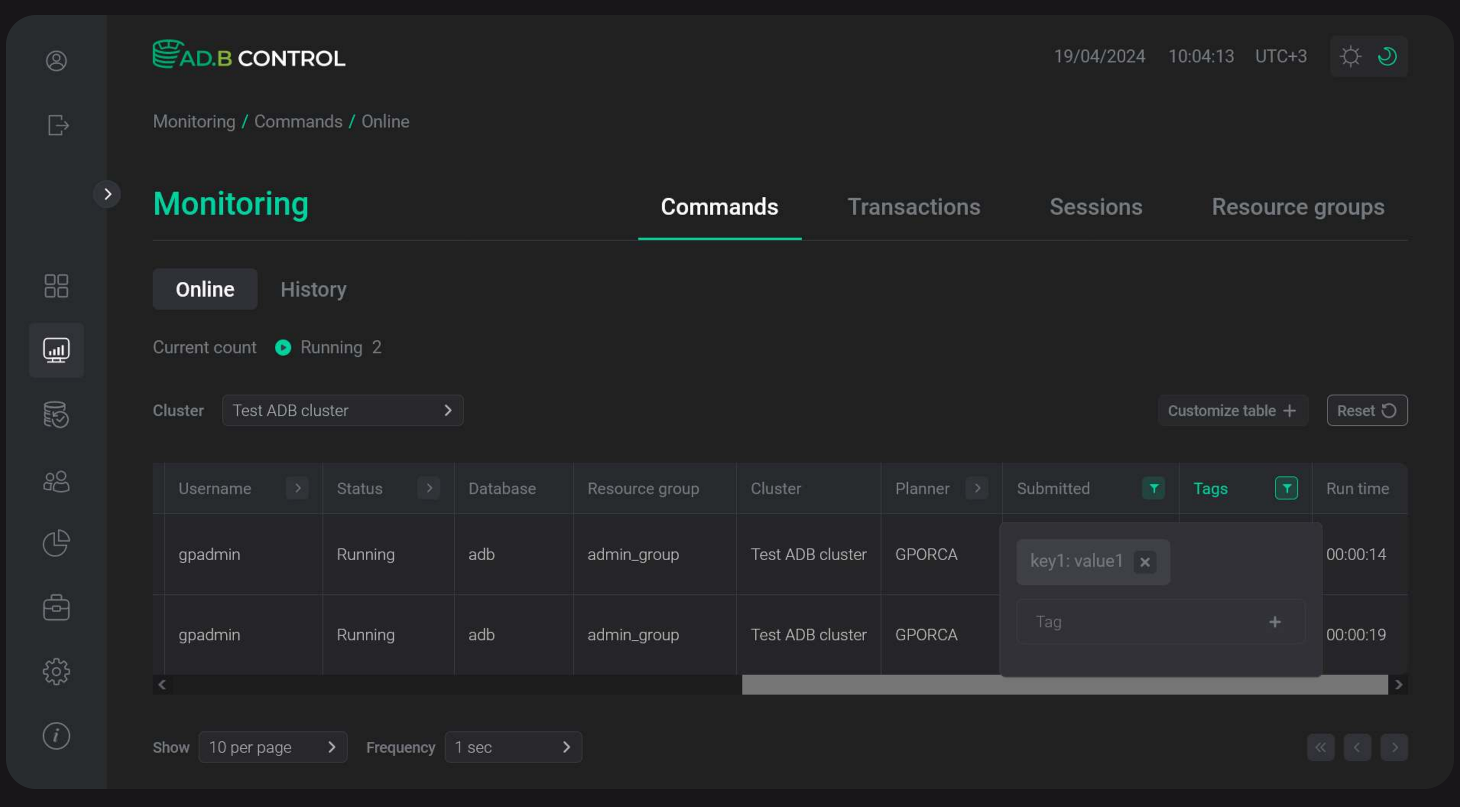
5. Run another SQL query within the second session.
6. Go to the ***Monitoring*** → ***Commands*** → ***Queries*** → ***Online*** page in the ADB Control web UI. Tag values for both running queries are shown in the ***Tags*** column. The same column is available on the ***Monitoring*** → ***Commands*** → ***Queries*** → ***History***, ***Monitoring*** → ***Transactions*** (***Online*** and ***History***), ***Monitoring*** → ***Sessions***, and ***Monitoring*** → ***Resource groups*** → ***Statistics*** → ***<resource group>*** → ***Commands*** → ***Queueing*** pages.


AD.B CONTROL
19/04/2024 10:02:19 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 2
Cluster
Test ADB cluster
Customize table
Reset
Show
10 per page
Frequency
1 sec

Username	Status	Database	Resource group	Cluster	Planner	Submitted	Tags	Run time
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA	19/04/2024 10:02:04	`key1: value1` `key2: value3`	00:00:14
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA	19/04/2024 10:02:02	`key1: value1` `key2: value2`	00:00:16

Tags in ADB Control

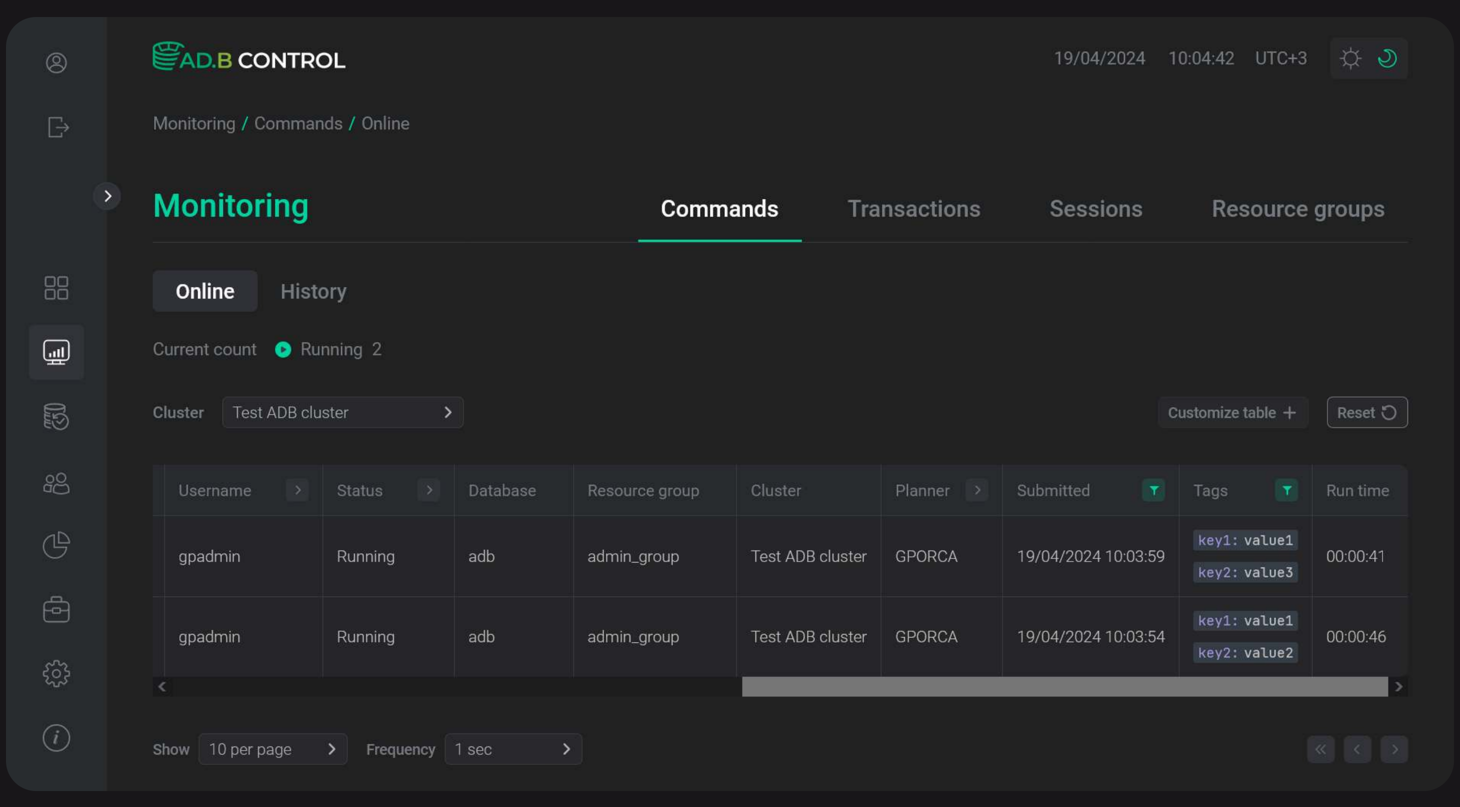
7. To select the queries with the specific tag value, open a drop-down list in the ***Tags*** column header, set the tag value in the ***Tag*** column as `<key>: <value>`, and click the + icon. To remove any tag from the filter, click ✕ .


AD.B CONTROL
19/04/2024 10:04:13 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 2
Cluster
Test ADB cluster
Customize table
Reset
key1: value1
Tag
Show
10 per page
Frequency
1 sec

Username	Status	Database	Resource group	Cluster	Planner	Submitted	Tags	Run time
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA			00:00:14
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA			00:00:19

Filter by the first tag value

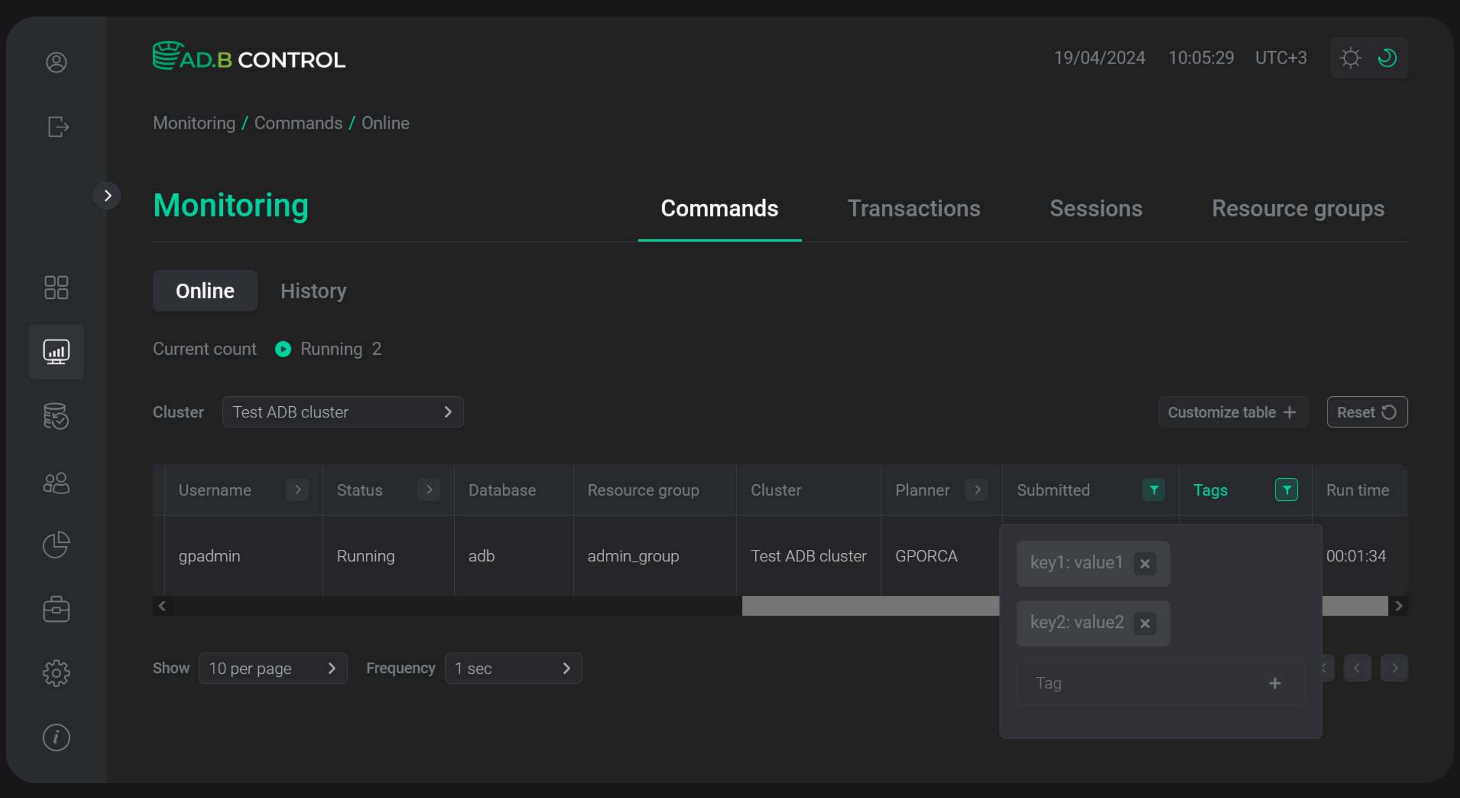
As a result of filtering, both queries are left on the page, because for both of them `value1` is set as the `key1` value.


AD.B CONTROL
19/04/2024 10:04:42 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 2
Cluster
Test ADB cluster
Customize table
Reset
Show
10 per page
Frequency
1 sec

Username	Status	Database	Resource group	Cluster	Planner	Submitted	Tags	Run time
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA	19/04/2024 10:03:59	`key1: value1` `key2: value3`	00:00:41
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA	19/04/2024 10:03:54	`key1: value1` `key2: value2`	00:00:46

Filtering result for one tag

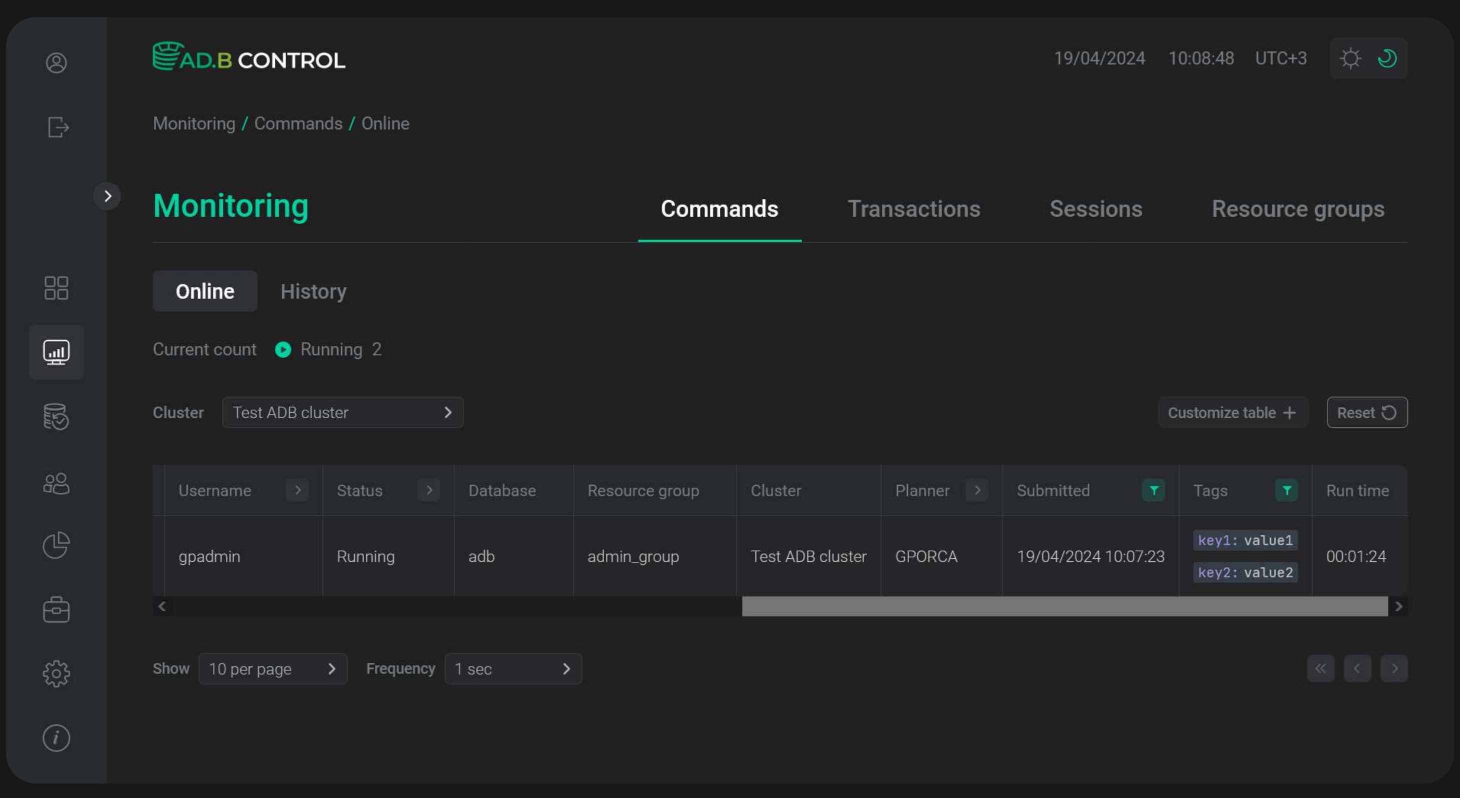
8. You can add a filter by the second tag value as shown below. Each tag value should be entered separately.


AD.B CONTROL
19/04/2024 10:05:29 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 2
Cluster
Test ADB cluster
Customize table
Reset
key1: value1
key2: value2
Tag
Show
10 per page
Frequency
1 sec

Username	Status	Database	Resource group	Cluster	Planner	Submitted	Tags	Run time
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA			00:01:34

Add a filter by the second tag value

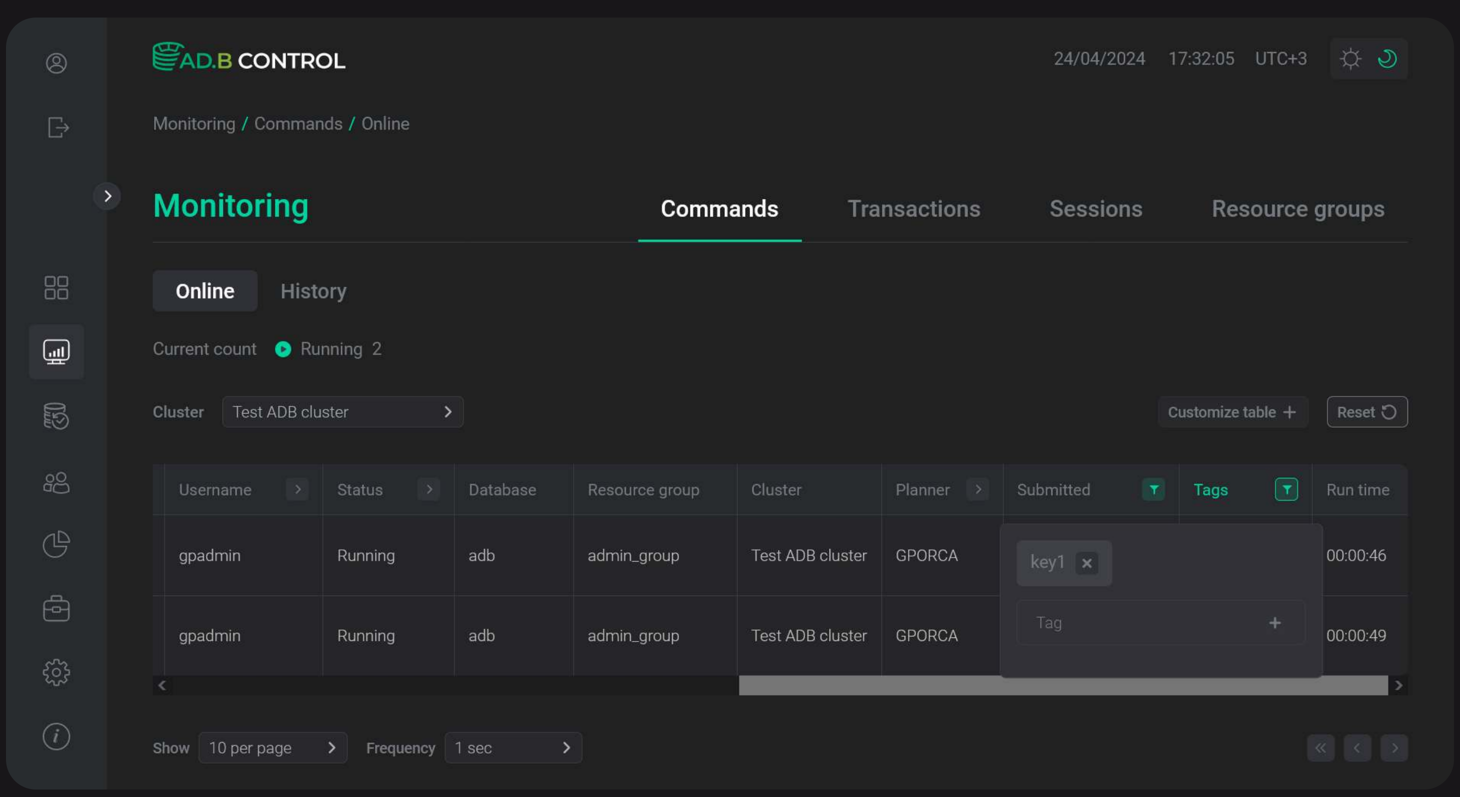
After you apply a filter, only the first session query is left on the page.


AD.B CONTROL
19/04/2024 10:08:48 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 2
Cluster
Test ADB cluster
Customize table
Reset
Show
10 per page
Frequency
1 sec

Username	Status	Database	Resource group	Cluster	Planner	Submitted	Tags	Run time
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA	19/04/2024 10:07:23	`key1: value1` `key2: value2`	00:01:24

Filtering result for two tags

9. Also, you can filter data by tag names without tag values. As a result, all queries, for which the selected tag is specified, are shown, despite the tag value.


AD.B CONTROL
24/04/2024 17:32:05 UTC+3
Monitoring / Commands / Online
Monitoring
Commands
Transactions
Sessions
Resource groups
Online
History
Current count
Running 2
Cluster
Test ADB cluster
Customize table
Reset
key1
Tag
Show
10 per page
Frequency
1 sec

Username	Status	Database	Resource group	Cluster	Planner	Submitted	Tags	Run time
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA			00:00:46
gpadmin	Running	adb	admin_group	Test ADB cluster	GPORCA			00:00:49

Filtering by the tag name

Audit

Daria Barysheva, Anton Vasilev

Contents

To Table of Contents

- ADB Control
  - Relations
  - Operations
  - Authorizations
- ADBM
  - Clusters
  - Storages
- Table filters

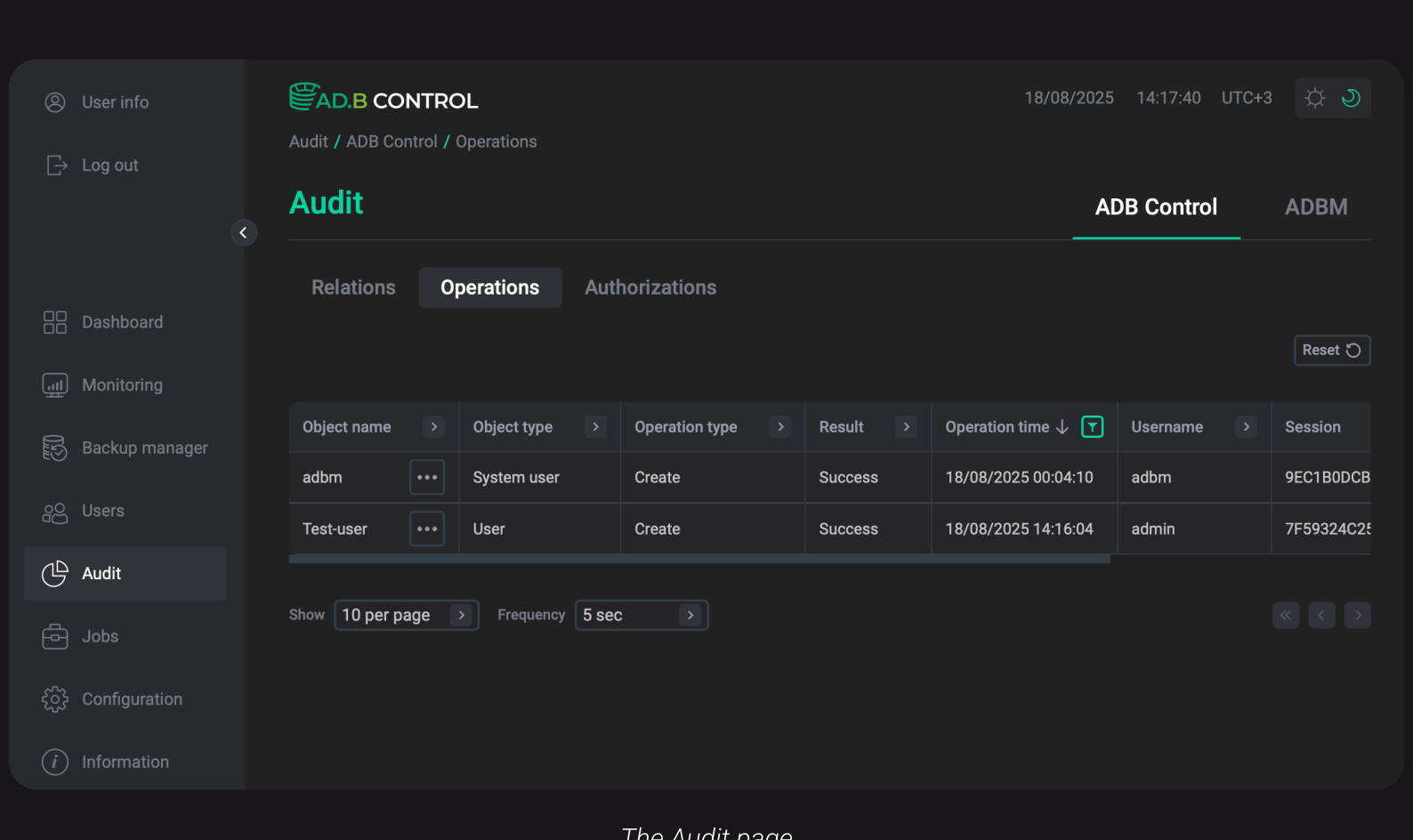
The ***Audit*** page in the ADB Control web interface displays information on user operations and authorizations, relation accesses (in clusters used for monitoring) as well as changes in clusters and storages. This page consists of two tabs:

- ADB Control
- ADBM (the tab is available starting with ADB Control 4.14.2)

ADB Control

The ***ADB Control*** page displays the following information in the form of separate tabs:

- Relations
- Operations
- Authorizations



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 14:17:40 UTC+3
Audit / ADB Control / Operations
Audit
ADB Control
ADBM
Relations
Operations
Authorizations
Reset
Show 10 per page
Frequency 5 sec

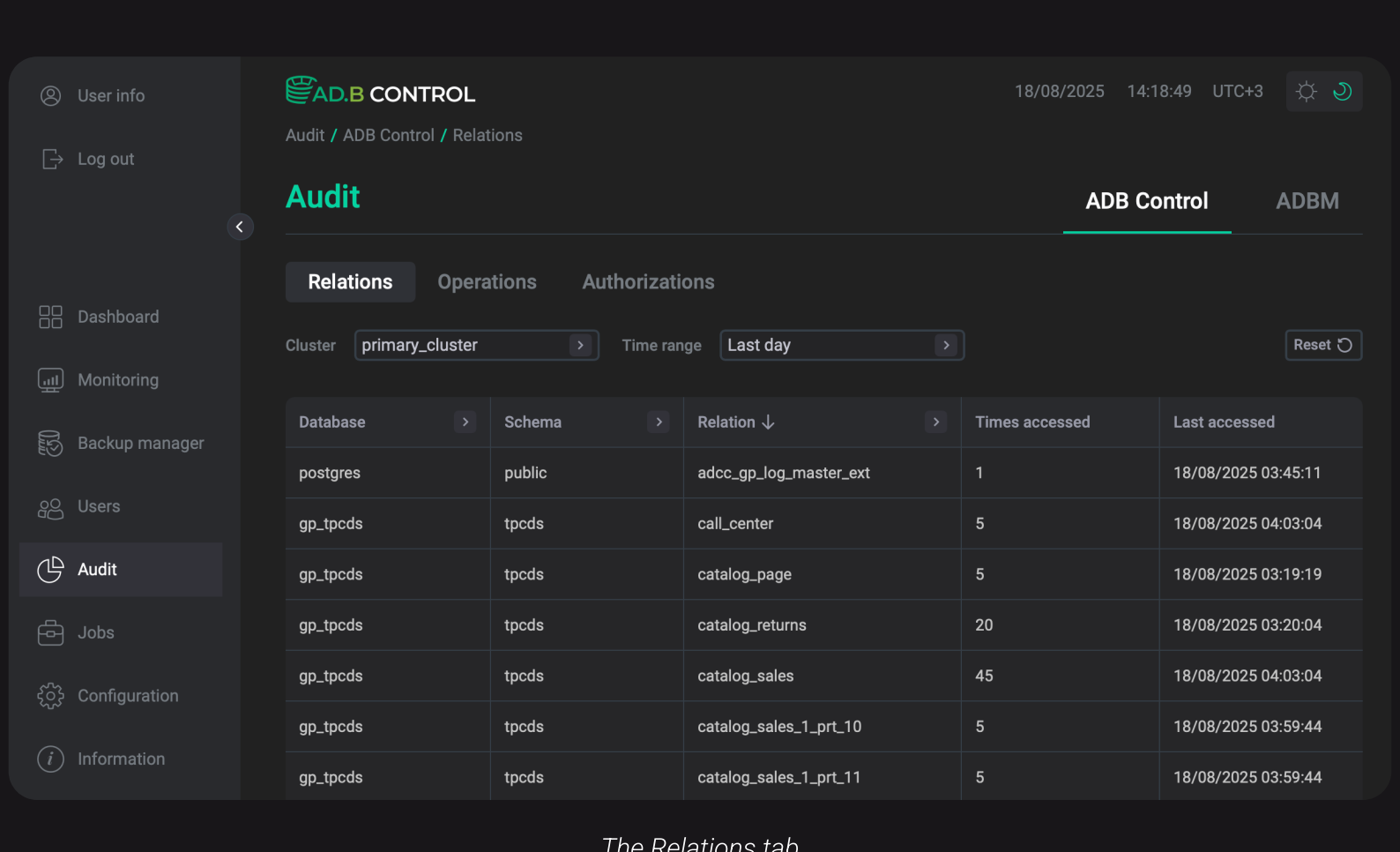
Object name	Object type	Operation type	Result	Operation time	Username	Session
adbm	System user	Create	Success	18/08/2025 00:04:10	adbm	9EC1B0DCB
Test-user	User	Create	Success	18/08/2025 14:16:04	admin	7F59324C2!

The Audit page

Relations

List of relations

The ***ADB Control*** → ***Relations*** tab on the ***Audit*** page displays the number of user accesses to relations of databases located in ADB clusters that are used for monitoring. These statistics include both individual SQL commands and commands from query groups. You can view data for the current date, week, or month.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 14:18:49 UTC+3
Audit / ADB Control / Relations
Audit
ADB Control
ADBM
Relations
Operations
Authorizations
Cluster primary_cluster
Time range Last day
Reset

Database	Schema	Relation	Times accessed	Last accessed
postgres	public	adcc_gp_log_master_ext	1	18/08/2025 03:45:11
gp_tpcds	tpcds	call_center	5	18/08/2025 04:03:04
gp_tpcds	tpcds	catalog_page	5	18/08/2025 03:19:19
gp_tpcds	tpcds	catalog_returns	20	18/08/2025 03:20:04
gp_tpcds	tpcds	catalog_sales	45	18/08/2025 04:03:04
gp_tpcds	tpcds	catalog_sales_1_prt_10	5	18/08/2025 03:59:44
gp_tpcds	tpcds	catalog_sales_1_prt_11	5	18/08/2025 03:59:44

The Relations tab

On the tab, there is a table with the following information.

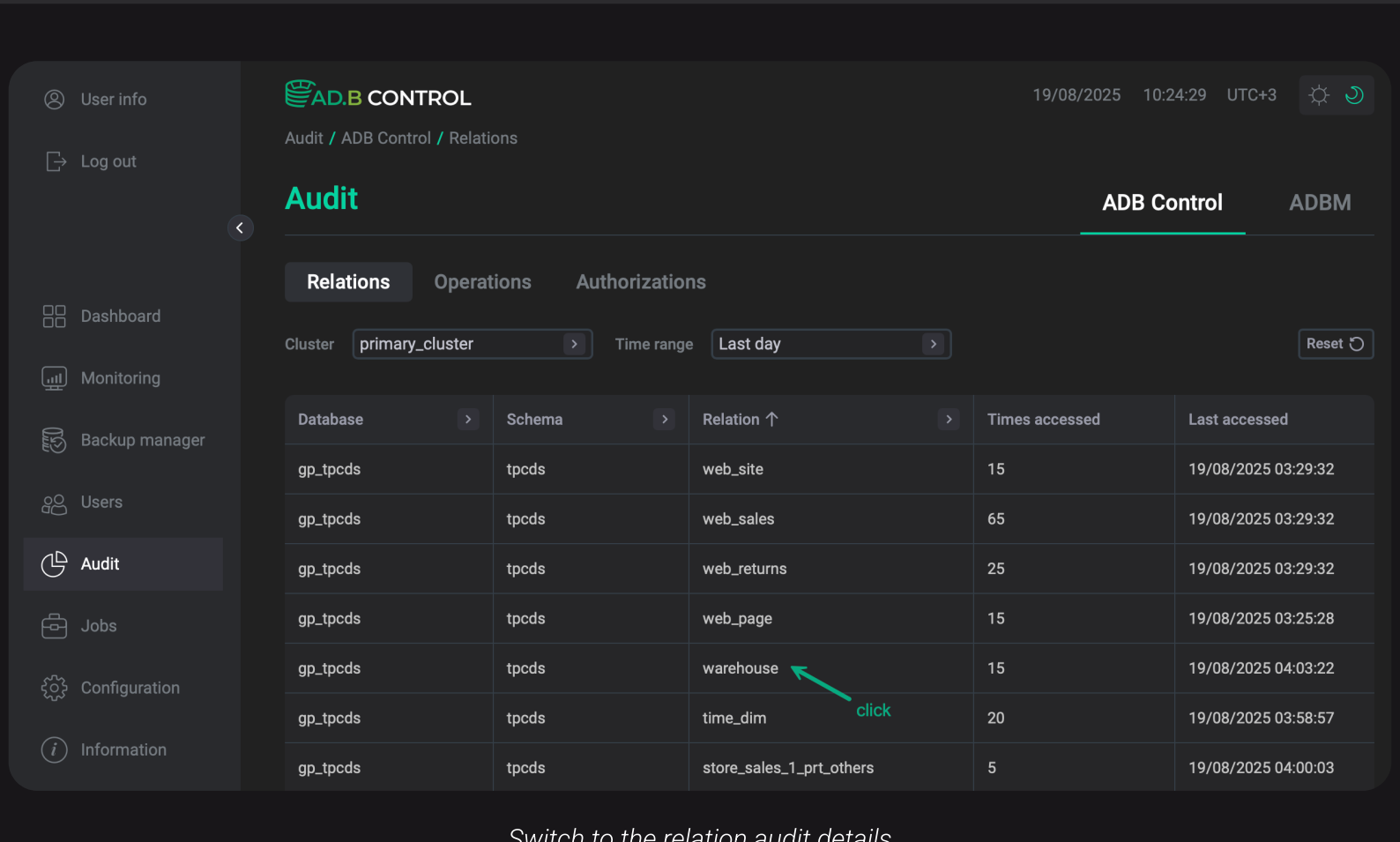
The Relations tab fields

Field	Description
Database	A database name
Schema	A schema name
Relation	A relation name
Times accessed	A number of accesses to the specified relation for the time period that is defined in the ***Select time period*** filter
Last accessed	A timestamp of the last access to the specified relation in the `DD/MM/YYYY HH:mm:ss` format

Relation audit details

To view audit details for the specific relation, you can click the row corresponding to that relation in the table listed above.

> **IMPORTANT**
>
> The ability to view the relation audit details is available for ADB Control users with appropriate permissions (see `View secured relation audit` in the Authorization article).



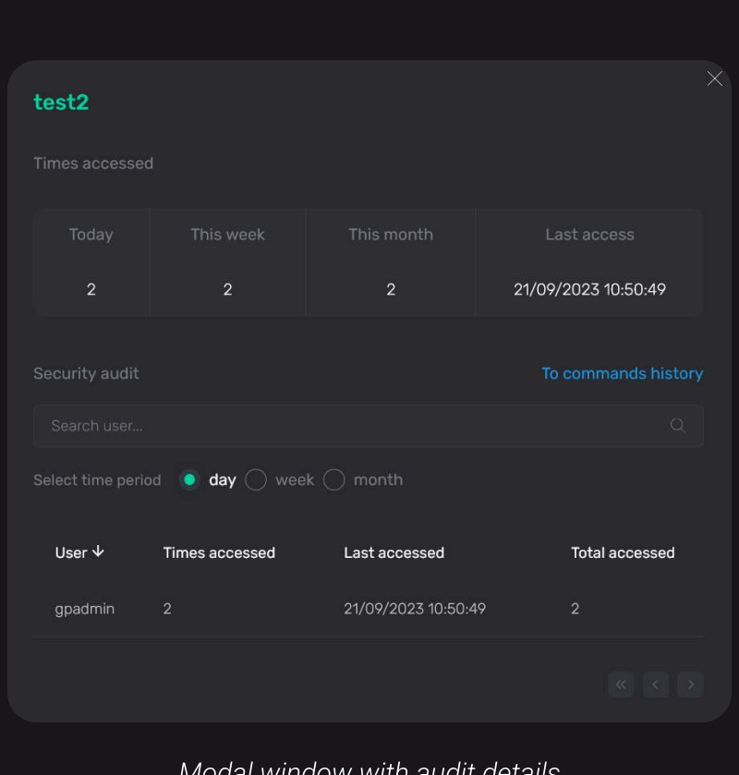
User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
19/08/2025 10:24:29 UTC+3
Audit / ADB Control / Relations
Audit
ADB Control
ADBM
Relations
Operations
Authorizations
Cluster primary_cluster
Time range Last day
Reset
click

Database	Schema	Relation	Times accessed	Last accessed
gp_tpcds	tpcds	web_site	15	19/08/2025 03:29:32
gp_tpcds	tpcds	web_sales	65	19/08/2025 03:29:32
gp_tpcds	tpcds	web_returns	25	19/08/2025 03:29:32
gp_tpcds	tpcds	web_page	15	19/08/2025 03:25:28
gp_tpcds	tpcds	warehouse	15	19/08/2025 04:03:22
gp_tpcds	tpcds	time_dim	20	19/08/2025 03:58:57
gp_tpcds	tpcds	store_sales_1_prt_others	5	19/08/2025 04:00:03

Switch to the relation audit details

The window that opens includes two tabs:

- ***Times accessed*** — general statistics of accesses to the relation over the entire observation period for all users.
- ***Security audit*** — statistics of accesses to the relation in the context of ADB users over the time range that is selected in the ***Select time period*** filter.



test2
Times accessed
Security audit
To commands history
Search user...
Select time period
● day ○ week ○ month

Today	This week	This month	Last access
2	2	2	21/09/2023 10:50:49

User	Times accessed	Last accessed	Total accessed
gpadmin	2	21/09/2023 10:50:49	2

Modal window with audit details

The Times accessed section fields

Field	Description
Today	A total number of accesses to the specified relation for the current day
This week	A total number of accesses to the specified relation for the current week
This month	A total number of accesses to the specified relation for the current month
Last access	A timestamp of the last access to the specified relation in the `DD/MM/YYYY HH:mm:ss` format

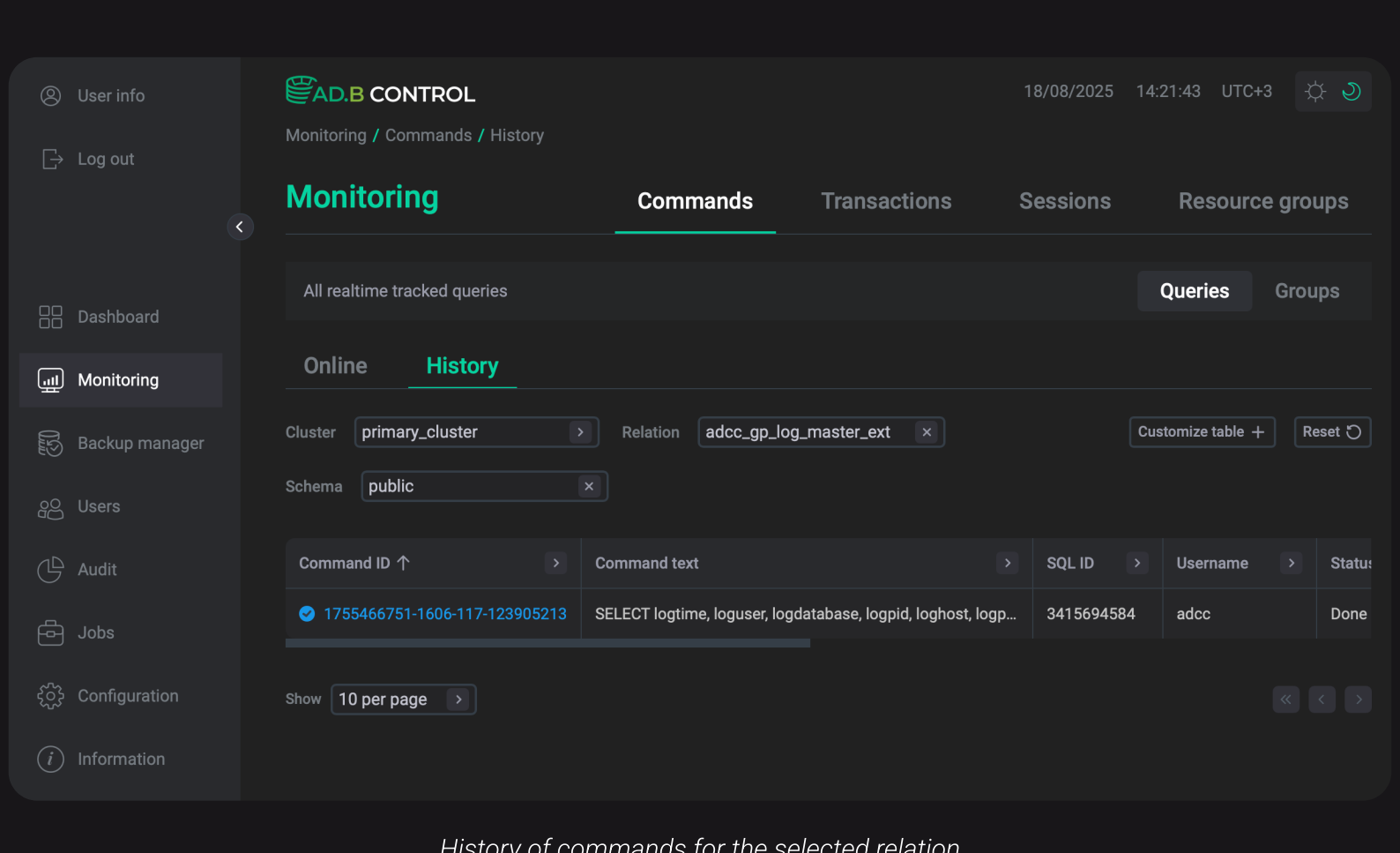
The Security audit tab

Field	Description
User	A name of the user who accessed the relation during the selected time period
Times accessed	A number of user accesses to the relation over the selected time period
Last accessed	A timestamp of the last user access to the relation in the `DD/MM/YYYY HH:mm:ss` format
Total accessed	A total number of user accesses to the relation over the entire observation period

Above the table in the Security audit section, there are filters that you can use to select specific data. Available filters are listed below:

- ***Search user...*** — filter by the user name (see ***User*** above). Enter a full value.
- ***Select time period*** — filter by the last access time of the specific user to the relation (see ***Last accessed*** above). Select one of the following values:
  - `day` — current day;
  - `week` — current week;
  - `month` — current month.

At the top of the Security audit section, the ***To commands history*** link is located. You can click it to view the history of SQL commands for the selected relation on the Monitoring → Commands → Queries → History page. The ***Relation*** and ***Schema*** filters are filled in on the history page automatically.



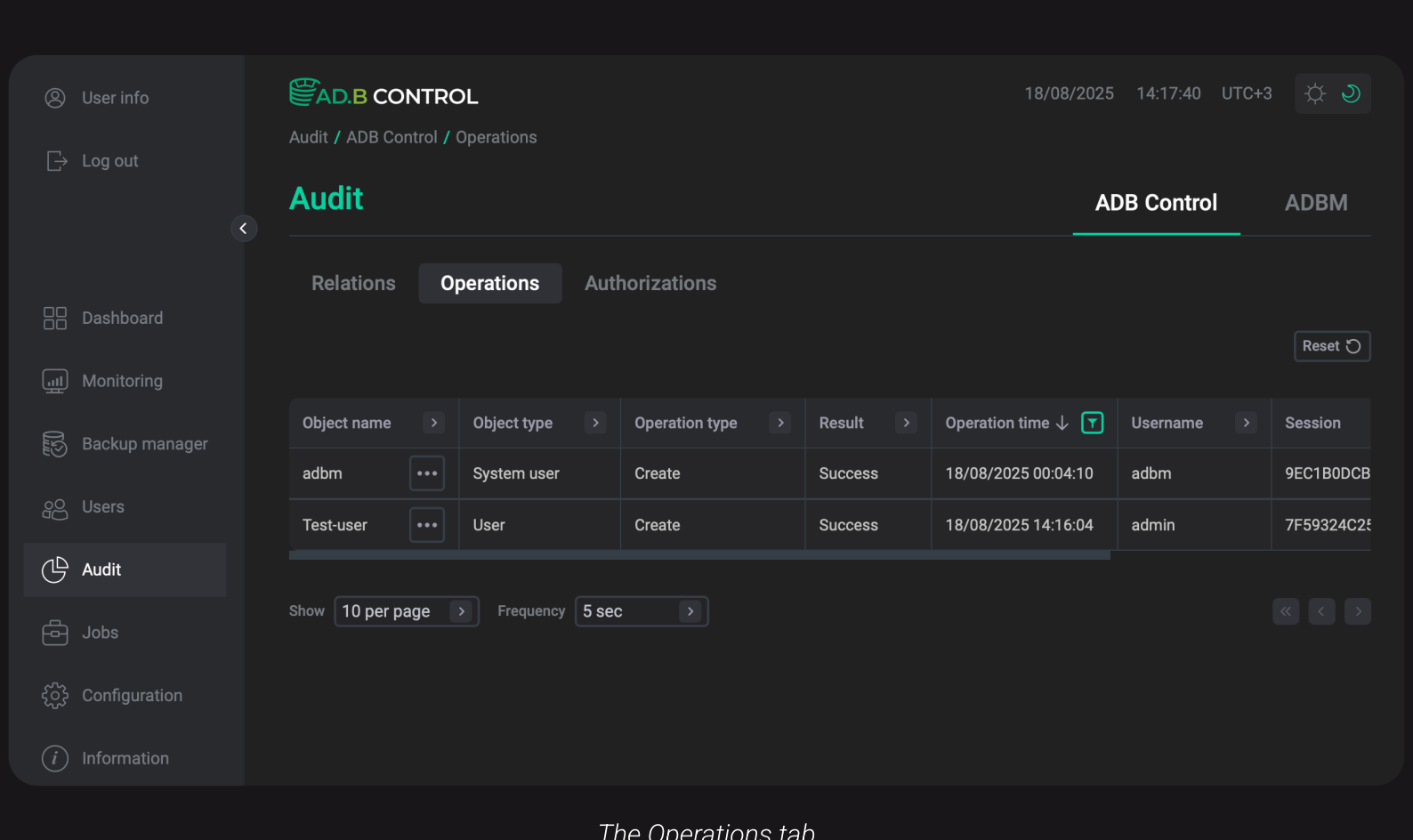
User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 14:21:43 UTC+3
Monitoring / Commands / History
Monitoring
Commands
Transactions
Sessions
Resource groups
All realtime tracked queries
Queries
Groups
Online
History
Cluster primary_cluster
Relation adcc_gp_log_master_ext
Schema public
Customize table
Reset
Show 10 per page

Command ID	Command text	SQL ID	Username	Status
1755466751-1606-117-123905213	SELECT logtime, loguser, logdatabase, logpid, loghost, logp...	3415694584	adcc	Done

History of commands for the selected relation

Operations

The ***ADB Control*** → ***Operations*** tab on the ***Audit*** page displays a list of operations launched by ADB Control users during the specified time period. Current day data is displayed by default.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 14:17:40 UTC+3
Audit / ADB Control / Operations
Audit
ADB Control
ADBM
Relations
Operations
Authorizations
Reset
Show 10 per page
Frequency 5 sec

Object name	Object type	Operation type	Result	Operation time	Username	Session
adbm	System user	Create	Success	18/08/2025 00:04:10	adbm	9EC1B0DCB
Test-user	User	Create	Success	18/08/2025 14:16:04	admin	7F59324C2!

The Operations tab

On the tab, there is a table with the following information.

The Operations tab fields

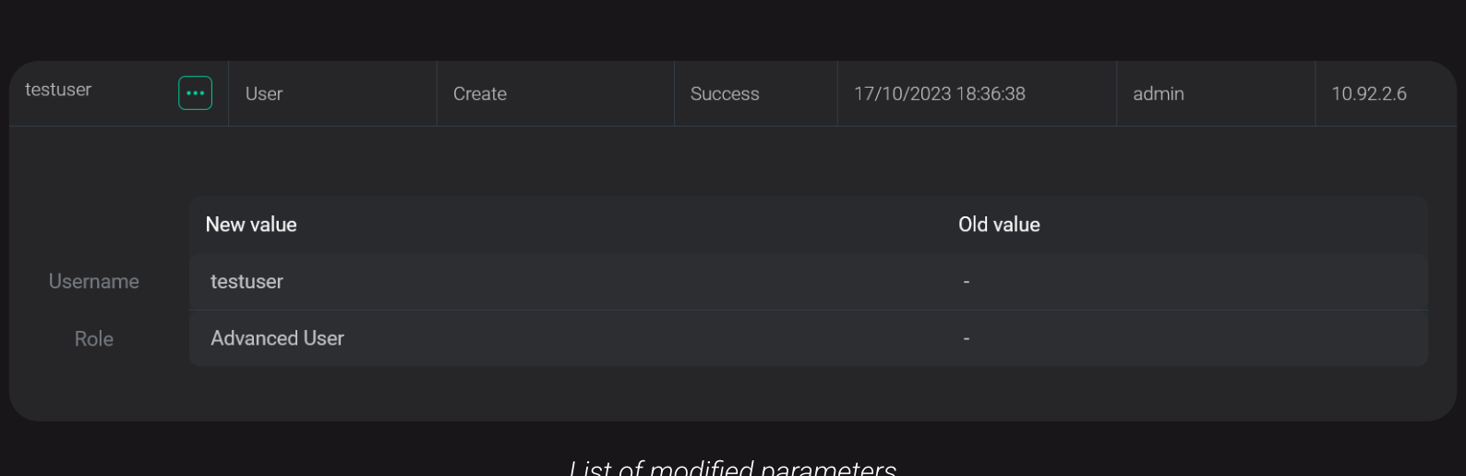
Field	Description
Object name	An object name
Object type	A type of the object on which the operation has been performed. Possible values: `User` — user; `User forbidden DB` — assigning advanced permissions to a specific database (actual for users with the `Advanced User` role); `LDAP mapping` — role mapping with LDAP group; `Transaction` — transaction; `Cluster` — cluster; `Cluster export databases` — databases, whose metrics will be offloaded; `Job` — ADB Control job; `System settings` — system configuration parameters; `System user` — system user; `Session` — ADB session; `User session` — ADB Control user session; `Cluster guc` — cluster GUC; `Cluster databases` — databases that should be monitored in ADB Control.

Operation type	A type of the operation that has been applied to the ***Object name*** object. Possible values:
	`Cancel`
	`Create`
	`Delete`
	`Resource group change`
	`Terminate`
	`Update`
	`Archive`
	`Activate`

Result	An operation result. Possible values:
	`Success`
	`Failed`
	`Denied`

Operation time	An operation end timestamp in the `DD/MM/YYYY HH:mm:ss` format
Username	A name of the user who performed the operation
Session	An identifier of the ADB Control user session where the operation was initiated
Host	IP address of the host from which the operation was initiated

The first table column ***Object name*** contains the ••• button. When you click this button, a block appears under the current table row showing a list of object parameters that were modified during the current operation.



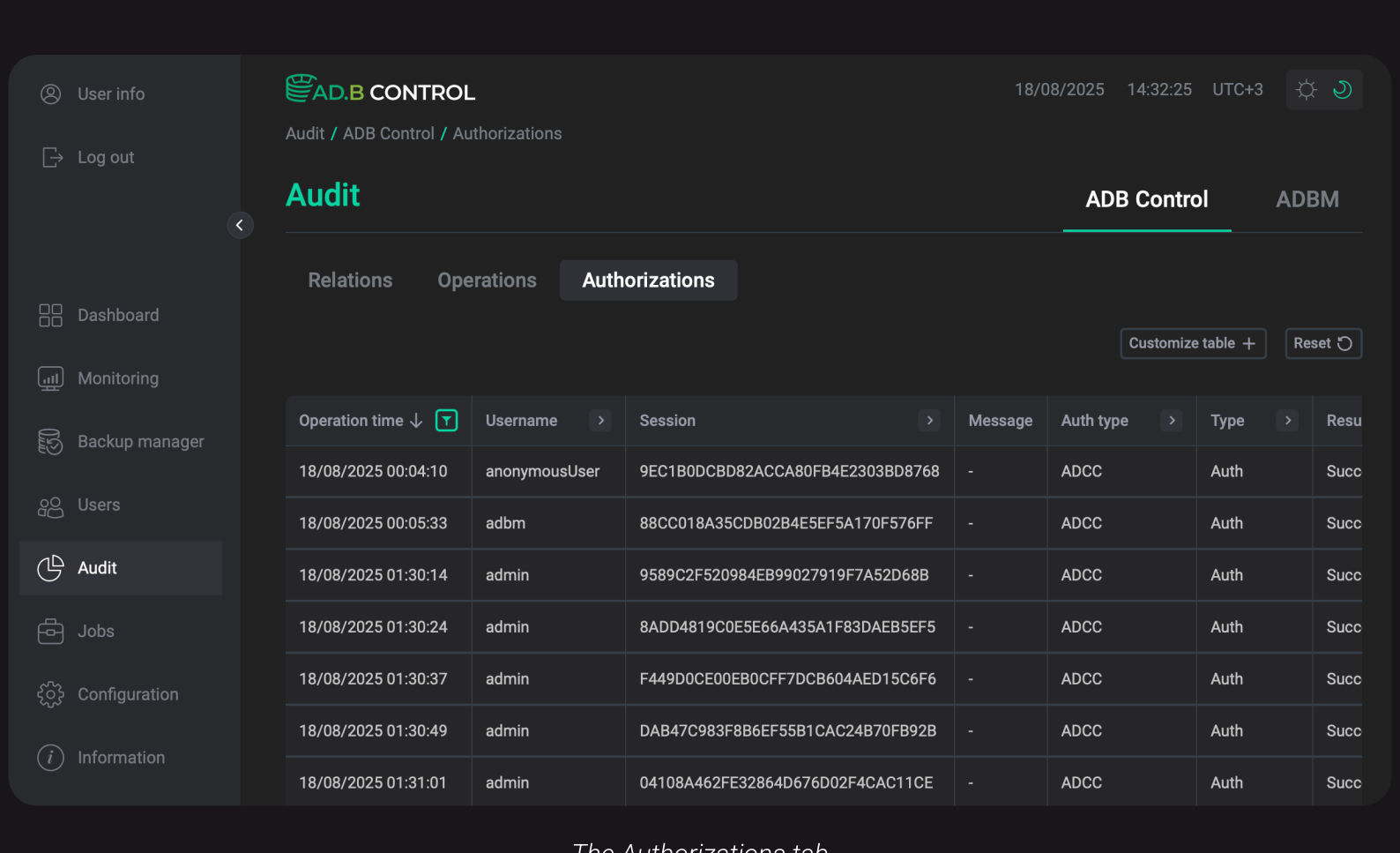
testuser	User	Create	Success	17/10/2023 18:36:38	admin	10.92.2.6

	New value	Old value
Username	testuser	-
Role	Advanced User	-

List of modified parameters

Authorizations

The ***ADB Control*** → ***Authorizations*** tab on the ***Audit*** page displays a list of ADB Control user authorizations for the selected time period. It displays both successful and failed login attempts as well as failed authorizations in ADB. Current day data is displayed by default.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 14:32:25 UTC+3
Audit / ADB Control / Authorizations
Audit
ADB Control
ADBM
Relations
Operations
Authorizations
Customize table
Reset

Operation time	Username	Session	Message	Auth type	Type	Resu
18/08/2025 00:04:10	anonymousUser	9EC1B0DCBD82ACCA80FB4E2303BD8768	-	ADCC	Auth	Succ
18/08/2025 00:05:33	adbm	88CC018A35CDB02B4E5EF5A170F576FF	-	ADCC	Auth	Succ
18/08/2025 01:30:14	admin	9589C2F520984EB99027919F7A52D68B	-	ADCC	Auth	Succ
18/08/2025 01:30:24	admin	8ADD4819C0E5E66A435A1F83DAEB5EF5	-	ADCC	Auth	Succ
18/08/2025 01:30:37	admin	F449D0CE00EB0CFF7DCB604AED15C6F6	-	ADCC	Auth	Succ
18/08/2025 01:30:49	admin	DAB47C983F8B6EF55B1CAC24B70FB92B	-	ADCC	Auth	Succ
18/08/2025 01:31:01	admin	04108A462FE32864D676D02F4CAC11CE	-	ADCC	Auth	Succ

The Authorizations tab

On the tab, there is a table with the following information.

The Authorizations tab fields

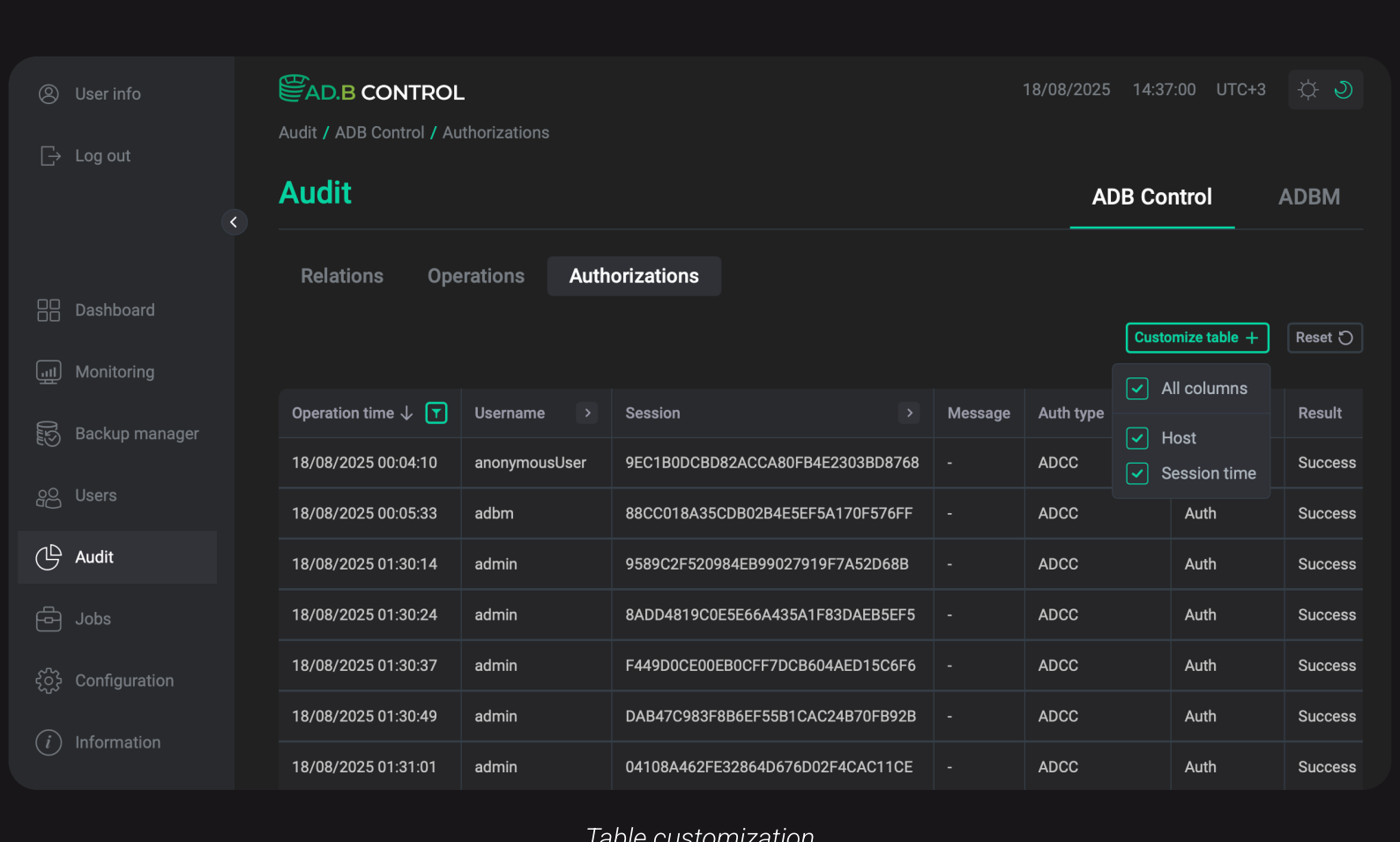
Field	Description
Operation time	An operation end timestamp in the `DD/MM/YYYY HH:mm:ss` format
Username	A name of the user who performed the operation
Session	An identifier of the ADB Control user session where the operation was initiated
Message	An error message. For example, the following message indicates incorrect credentials: `Bad credentials, 4 attempts left`
Auth type	An authorization type: `ADCC` — ADB Control login attempt via basic authentication; `ADCC LDAP` — ADB Control login attempt via LDAP authentication; `ADB` — failed ADB login attempt; `ADB LDAP` — failed ADB login attempt via LDAP authentication.

Type	An event type. Possible values:
	`Auth`
	`Logout`
	`Session expired`

Result	An event result. Possible values:
	`Success`
	`Failed`

Host	IP address of the host from which the event was initiated. This column becomes visible after table customization
Session Time	A start timestamp of the session (within which the current event took place) in the `DD/MM/YYYY HH:mm:ss` format. This column becomes visible after table customization

If necessary, you can add columns to the table. To do this, click ***Customize table*** and select column names in the section that opens. Currently you can add the ***Host*** and ***Session Time*** columns (see column descriptions above).



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 14:37:00 UTC+3
Audit / ADB Control / Authorizations
Audit
ADB Control
ADBM
Relations
Operations
Authorizations
Customize table
Reset
☑ All columns
☑ Host
☑ Session time

Operation time	Username	Session	Message	Auth type	Type	Result
18/08/2025 00:04:10	anonymousUser	9EC1B0DCBD82ACCA80FB4E2303BD8768	-	ADCC		Success
18/08/2025 00:05:33	adbm	88CC018A35CDB02B4E5EF5A170F576FF	-	ADCC	Auth	Success
18/08/2025 01:30:14	admin	9589C2F520984EB99027919F7A52D68B	-	ADCC	Auth	Success
18/08/2025 01:30:24	admin	8ADD4819C0E5E66A435A1F83DAEB5EF5	-	ADCC	Auth	Success
18/08/2025 01:30:37	admin	F449D0CE00EB0CFF7DCB604AED15C6F6	-	ADCC	Auth	Success
18/08/2025 01:30:49	admin	DAB47C983F8B6EF55B1CAC24B70FB92B	-	ADCC	Auth	Success
18/08/2025 01:31:01	admin	04108A462FE32864D676D02F4CAC11CE	-	ADCC	Auth	Success

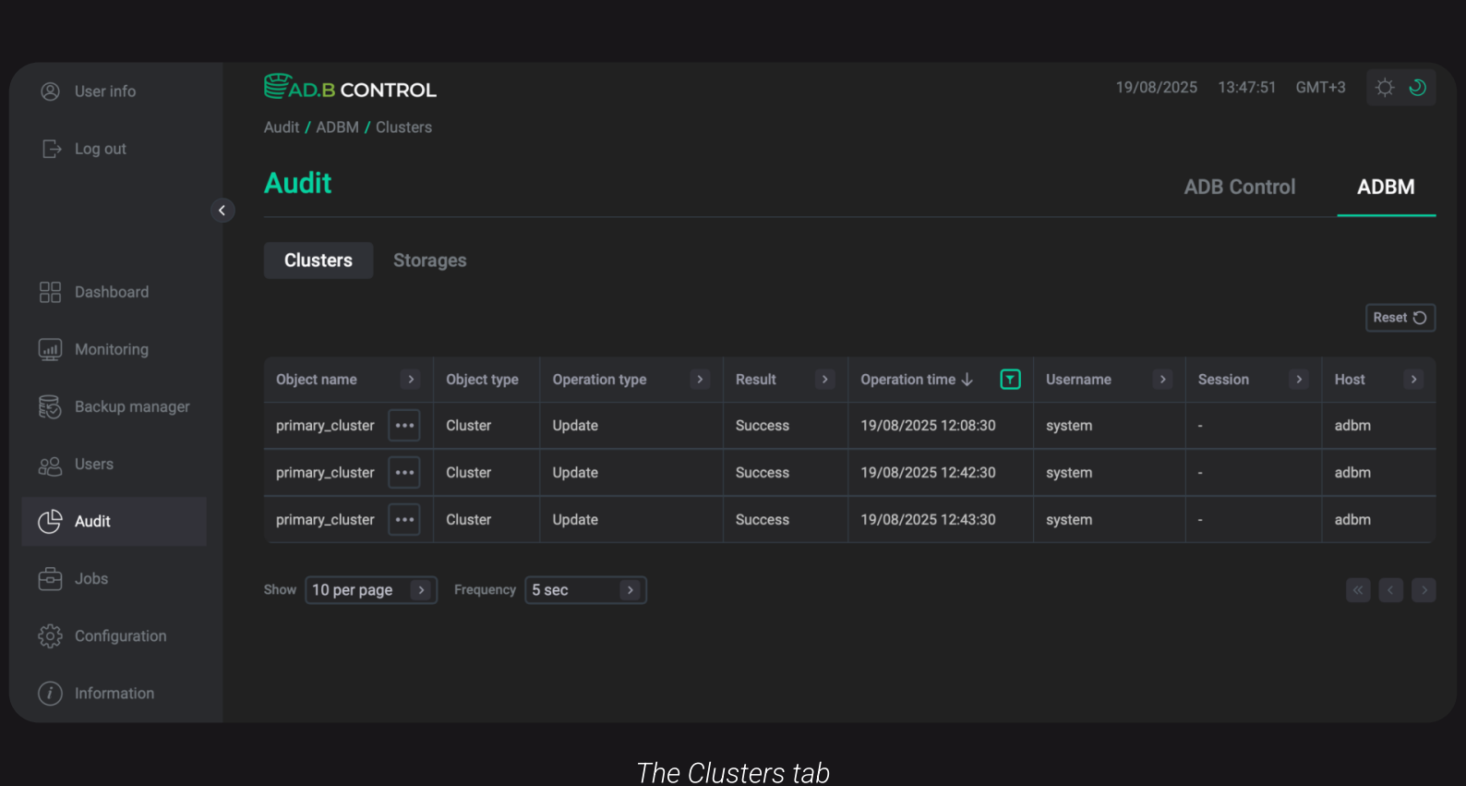
Table customization

ADBM

The ***ADBM*** tab displays information on user interactions with clusters and storages.

Clusters

The ***ADBM*** → ***Clusters*** tab displays information about creation, modification, and deletion of clusters.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
19/08/2025 13:47:51 GMT+3
Audit / ADBM / Clusters
Audit
ADB Control
ADBM
Clusters
Storages
Reset
Show 10 per page
Frequency 5 sec

Object name	Object type	Operation type	Result	Operation time	Username	Session	Host
primary_cluster	Cluster	Update	Success	19/08/2025 12:08:30	system	-	adbm
primary_cluster	Cluster	Update	Success	19/08/2025 12:42:30	system	-	adbm
primary_cluster	Cluster	Update	Success	19/08/2025 12:43:30	system	-	adbm

The Clusters tab

The tab consists of a table with the following information.

The Clusters tab fields

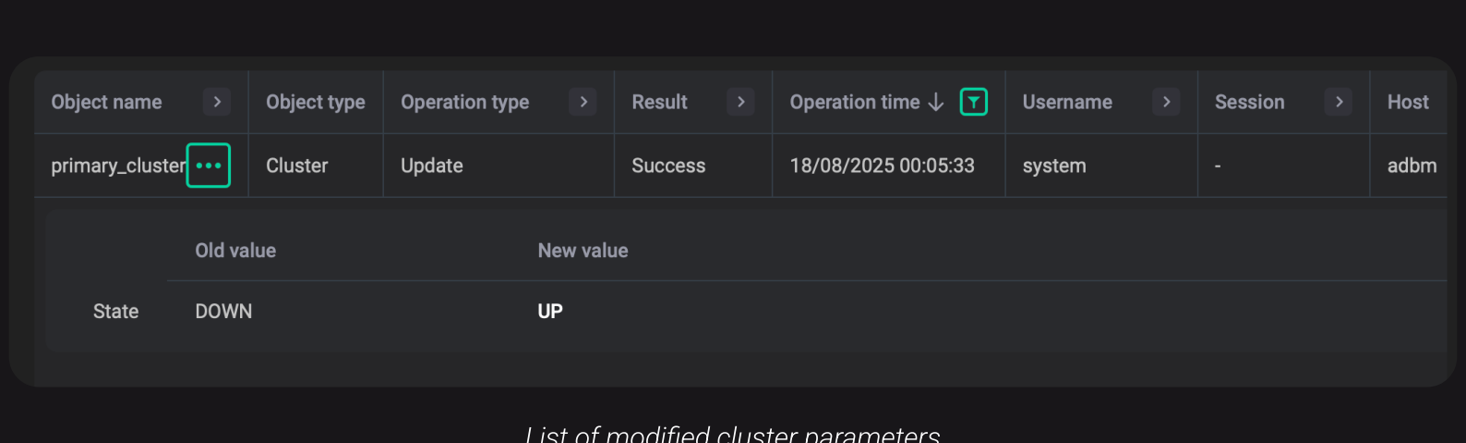
Field	Description
Object name	An object name
Object type	A type of the object on which the operation has been performed. For the ***Clusters*** tab, it's always `Cluster`

Operation type	A type of the operation that has been applied to the cluster. Possible values:
	`Create`
	`Update`
	`Delete`

Result	An operation result. Possible values:
	`Success`
	`Failed`
	`Denied`

Operation time	An operation end timestamp in the `DD/MM/YYYY HH:mm:ss` format
Username	A name of the user who performed the operation
Session	An identifier of the ADB Control user session where the operation was initiated
Host	IP address of the host from which the operation was initiated

The first table column ***Object name*** contains the ••• button. When you click this button, a block appears under the current table row showing a list of the cluster parameters that were modified during the current operation.



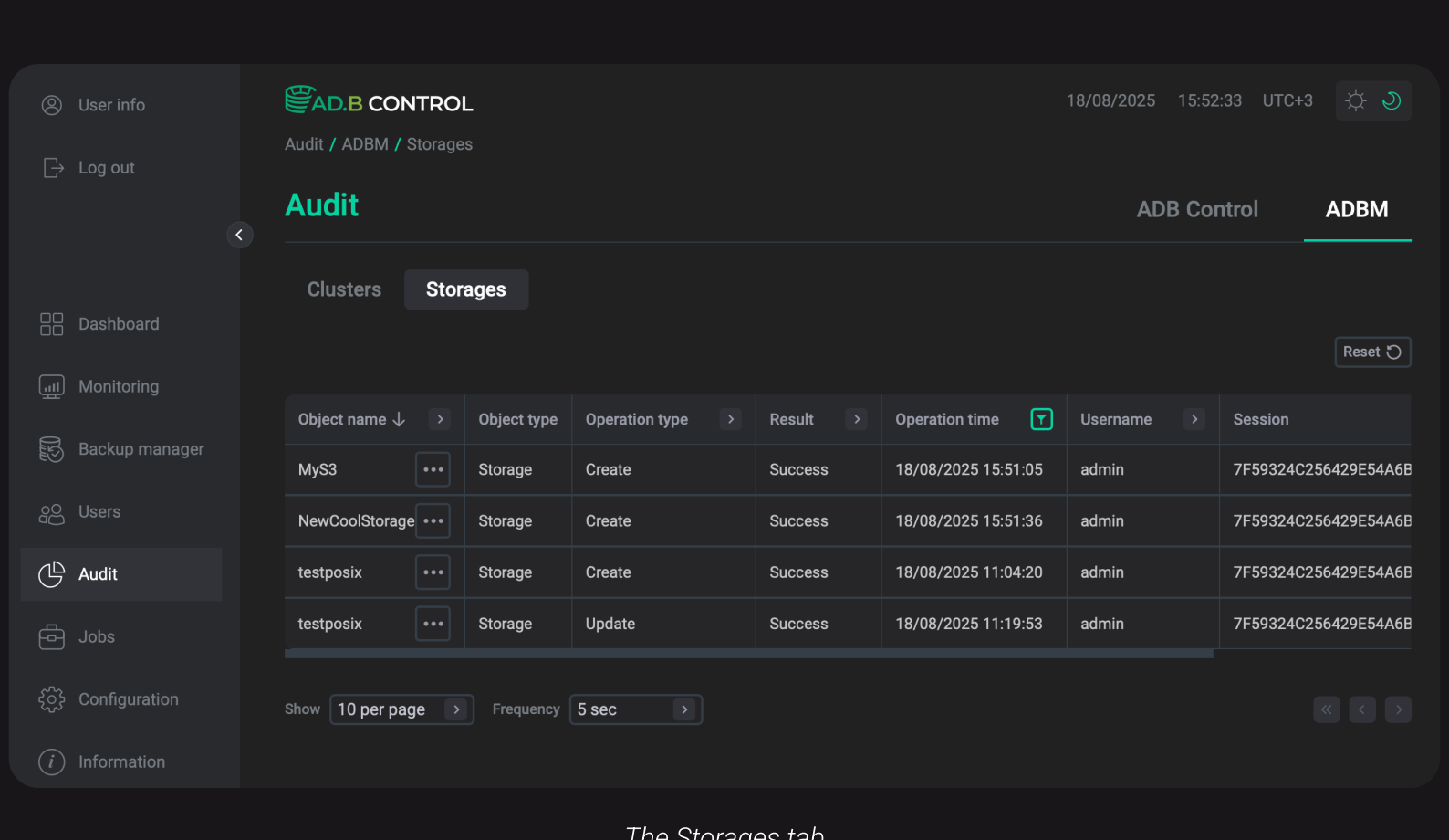
Object name	Object type	Operation type	Result	Operation time	Username	Session	Host
primary_cluster	Cluster	Update	Success	18/08/2025 00:05:33	system	-	adbm

	Old value	New value
State	DOWN	UP

List of modified cluster parameters

Storages

The ***ADBM*** → ***Storages*** tab displays information about creation, modification, and deletion of storages.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
18/08/2025 15:52:33 UTC+3
Audit / ADBM / Storages
Audit
ADB Control
ADBM
Clusters
Storages
Reset
Show 10 per page
Frequency 5 sec

Object name	Object type	Operation type	Result	Operation time	Username	Session
MyS3	Storage	Create	Success	18/08/2025 15:51:05	admin	7F59324C256429E54A6B
NewCoolStorage	Storage	Create	Success	18/08/2025 15:51:36	admin	7F59324C256429E54A6B
testposix	Storage	Create	Success	18/08/2025 11:04:20	admin	7F59324C256429E54A6B
testposix	Storage	Update	Success	18/08/2025 11:19:53	admin	7F59324C256429E54A6B

The Storages tab

The tab consists of a table with the following information.

The Storages tab fields

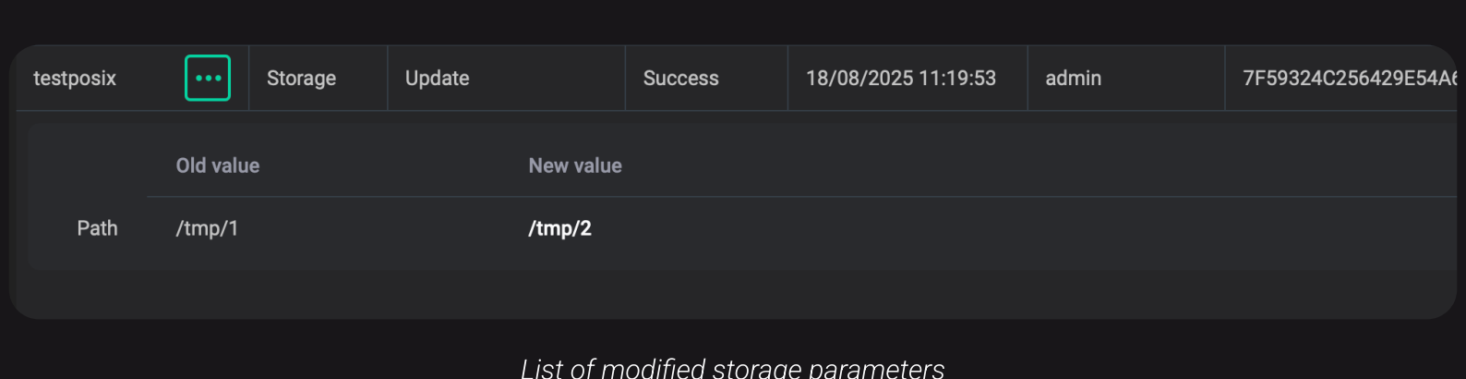
Field	Description
Object name	An object name
Object type	A type of the object on which the operation has been performed. For the ***Storages*** tab, it's always `Storage`

Operation type	A type of the operation that has been applied to the storage. Possible values:
	`Create`
	`Update`
	`Delete`

Result	An operation result. Possible values:
	`Success`
	`Failed`
	`Denied`

Operation time	An operation end timestamp in the `DD/MM/YYYY HH:mm:ss` format
Username	A name of the user who performed the operation
Session	An identifier of the ADB Control user session where the operation was initiated
Host	IP address of the host from which the operation was initiated

The first table column ***Object name*** contains the ••• button. When you click this button, a block appears under the current table row showing a list of the storage parameters that were modified during the current operation.



testposix	Storage	Update	Success	18/08/2025 11:19:53	admin	7F59324C256429E54A6

	Old value	New value
Path	/tmp/1	/tmp/2

List of modified storage parameters

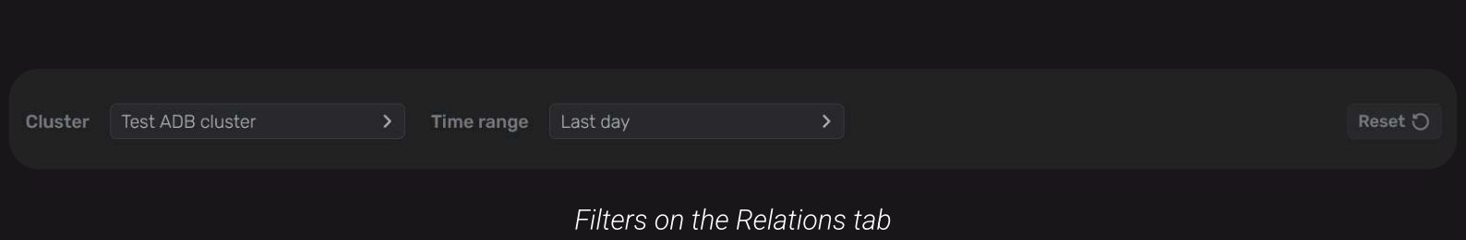
Table filters

In most of the column headers, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Auth type***), you can select a value from the drop-down list. For some columns (e.g. ***Username***), the search value should be entered. For columns that show date and time (e.g. ***Operation time***), the time range can be selected from the calendar.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Additionally, the Relations table contains the following filters:

- ***Cluster*** — filter by the cluster name. Select a value from the drop-down list.
- ***Time range*** — filter by the last access time (see ***Last accessed*** above). Select one of the following values:
  - `Last day` — current day;
  - `Last week` — current week;
  - `Last month` — current month.



Cluster
Test ADB cluster
Time range
Last day
Reset

Filters on the Relations tab

View jobs

Daria Barysheva

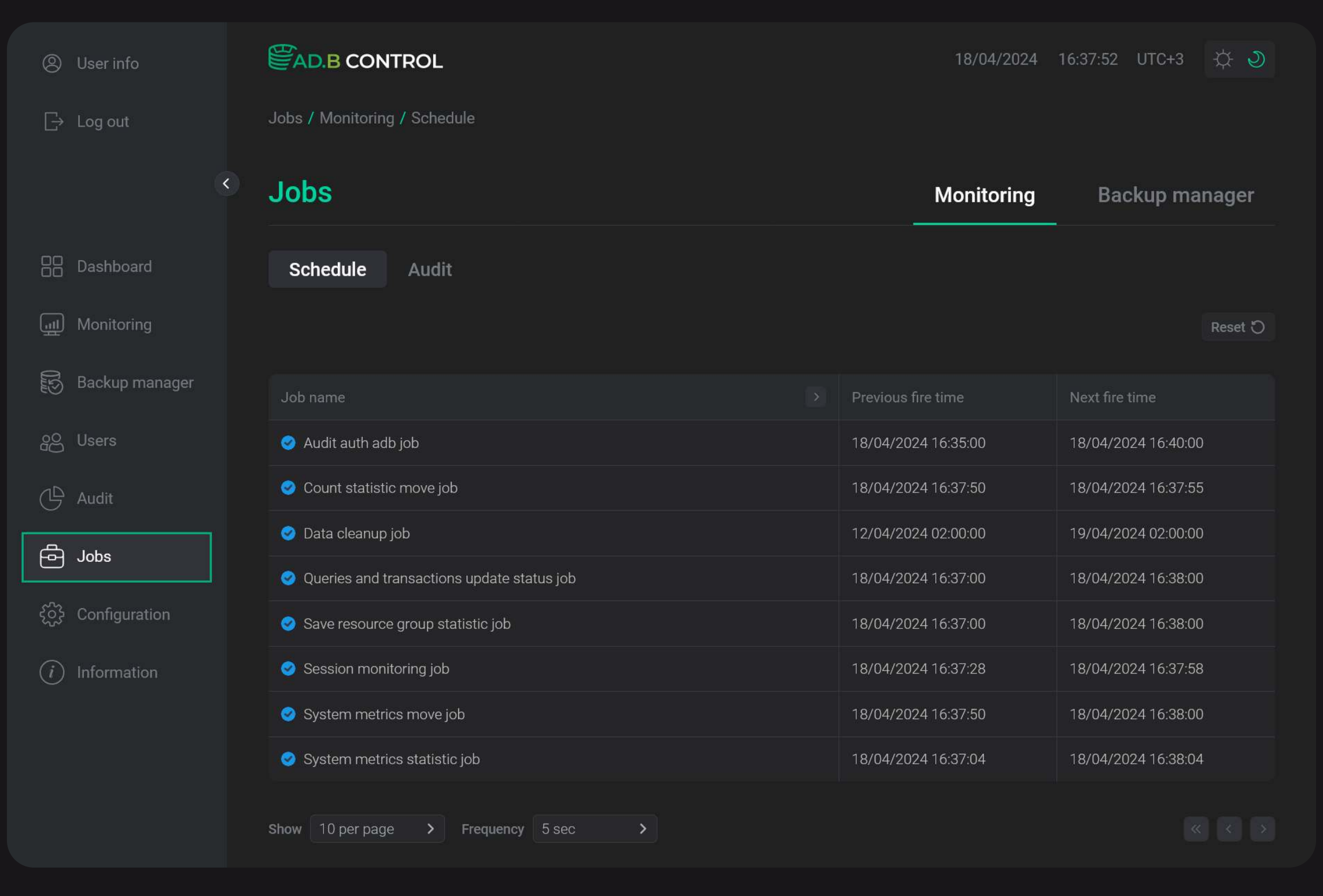
AD.B CONTROL

Contents

To Table of Contents

- Monitoring
  - Schedule
  - Audit
- Backup manager
  - Schedule
  - Audit

The *Jobs* page in the ADB Control web interface displays information on planned and completed system jobs in ADB Control/ADB Backup Manager. This page includes multiple tabs, each of which is described in detail below.



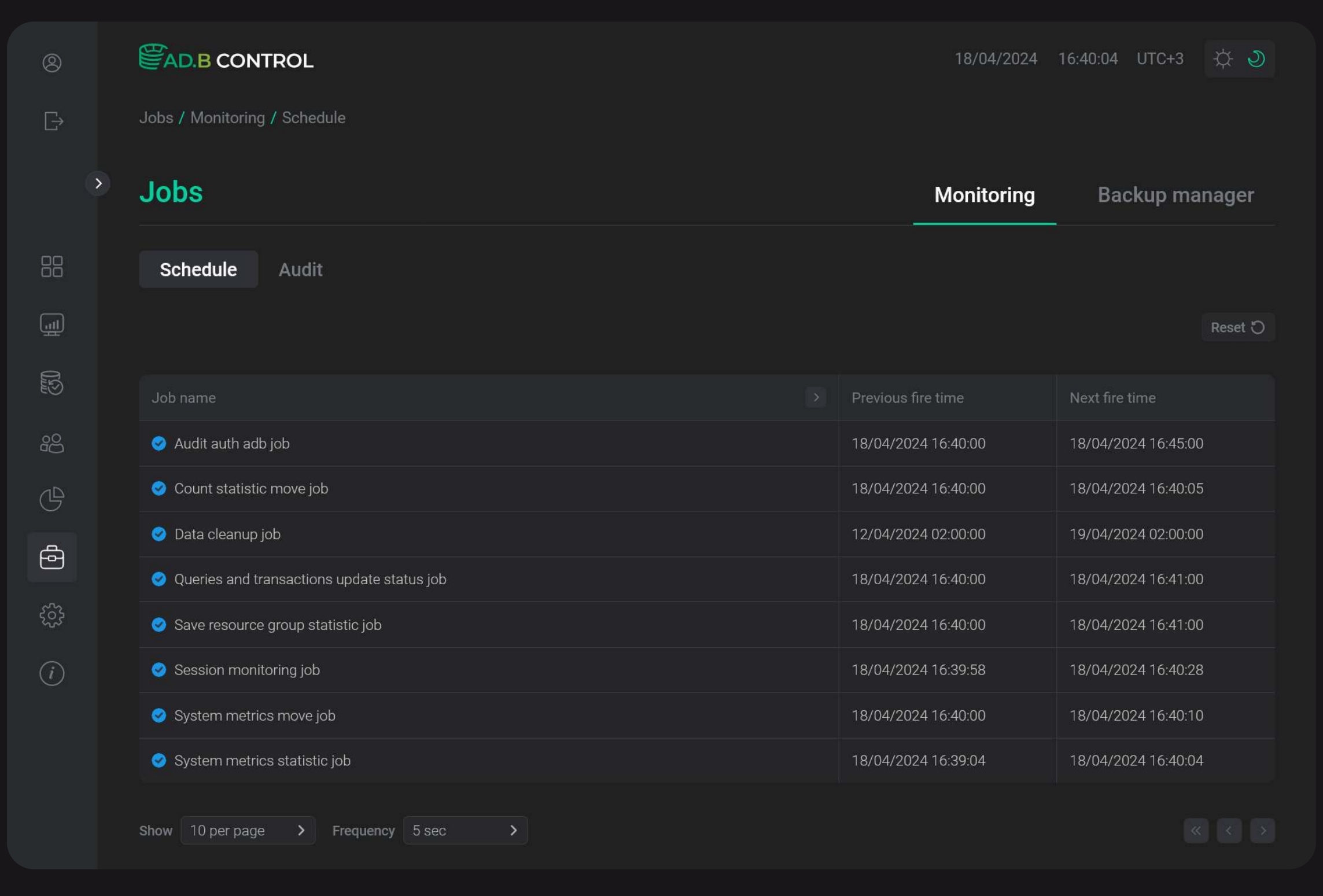
Job name	Previous fire time	Next fire time
Audit auth adb job	18/04/2024 16:35:00	18/04/2024 16:40:00
Count statistic move job	18/04/2024 16:37:50	18/04/2024 16:37:55
Data cleanup job	12/04/2024 02:00:00	19/04/2024 02:00:00
Queries and transactions update status job	18/04/2024 16:37:00	18/04/2024 16:38:00
Save resource group statistic job	18/04/2024 16:37:00	18/04/2024 16:38:00
Session monitoring job	18/04/2024 16:37:28	18/04/2024 16:37:58
System metrics move job	18/04/2024 16:37:50	18/04/2024 16:38:00
System metrics statistic job	18/04/2024 16:37:04	18/04/2024 16:38:04

The Jobs page

Monitoring

Schedule

The ***Monitoring*** → ***Schedule*** tab on the ***Jobs*** page displays scheduled system jobs in ADB Control.



Job name	Previous fire time	Next fire time
Audit auth adb job	18/04/2024 16:40:00	18/04/2024 16:45:00
Count statistic move job	18/04/2024 16:40:00	18/04/2024 16:40:05
Data cleanup job	12/04/2024 02:00:00	19/04/2024 02:00:00
Queries and transactions update status job	18/04/2024 16:40:00	18/04/2024 16:41:00
Save resource group statistic job	18/04/2024 16:40:00	18/04/2024 16:41:00
Session monitoring job	18/04/2024 16:39:58	18/04/2024 16:40:28
System metrics move job	18/04/2024 16:40:00	18/04/2024 16:40:10
System metrics statistic job	18/04/2024 16:39:04	18/04/2024 16:40:04

The Monitoring → Schedule tab

On the tab, there is a table with the following information.

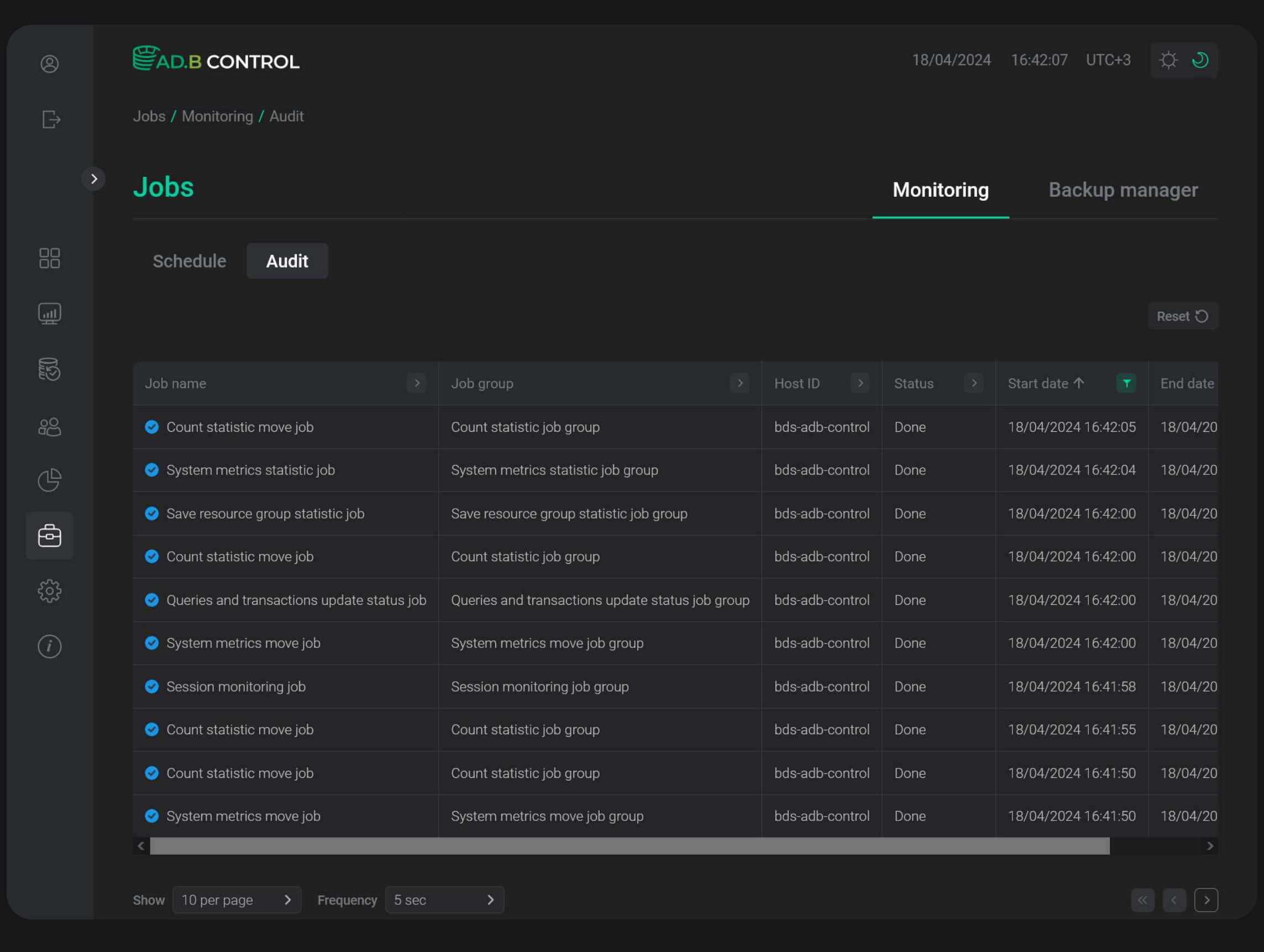
The "Monitoring → Schedule" tab fields

Field	Description
Job name	A system job name. You can get a full job description by pointing a cursor at its name
Previous fire time	A timestamp of the previous job launch in the `DD/MM/YYYY HH:mm:ss` format
Next fire time	A timestamp of the next scheduled job launch in the `DD/MM/YYYY HH:mm:ss` format

In the ***Job name*** column header of the table with a list of scheduled jobs, there is a filter that you can use to select data by the job name. You can search by the first characters entered. To open a filter, click the > icon. The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Audit

The ***Monitoring*** → ***Audit*** tab on the ***Jobs*** page displays all jobs launched in ADB Control during the specified time period. Current day data is displayed by default.



Job name	Job group	Host ID	Status	Start date	End date
Count statistic move job	Count statistic job group	bds-adb-control	Done	18/04/2024 16:42:05	18/04/20
System metrics statistic job	System metrics statistic job group	bds-adb-control	Done	18/04/2024 16:42:04	18/04/20
Save resource group statistic job	Save resource group statistic job group	bds-adb-control	Done	18/04/2024 16:42:00	18/04/20
Count statistic move job	Count statistic job group	bds-adb-control	Done	18/04/2024 16:42:00	18/04/20
Queries and transactions update status job	Queries and transactions update status job group	bds-adb-control	Done	18/04/2024 16:42:00	18/04/20
System metrics move job	System metrics move job group	bds-adb-control	Done	18/04/2024 16:42:00	18/04/20
Session monitoring job	Session monitoring job group	bds-adb-control	Done	18/04/2024 16:41:58	18/04/20
Count statistic move job	Count statistic job group	bds-adb-control	Done	18/04/2024 16:41:55	18/04/20
Count statistic move job	Count statistic job group	bds-adb-control	Done	18/04/2024 16:41:50	18/04/20
System metrics move job	System metrics move job group	bds-adb-control	Done	18/04/2024 16:41:50	18/04/20

The Monitoring → Audit tab

On the tab, there is a table with the following information.

The "Monitoring → Audit" tab fields

Field	Description
Job name	A job name. If the job has the `Error` status, the ***Job name*** table column contains ••• . When you click this button, under the current table row a block appears with a message on the source error, which can be copied by clicking Copy in the top right corner of the block
Job group	A job group name
Host ID	An identifier of the host where the job was launched (in the `<IP address>:<port>` format)
Status	A job status. Possible values: ▪ `Done` ▪ `Error` ▪ `Unknown` ▪ `Running`
Start date	A job start timestamp in the `DD/MM/YYYY HH:mm:ss` format
End date	A job end timestamp in the `DD/MM/YYYY HH:mm:ss` format

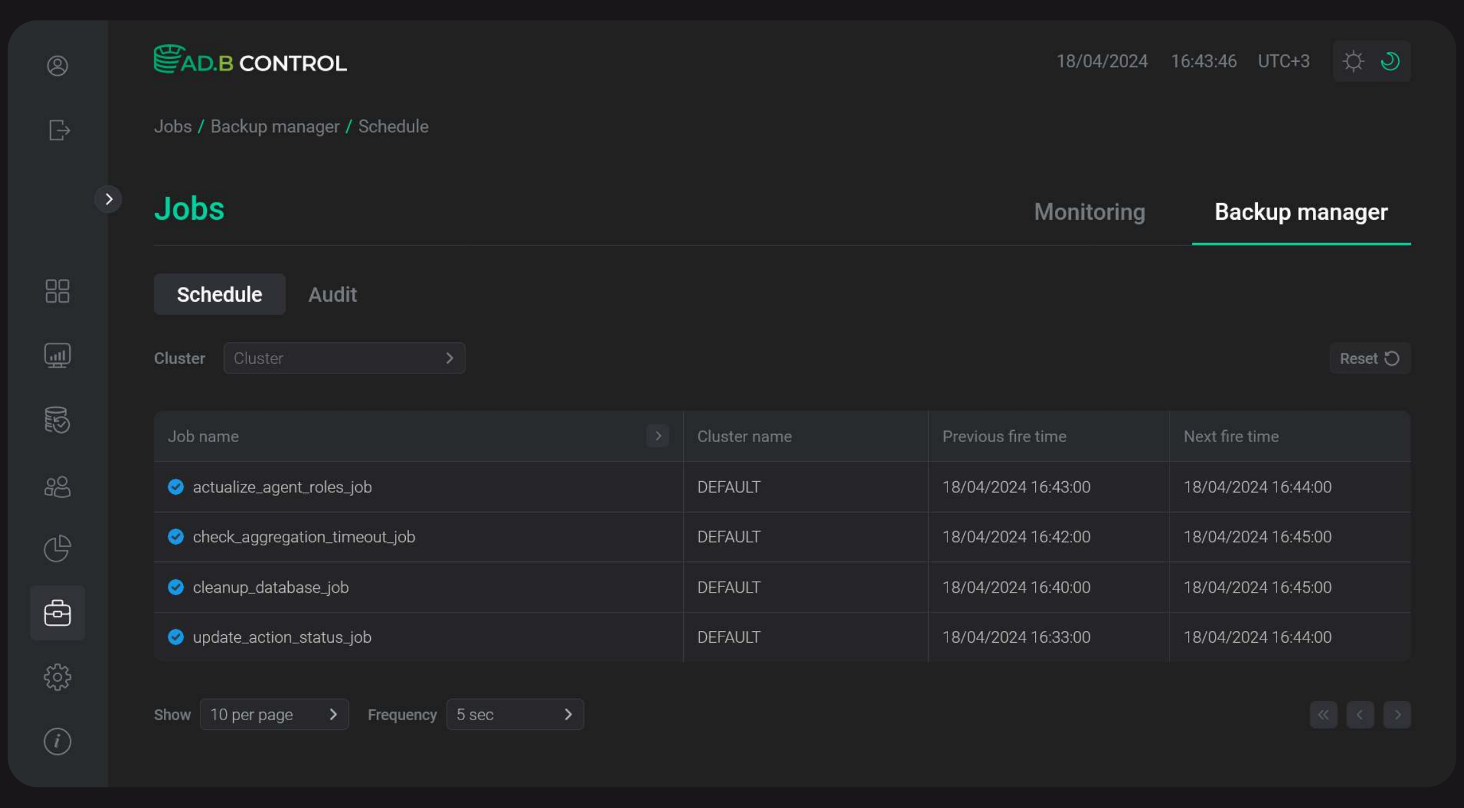
In the column headers of the table with a list of launched jobs, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Job name***), you can select a value from the drop-down list. For some columns (e.g. ***Host ID***), the search value should be entered. For columns that show date and time (e.g. ***Start date***), the time range can be selected from the calendar.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Backup manager

Schedule

The ***Backup manager*** → ***Schedule*** tab on the ***Jobs*** page displays scheduled system jobs in ADB Backup Manager.



Job name	Cluster name	Previous fire time	Next fire time
actualize_agent_roles_job	DEFAULT	18/04/2024 16:43:00	18/04/2024 16:44:00
check_aggregation_timeout_job	DEFAULT	18/04/2024 16:42:00	18/04/2024 16:45:00
cleanup_database_job	DEFAULT	18/04/2024 16:40:00	18/04/2024 16:45:00
update_action_status_job	DEFAULT	18/04/2024 16:33:00	18/04/2024 16:44:00

The Backup manager → Schedule tab

On the tab, there is a table with the following information.

The "Backup manager → Schedule" tab fields

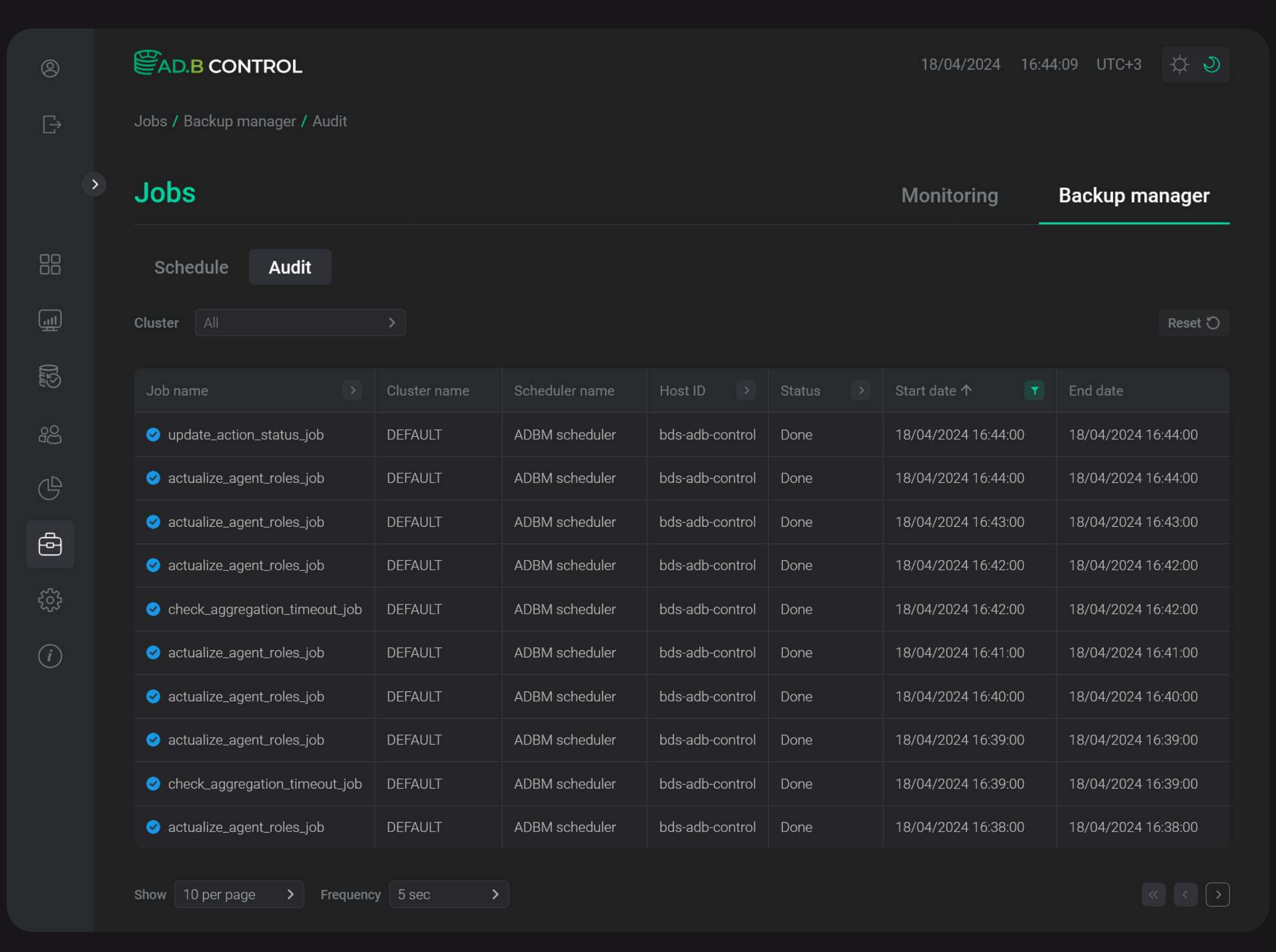
Field	Description
Job name	A system job name. You can get a full job description by pointing a cursor at its name
Cluster name	A name of the cluster for which the job is scheduled. The `DEFAULT` value is used for common jobs that are not related to any specific cluster
Previous fire time	A timestamp of the previous job launch in the `DD/MM/YYYY HH:mm:ss` format
Next fire time	A timestamp of the next scheduled job launch in the `DD/MM/YYYY HH:mm:ss` format

Above the table with a list of scheduled jobs, the ***Cluster*** filter is located. You can use this filter to select the ADB cluster for which you want to display data in the table. By default, all clusters are selected.

In the ***Job name*** column header of the table with a list of scheduled jobs, there is a filter that you can use to select data by the job name. You can search by the first characters entered. To open a filter, click the > icon. The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Audit

The ***Backup manager*** → ***Audit*** tab on the ***Jobs*** page displays all jobs launched in ADB Backup Manager during the specified time period. Current day data is displayed by default.



Job name	Cluster name	Scheduler name	Host ID	Status	Start date	End date
update_action_status_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:44:00	18/04/2024 16:44:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:44:00	18/04/2024 16:44:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:43:00	18/04/2024 16:43:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:42:00	18/04/2024 16:42:00
check_aggregation_timeout_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:42:00	18/04/2024 16:42:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:41:00	18/04/2024 16:41:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:40:00	18/04/2024 16:40:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:39:00	18/04/2024 16:39:00
check_aggregation_timeout_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:39:00	18/04/2024 16:39:00
actualize_agent_roles_job	DEFAULT	ADBM scheduler	bds-adb-control	Done	18/04/2024 16:38:00	18/04/2024 16:38:00

The Backup manager → Audit tab

On the tab, there is a table with the following information.

The "Backup manager → Audit" fields

Field	Description
Job name	A job name. If the job has the `Error` status, the ***Job name*** table column contains ••• . When you click this button, under the current table row a block appears with a message on the source error, which can be copied by clicking Copy in the top right corner of the block
Cluster name	A name of the cluster where the job was launched. The `DEFAULT` value is used for common jobs that are not related to any specific cluster
Scheduler name	A scheduler name
Host ID	A name of the host where the job was launched
Status	A job status. Possible values: ▪ `Done` ▪ `Error` ▪ `Unknown` ▪ `Running`
Start date	A job start timestamp in the `DD/MM/YYYY HH:mm:ss` format
End date	A job end timestamp in the `DD/MM/YYYY HH:mm:ss` format

Above the table with a list of launched jobs, the ***Cluster*** filter is located. You can use this filter to select the ADB cluster for which you want to display data in the table. By default, all clusters are selected.

In the column headers of the table with a list of launched jobs, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Job name***), you can select a value from the drop-down list. For some columns (e.g. ***Host ID***), the search value should be entered. For columns that show date and time (e.g. ***Start date***), the time range can be selected from the calendar.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

View information on components

Daria Barysheva

Contents

To Table of Contents

- ADB Control
  - UI backend server
  - Agent
  - Backend server
  - Scheduler
- Backup manager

The ***Information*** page in the ADB Control web interface displays the state of the ADB Control and ADBM components. This page includes multiple tabs, each of which is described in detail below.

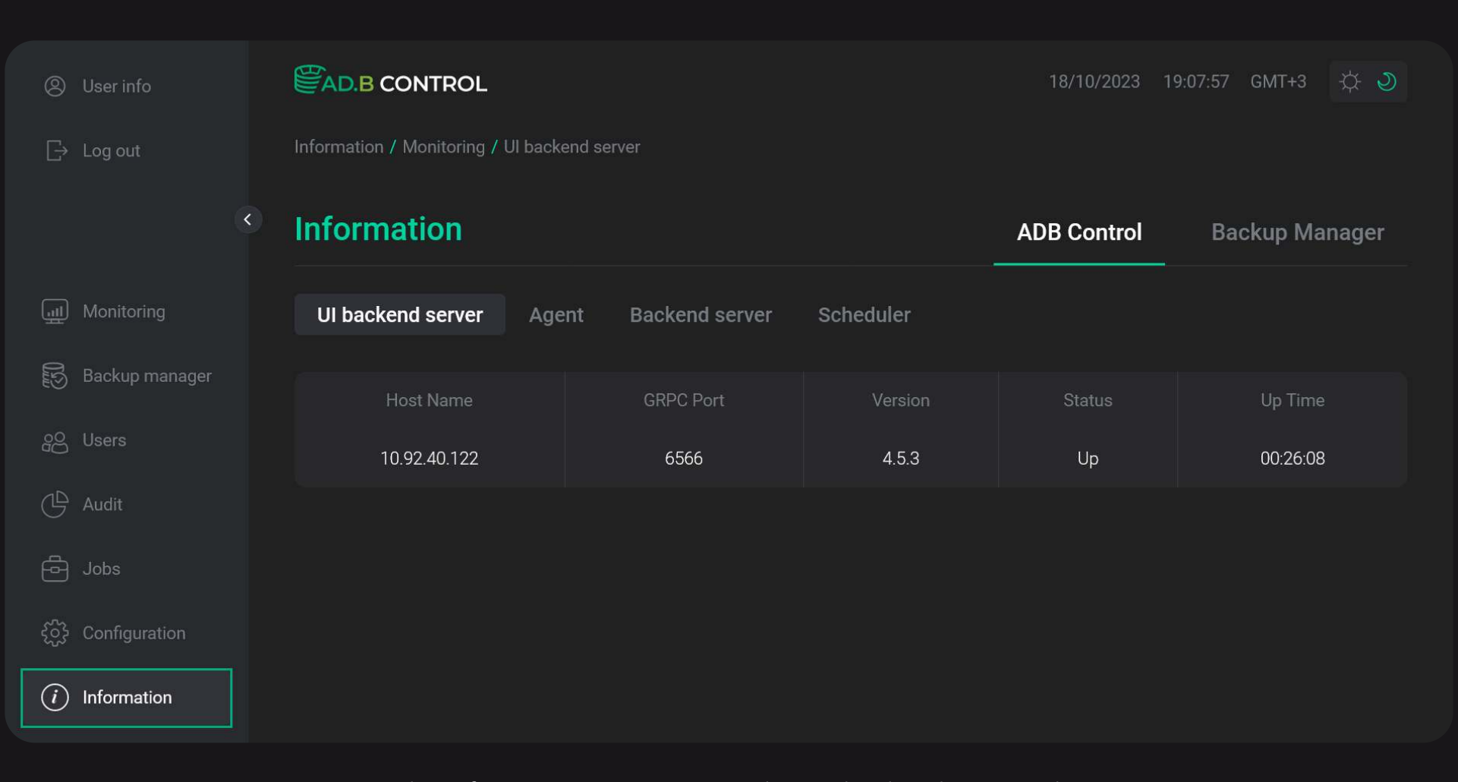
ADB Control

On the ***Information*** → ***ADB Control*** page, the status of ADB Control components is displayed.

> **NOTE**
>
> For more information on the ADB Control architecture, see Arenadata DB Control overview.

UI backend server

The ***UI backend server*** tab on the ***Information*** → ***ADB Control*** page contains information on the backend server of the ADB Control web interface (UI backend server).



Host Name	GRPC Port	Version	Status	Up Time
10.92.40.122	6566	4.5.3	Up	00:26:08

The Information → ADB Control → UI backend server tab

Fields of the "Information → ADB Control → UI backend server" tab

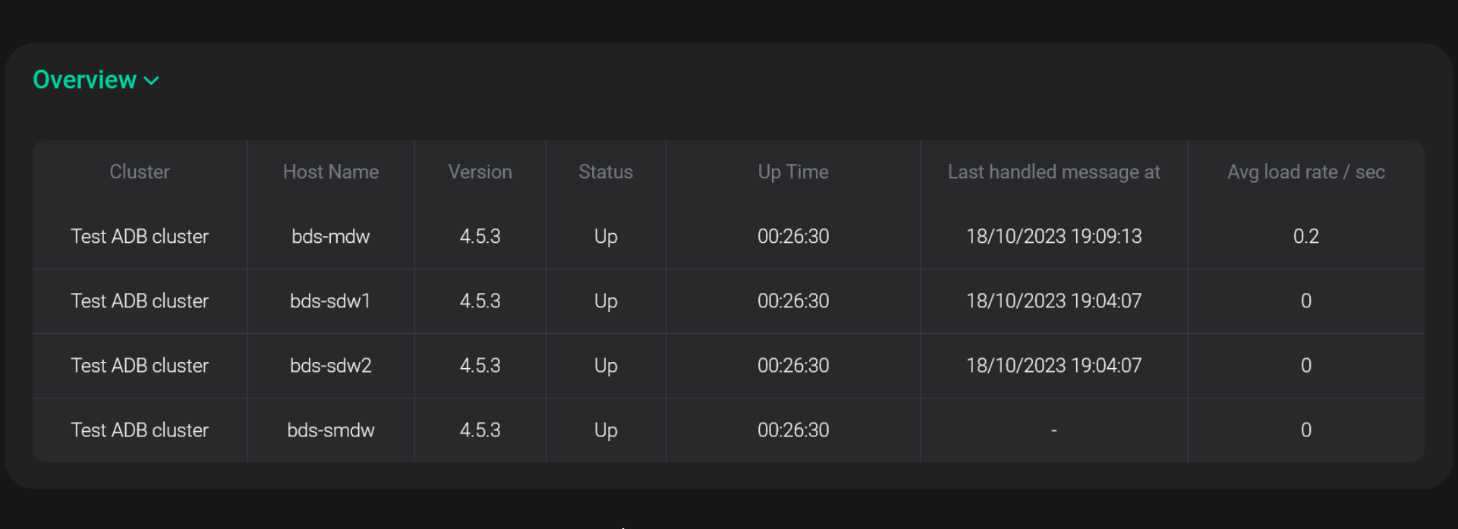
Field	Description
Host Name	An IP address of the host where UI backend server is deployed
GRPC Port	The gRPC port that is used by UI backend server
Version	A UI backend server version
Status	A UI backend server status. The possible value is `UP`, which means that UI backend server works
Up Time	The time since the last UI backend server startup (in hours, minutes, seconds)

Agent

The ***Agent*** tab on the ***Information*** → ***ADB Control*** page displays information on ADB Control agents. The tab consists of the following sections:

- ***Overview***
- ***Aggregated load ratio info***

The ***Overview*** section displays the general information on all ADB Control agents installed on hosts of the ADB clusters that are connected to the monitoring system.



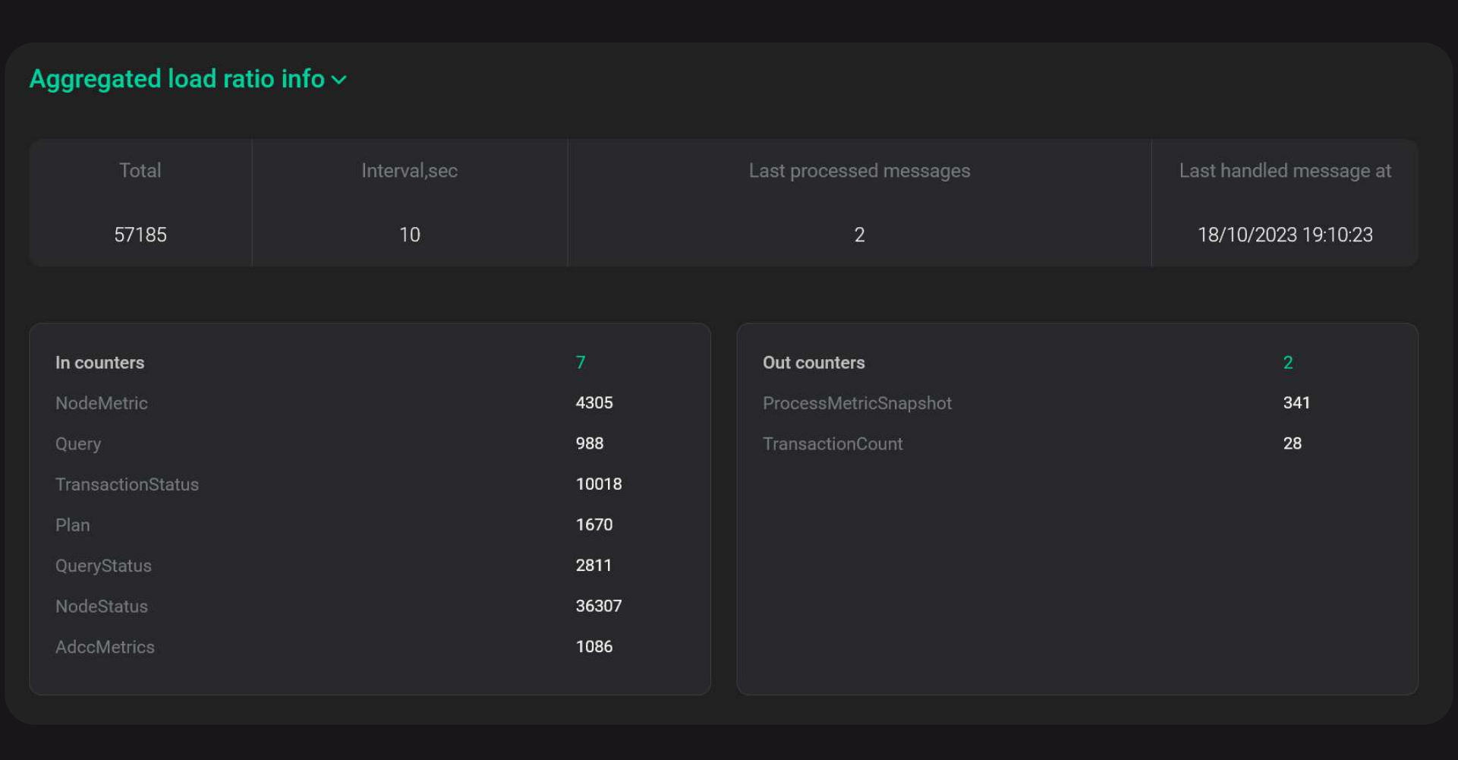
Cluster	Host Name	Version	Status	Up Time	Last handled message at	Avg load rate / sec
Test ADB cluster	bds-mdw	4.5.3	Up	00:26:30	18/10/2023 19:09:13	0.2
Test ADB cluster	bds-sdw1	4.5.3	Up	00:26:30	18/10/2023 19:04:07	0
Test ADB cluster	bds-sdw2	4.5.3	Up	00:26:30	18/10/2023 19:04:07	0
Test ADB cluster	bds-smdw	4.5.3	Up	00:26:30	-	0

The Overview section

Fields of the Overview section

Field	Description
Cluster	A name of the ADB cluster
Host Name	A name of the host where the ADB Control agent is deployed
Version	A version of the ADB Control agent
Status	A status of the ADB Control agent. The possible value is `Up`, which means that the agent works. Stopped agents are not displayed in the table (for example, when a host is disabled)
Up Time	The time since the last agent startup in hours, minutes, seconds
Last handled message at	A timestamp of the last processed protobuf message (received from segments) in the `DD/MM/YYYY HH:mm:ss` format
Avg load rate / sec	An average number of incoming protobuf messages per second. It is calculated as a ratio of the number of messages received by the agent during the time that is defined in the Interval,sec field to the value of that period

The ***Aggregated load ratio info*** displays the aggregated statistics on the processing of protobuf messages (received from segments) by all ADB Control agents.



Total	Interval,sec	Last processed messages	Last handled message at
57185	10	2	18/10/2023 19:10:23

In counters	7
NodeMetric	4305
Query	988
TransactionStatus	10018
Plan	1670
QueryStatus	2811
NodeStatus	36307
AdccMetrics	1086

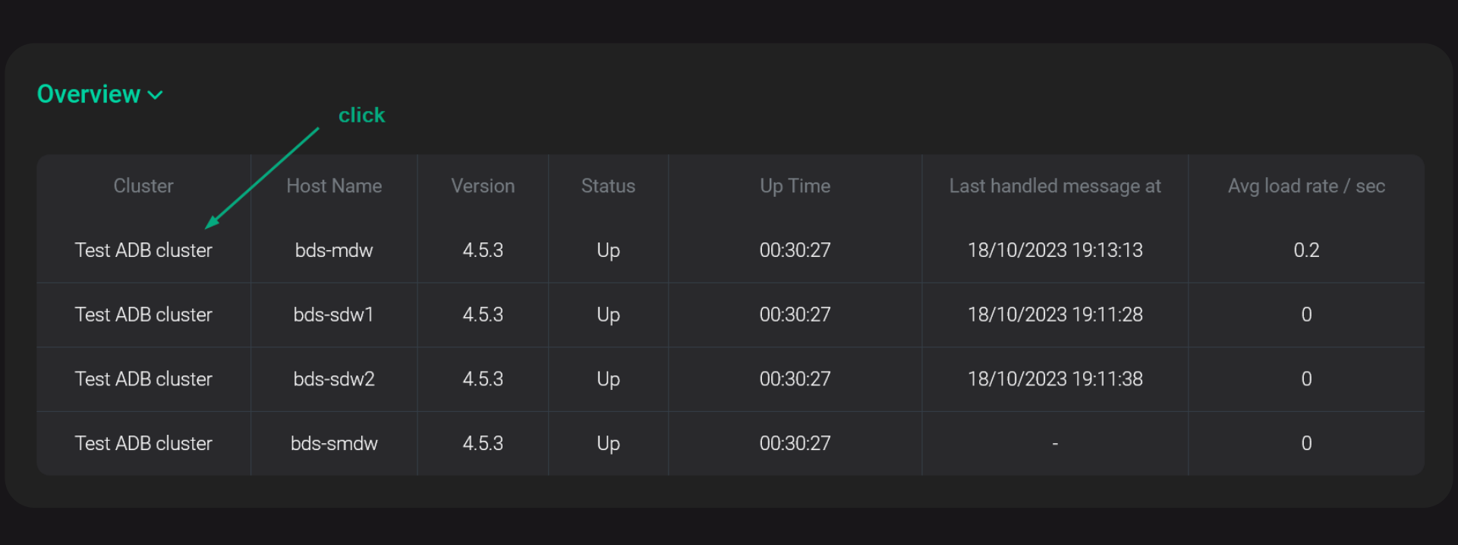
Out counters	2
ProcessMetricSnapshot	341
TransactionCount	28

The Aggregated load ratio info section

Fields of the "Aggregated load ratio info" section

Field	Description
Total	A total number of received protobuf messages (since agent startup)
Interval,sec	A time interval between counters sampling (in seconds). By default, `10`. You can change the limit via the `eureka.update.counter.interval` configuration parameter
Last processed messages	A number of processed messages during the last ***Interval,sec*** time period
Last handled message at	A timestamp of the last processed protobuf message (in the `DD/MM/YYYY HH:mm:ss` format)
In counters	A total number of protobuf messages received from segments by ADB Control agents (since agent startup). Messages are grouped by query type (counter). The number of types for incoming messages is displayed at the top of the list
Out counters	A total number of protobuf messages sent by ADB Control agents to Backend server (since agent startup). Messages are grouped by query type (counter). The number of types for outcoming messages is displayed at the top of the list

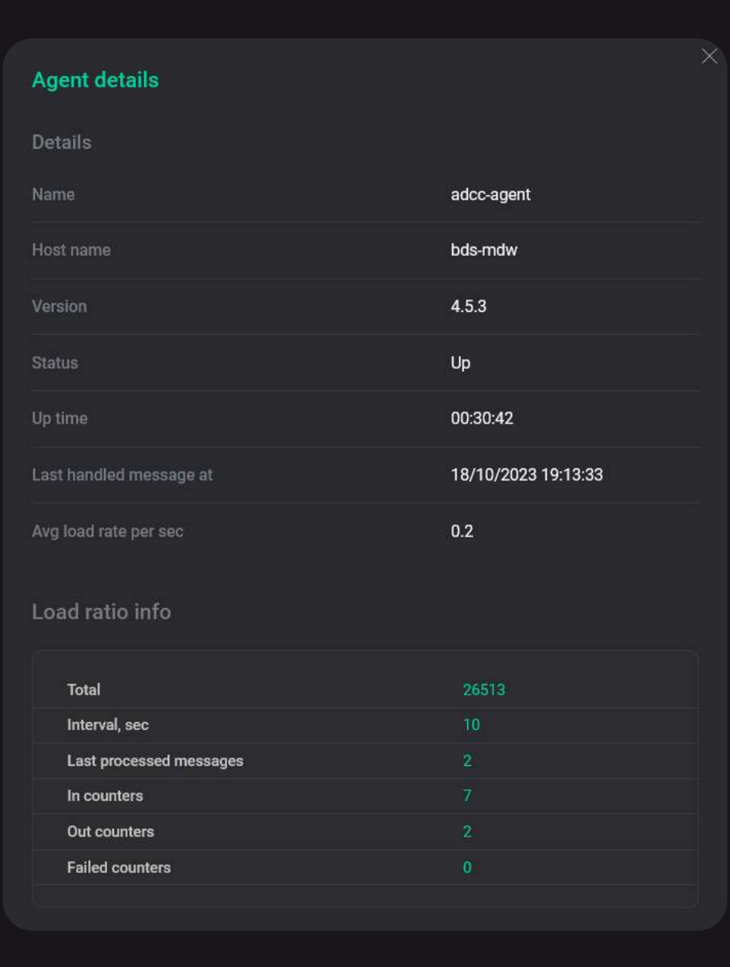
The information that is described above for the Overview and Aggregated load ratio info sections can be obtained for a specific ADB Control agent. To view the agent details, click the corresponding row in the ***Overview*** section.



Cluster	Host Name	Version	Status	Up Time	Last handled message at	Avg load rate / sec
Test ADB cluster	bds-mdw	4.5.3	Up	00:30:27	18/10/2023 19:13:13	0.2
Test ADB cluster	bds-sdw1	4.5.3	Up	00:30:27	18/10/2023 19:11:28	0
Test ADB cluster	bds-sdw2	4.5.3	Up	00:30:27	18/10/2023 19:11:38	0
Test ADB cluster	bds-smdw	4.5.3	Up	00:30:27	-	0

Switch to the detailed agent information

All fields displayed for the selected agent are almost identical to those described above. The ***Failed counters*** field displays the number of out counters, which messages failed to be sent. Also note that an average load rate and all counters are calculated and displayed for the selected agent.



Details

Name	adcc-agent
Host name	bds-mdw
Version	4.5.3
Status	Up
Up time	00:30:42
Last handled message at	18/10/2023 19:13:33
Avg load rate per sec	0.2

Load ratio info

Total	26513
Interval, sec	10
Last processed messages	2
In counters	7
Out counters	2
Failed counters	0

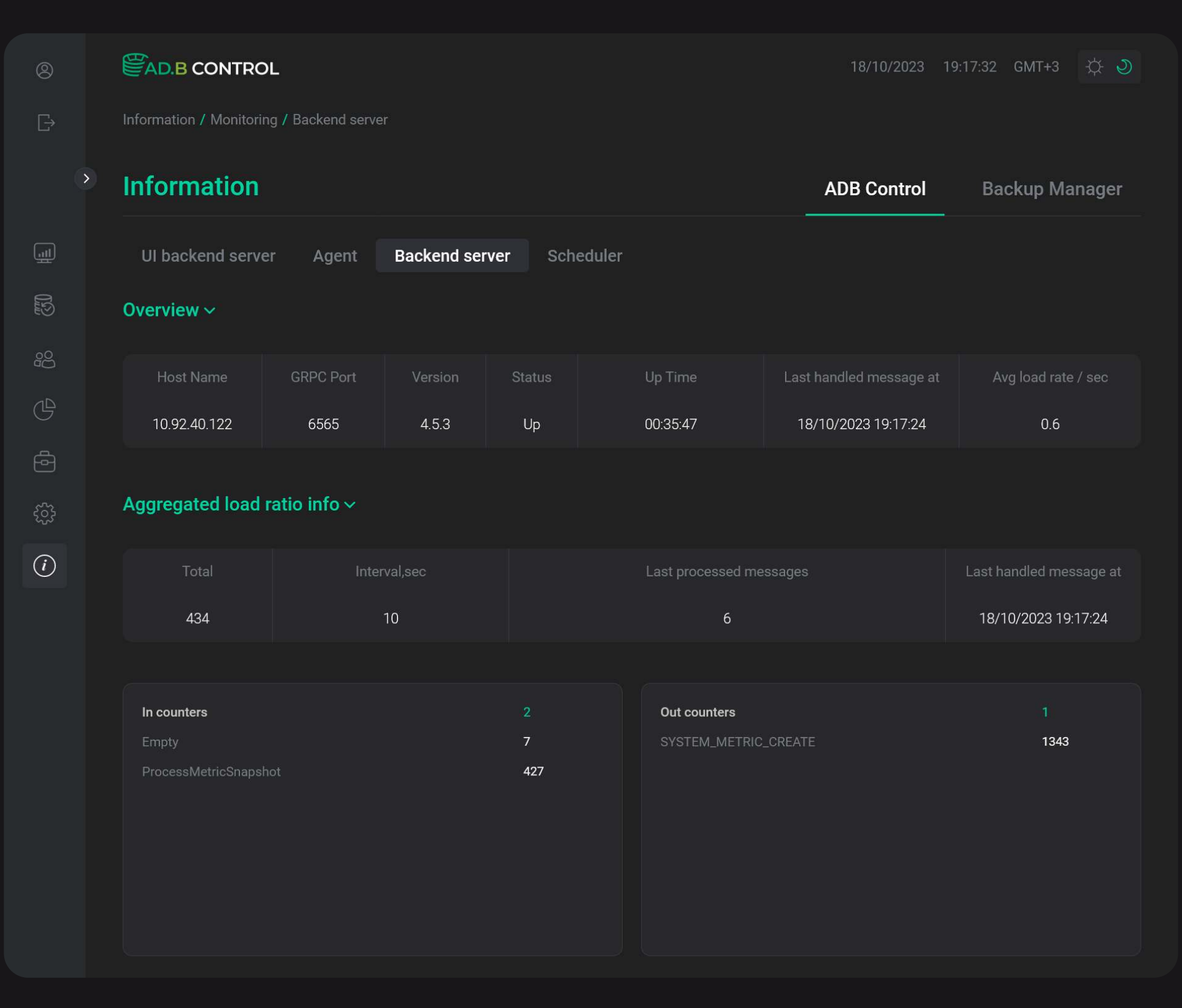
Window with agent details

Backend server

The ***Backend server*** tab on the ***Information*** → ***ADB Control*** page displays information on the ADB Control Backend server. The tab functionality is completely identical to the Agent tab described above. The only difference is that for Backend server, ADB Control agents are the source of incoming messages, and PostgreSQL is the receiver of outcoming requests (***Query Database*** on the ADB Control architectural schema).

You can also view the Backend server details in a separate window by clicking the row in the ***Overview*** section.

> **NOTE**
>
> The counter names in the ***In counters*** and ***Out counters*** sections may not be the same because Backend server aggregates and otherwise processes incoming messages and then creates new objects.



Host Name	GRPC Port	Version	Status	Up Time	Last handled message at	Avg load rate / sec
10.92.40.122	6565	4.5.3	Up	00:35:47	18/10/2023 19:17:24	0.6

Total	Interval,sec	Last processed messages	Last handled message at
434	10	6	18/10/2023 19:17:24

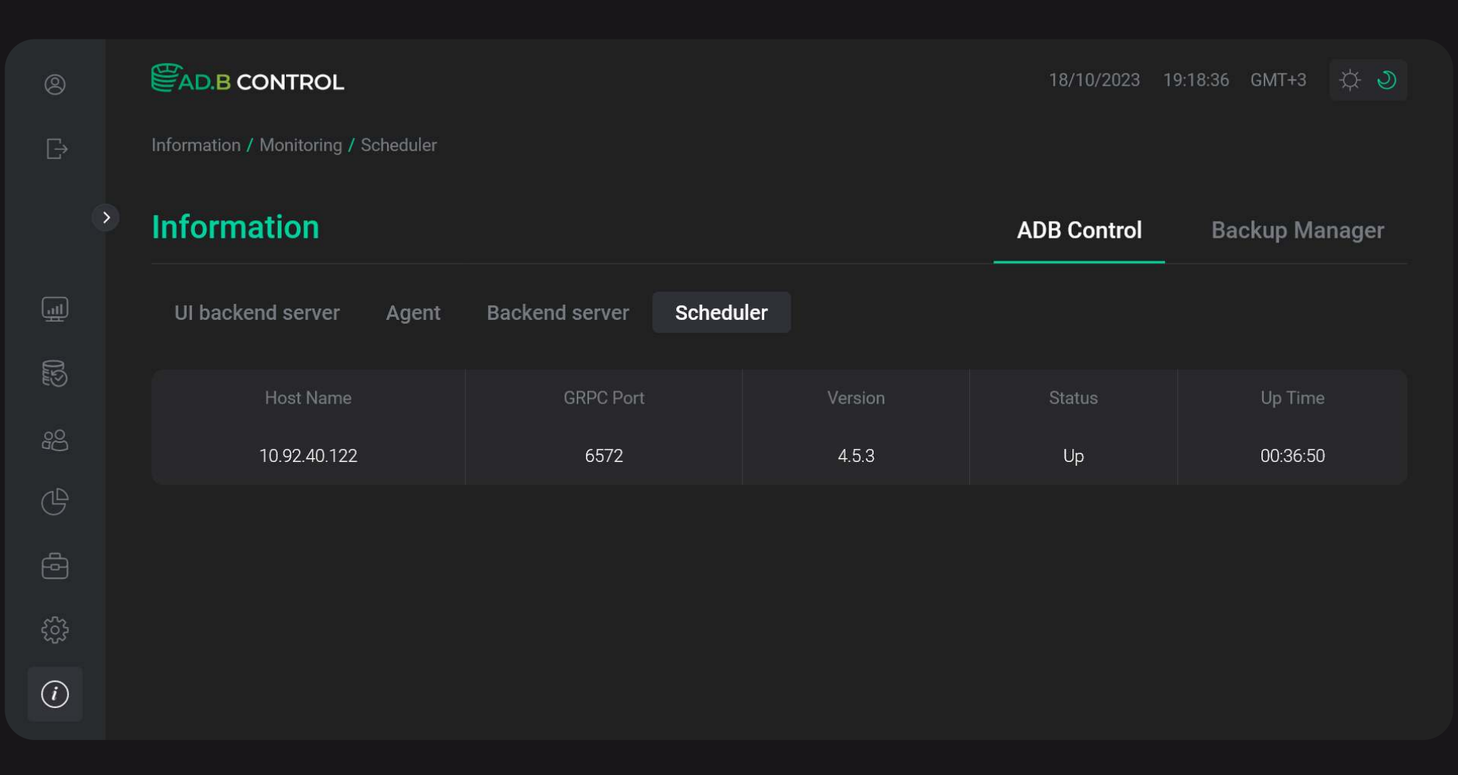
In counters	2
Empty	7
ProcessMetricSnapshot	427

Out counters	1
SYSTEM_METRIC_CREATE	1343

The Information → ADB Control → Backend server tab

Scheduler

The ***Scheduler*** tab on the ***Information*** → ***ADB Control*** page contains information on the ADB Control Scheduler.



Host Name	GRPC Port	Version	Status	Up Time
10.92.40.122	6572	4.5.3	Up	00:36:50

The Information → ADB Control → Scheduler tab

Fields of the Information → ADB Control → Scheduler tab

Field	Description
Host Name	An IP address of the host where Scheduler is deployed
GRPC Port	The gRPC port that is used by Scheduler
Version	A Scheduler version
Status	A Scheduler status. The possible value is `UP`, which means that Scheduler works
Up Time	The time since the last Scheduler startup (in hours, minutes, seconds)

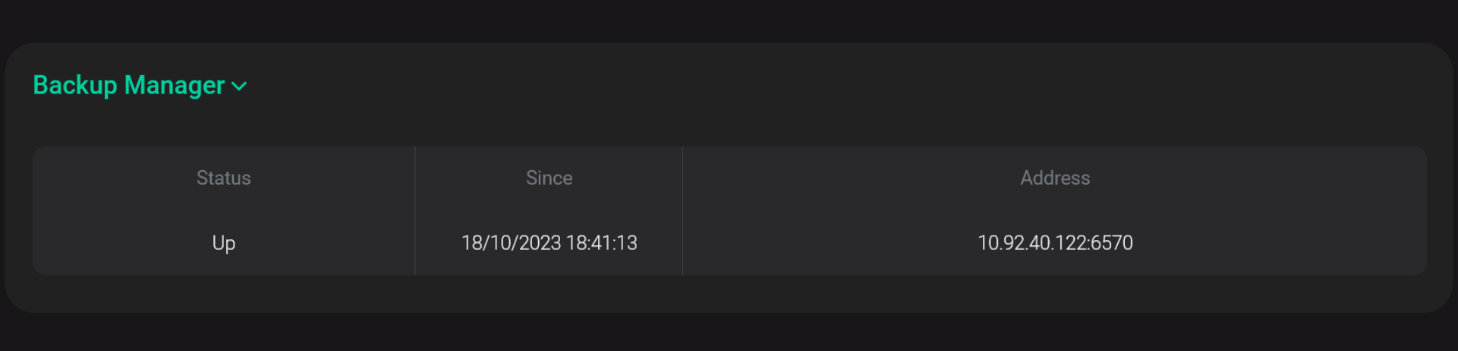
Backup manager

> **NOTE**
>
> For more information on the ADBM architecture, see Arenadata DB Backup Manager overview.

The ***Information*** → ***Backup manager*** page displays the status of ADBM and its agents. The page consists of two sections:

- ***Backup manager***
- ***Database & Agents***

The ***Backup manager*** section displays the ADBM state.



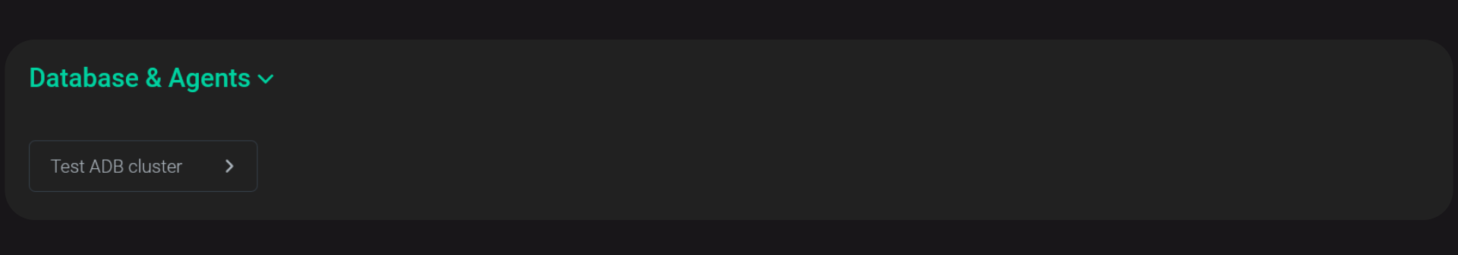
Status	Since	Address
Up	18/10/2023 18:41:13	10.92.40.122:6570

The Backup manager section

Fields of the "Backup manager" section

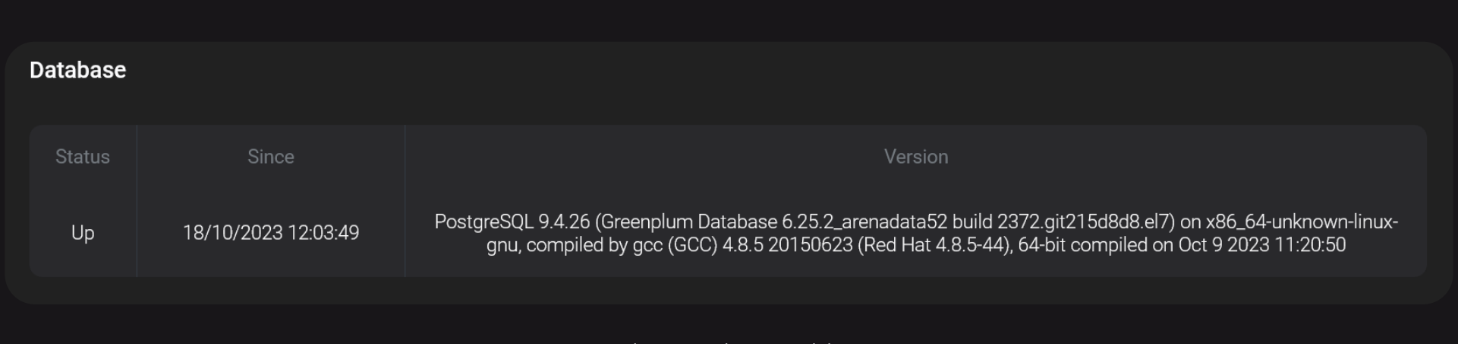
Field	Description
Status	An ADBM status. The possible value is `Up`, which means that ADBM works. If ADBM is stopped, the ***Backup manager*** page will be empty
Since	A timestamp of the last ADBM startup in the `DD/MM/YYYY HH:mm:ss` format
Address	An address of the host where ADBM is deployed (in the `<IP address>:<port>` format)

The ***Database & Agents*** section displays information on the ADB database that is connected for backing up and the ADBM agents that are installed on cluster hosts. The data is displayed for the ADB cluster that is selected in the filter directly under the section name.



Filter by clusters

The ***Database & Agents*** section includes two tables: ***Database*** and ***Agents***. The ***Database*** table displays information on the ADB database.



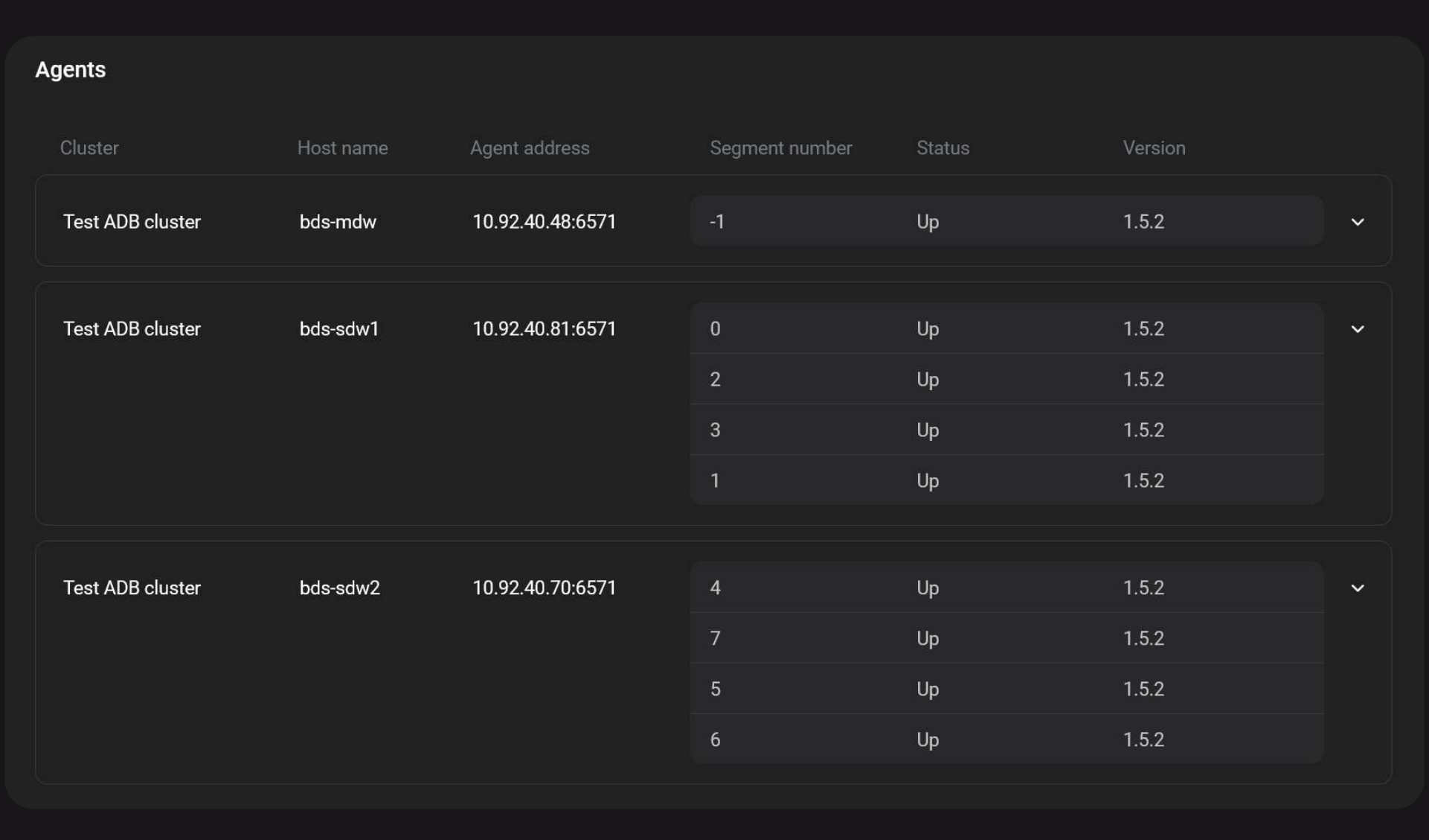
Status	Since	Version
Up	18/10/2023 12:03:49	PostgreSQL 9.4.26 (Greenplum Database 6.25.2_arenadata52 build 2372.git215d8d8.el7) on x86_64-unknown-linux-gnu, compiled by gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44), 64-bit compiled on Oct 9 2023 11:20:50

The Database table

Fields of the Database table

Field	Description
Status	A database status. Possible values: `Up` — the database works. `Down` — the database is stopped. In this case, the Agents table will be empty.
Since	A timestamp of the last database startup in the `DD/MM/YYYY HH:mm:ss` format
Version	A detailed description of the ADB build including PostgreSQL and Greengage DB Database versions

The ***Agents*** table displays information on ADBM agents.



Cluster	Host name	Agent address	Segment number	Status	Version
Test ADB cluster	bds-mdw	10.92.40.48:6571	-1	Up	1.5.2
Test ADB cluster	bds-sdw1	10.92.40.81:6571	0	Up	1.5.2
			2	Up	1.5.2
			3	Up	1.5.2
			1	Up	1.5.2
Test ADB cluster	bds-sdw2	10.92.40.70:6571	4	Up	1.5.2
			7	Up	1.5.2
			5	Up	1.5.2
			6	Up	1.5.2

The Agents table

Fields of the Agents table

Field	Description
Cluster	A name of the ADB cluster
Host name	A name of the host where the ADBM agent is deployed
Agent address	An address of the host where the ADBM agent is deployed (in the `<IP address>:<port>` format)
Segment number	A unique identifier of the primary segment on the cluster. Refers to `gp_segment_configuration.content`. For more information, see Tables overview
Status	An ADBM agent status. Possible values: `Up` — the agent works. `Down` — the agent is in the process of stopping (for example, if the host is being disconnected). The status is displayed for a short time, after which a request is sent to the Service Registry to delete the agent metadata and the agent entry is removed from the table.
Version	A version of the ADBM agent

Arenadata DB Backup Manager overview

Daria Barysheva

Contents

To Table of Contents

- Features
- Architecture
- Concepts
  - Backup
  - WAL
  - Restore point
  - Timeline
- Quick start

Features

Arenadata DB Backup Manager (ADBM) is a fault-tolerant system for ADB binary backups management, which is built on the base of pgbackrest.

ADBM allows you to perform the following functions:

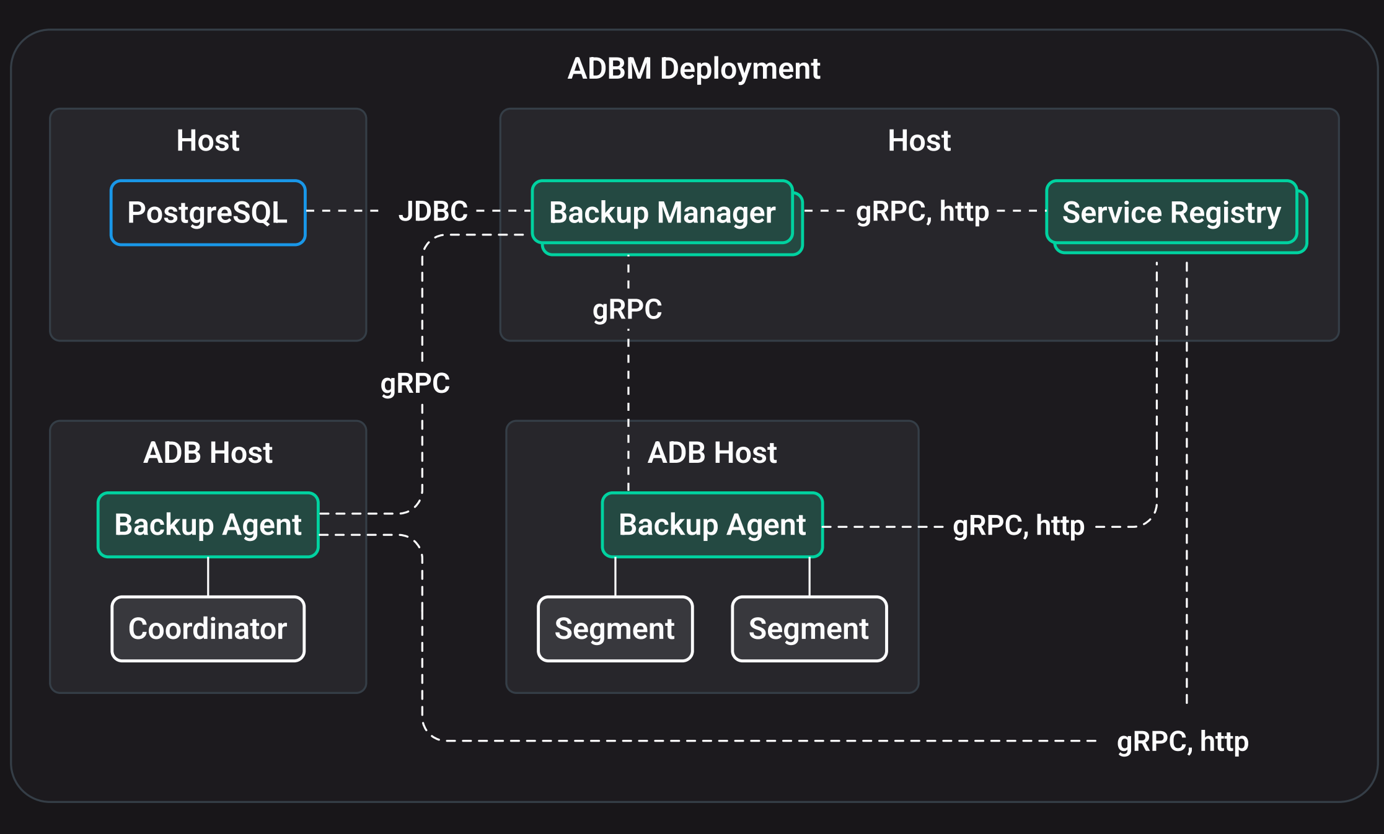
- Configure backup policies including separate schedules for different backup types.
- Create database backups according to the configured schedule and on demand.
- Maintain a list of backups with the ability to search for them and view details (including a cluster topology).
- Cleanup backups according to existing policies on a schedule and manually.
- Create restore points according to the configured schedule and manually.
- Restore databases at the moment of the selected restore point.
- Log all system actions with the ability to view details of the failed actions.
- Work with multiple ADB clusters of different types (with/without standby and mirroring).
- Use S3 and Posix compatible repositories to store backups.

NOTE

- ADBM is available in the ADB Enterprise Edition.
- Currently, ADBM can be installed only offline.

Architecture

The high-level architecture view of ADBM is shown below.



ADBM architecture

ADBM architecture is based on the following components:

- **Backup Manager**. Orchestrates cluster actions and performs background work according to the configured schedules: creates backups and recovery points, cleans up old and invalid data, etc. Backup Manager and ADB clusters interact via Backup Agents.
- **Backup Agent**. Agents are installed one per ADB cluster host. They are responsible for segment data management:
  - Run `pgbackrest` commands.
  - Manage `pgbackrest` configuration.
  - On coordinator host — manage ADB cluster operations (for example, cluster restart).
- **Service Registry**. Responsible for service discovery in ADBM. Due to Service Registry, ADBM discovers available agents (there is no pre-configured agent map in ADBM). And Backup Agents, in turn, find necessary ADBM services to send responses. Since Backup Manager can work with multiple ADB clusters at the same time — when you add a cluster, new agents are also registered via Service Registry.
- **PostgreSQL**. It is used to store the following data:
  - Cluster-wide locks to ensure the exclusivity of the actions to be launched.
  - Information about the current active action state. If Backup Manager stops or fails — the current action metadata can be retrieved from PostgreSQL.
  - Historical data on backups and configurations.

  By default, it is deployed via the Docker container. The external PostgreSQL database can also be used.

NOTE

- To use external PostgreSQL, fill in the ***External database parameters*** section parameters when configuring the ADBM service during ADB installation (for more details, see Configuration parameters → ADBM → External database parameters).
- Starting with ADBM 1.7.3, you need to create users in the external database and grant them permissions manually. For more details, see Configure an external database for ADBM.

Concepts

ADBM implements **point-in-time recovery (PITR)** — the ability to restore databases to the selected point in time. This approach is based on several concepts, which are summarized below.

Backup

Backup is a consistent copy of an ADB cluster that can be used to restore databases in case of hardware and other failures. ADBM supports the following backup types:

- **Full**. When a full backup is created, the entire database content is backed up. The first backup running in ADBM within each timeline always has the full type, even if according to the schedule (or user choice — when launching a backup manually) the first should be a differential or incremental backup. A full backup can be used to restore data directly as it does not depend on any external files from other backups. The advantage of full backups is that all files are quickly restored (compared to other types). However, it is not recommended to create full backups regularly (hourly or daily) as they take a lot of time to be generated and occupy significant disk space.
- **Differential**. A differential backup contains only those database files that have changed since the last full backup was launched. In comparison with full backups, differential backups are faster and require less disk space. However, to restore data from a backup of this type, it is necessary to copy both its contents and files from the last full backup.
- **Incremental**. An incremental backup contains only those database files that have changed since the last backup of any other type was launched (full, differential, or incremental). Of all types, incremental backups are the fastest and occupy minimal disk space. However, their restoring takes longer because it is necessary to extract files from the last full and, if available, differential backup, and then apply all incremental backups sequentially.

WAL

Write-ahead log (WAL) is a standard mechanism to ensure that no committed changes are lost. All changes to data files (that contain tables, indexes, etc.) are written sequentially to the WAL. Afterwards, a background process writes the changes into the main database cluster files. In case of failures, the WAL can be replayed to make the database consistent.

WAL is broken up into individual 64 MB files called **segments**. Every record in a segment has a 16-digit logical sequence number (LSN) that is used to find this record by offset in the current segment file. A segment name, in turn, consists of the timeline number and LSN of the first segment record.

Using WAL significantly reduces the number of disk writes since there is no need to flush data pages to disk on every transaction commit. Only WAL files should be flushed to disk to guarantee that a transaction is committed.

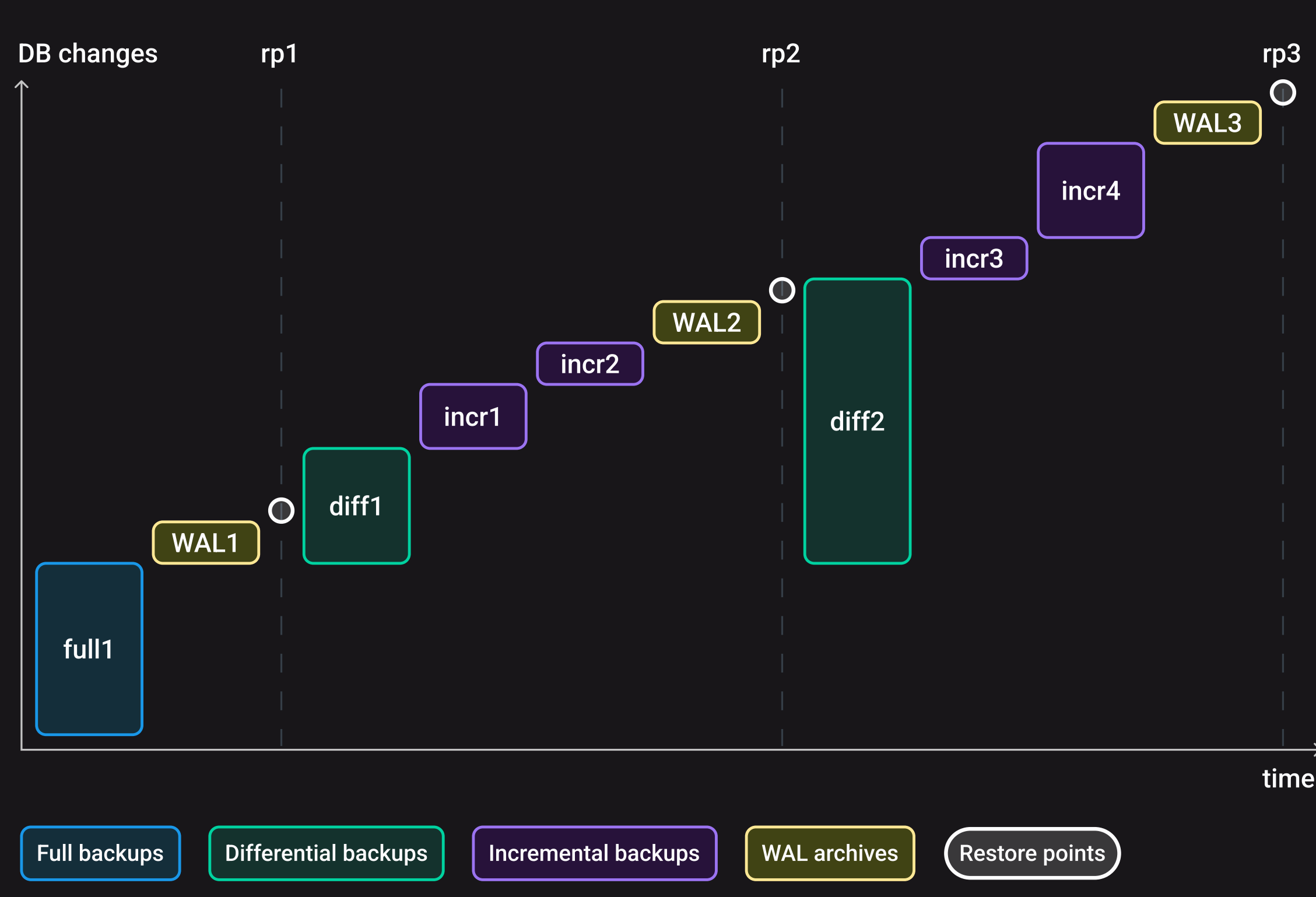
WAL plays a central role in the mechanism of point-in-time recovery (PITR) that is used in ADBM. By archiving the WAL data, ADBM allows you to restore databases at moment of any restore point covered by the available WAL data. In fact, the restore process requires two steps:

- Install a prior backup (or backups).
- Replay the WAL records, written after this backup, up to the specified named point.

Restore point

Restore point means a named recovery point, which is the minimum unit of data consistency granularity in ADB clusters. After a restore point is created (on a schedule or manually), you can restore the database state at the moment of the point creation. For each existing restore point, ADBM stores information on which backups should be used and which WAL segments should be reproduced to return the database state at the creation time of this point.

To better understand the mechanism of restore points, look at the figure below.



Example with restore points and different backup types

The following table explains which backups and WAL files will be used if you select any of the shown restore points when running the restore action.

Logic of restore points

Selected restore point	Objects to be used for recovery
rp1	full1, WAL1
rp2	full1, diff1, incr1, incr2, WAL2
rp3	full1, diff2, incr3, incr4, WAL3

Timeline

Timeline is a mechanism that is used in ADBM to distinguish the WAL series generated after the database recovery at the specified restore point from those created in the original database history (before the restore action application).

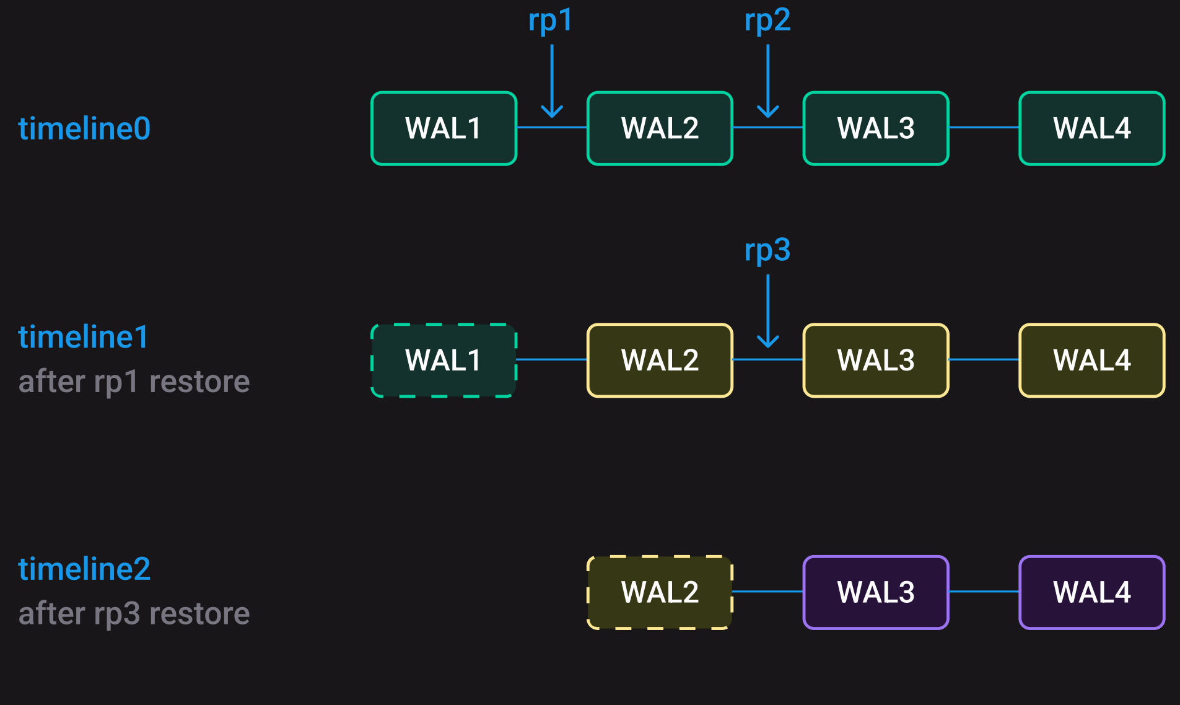
Suppose that to recover the database at the creation time of some restore point, after loading one or more backups it is necessary to reproduce `WAL1` (the number is used to simplify). After the archive recovery completes, WAL will continue to populate. If the WAL archives are added to the same directory, the subsequent WAL records will overwrite those that exist in the database original history (`WAL2`, `WAL3` and so on). This, in turn, will result in the failure of database recovery for other restore points added before recovery.

To avoid the conflicts described, ADBM initializes a new timeline number after each data recovery to identify post-recovery WAL archives. Timelines are numbered starting with `0`, with each successful restore action increasing the number by one. This number is recorded in the names of the directories that store WAL archives and backups divided by segments (so-called **stanza**). With each data recovery, all necessary directories are created with a new timeline number in the name, and all subsequent WAL records are placed into new directories without overwriting files in the directories that have the previous number.

IMPORTANT

Since newly created directories initially do not contain backups, the first backup after each data restore action will be of type `full`, even if a differential or incremental backup should be launched first according to the schedule (or user choice — when launching a backup manually).

The following figure illustrates the timeline logic in a simplified way. After you restore data at the moment of the `rp1` point, `WAL1` is replayed and `timeline1` is created. Subsequent WAL records are added to new directories. The next restore action creates `timeline2` and so on. Note that after creation of `timeline1` and `timeline2` it is still possible to restore data at the moment of the `rp2` point — without applying the timeline concept, this is impossible.



Timeline logic

Quick start

The typical sequence of actions when working with ADBM via the web interface includes the following steps:

1. Connect to ADBM.
2. Create a configuration, which will be used to create backups and perform other related actions.
3. Run a backup manually or wait for the automatic backup launch on a schedule. If necessary — view a list of backups.
4. Create a restore point manually or wait for the automatic restore point creation on a schedule.
5. Run the auxiliary operations if necessary:
   - Cleanup backups
   - Verify backups
   - Terminate the running backup/cleanup action
   - Resume the latest terminated backup
6. Restore data from a backup if it is necessary to return the database state at the moment of one of existing restore points.

On each of the steps, you can view the details of running actions, including causes of the failures (if they are fixed).

Configure an external database for ADBM

Daria Barysheva

To Table of Contents

For production environments, it is recommended to use an external PostgreSQL database as ADBM data storage for high performance and fault tolerance. Starting with ADBM 1.7.3, to configure an external database, you need to follow the steps that are described below:

1. Perform the following queries in PostgreSQL. The queries should be run under the default user `postgres` with the `superuser` rights:

```
CREATE ROLE <user_name> LOGIN PASSWORD '<user_password>'; ①
ALTER ROLE <user_name> SET search_path TO public, adbm;
CREATE DATABASE <database_name> WITH owner <user_name>; ②
```

① Create the `<user_name>` user with appropriate rights for connecting to the external database.

② Create the `<database_name>` database for ADBM data migration.

2. Connect to the created database `<database_name>` under a superuser and run:

```
CREATE SCHEMA adbm; ①
GRANT USAGE, CREATE ON SCHEMA adbm TO <user_name>; ②
ALTER DEFAULT PRIVILEGES IN SCHEMA adbm GRANT UPDATE, INSERT, SELECT, DELETE ON TABLES TO <user_name>;
CREATE EXTENSION btree_gin WITH schema adbm;
CREATE EXTENSION pg_trgm WITH schema adbm;
GRANT SELECT ON pg_extension TO <user_name>;
```

① Create the `adbm` schema in the `<database_name>` database for ADBM data migration.

② Create extensions in the `adbm` schema and grant of their usage to the `<user_name>` user.

> If you can assign superuser rights to a PostgreSQL user (which will be used for connecting to an external database from ADBM), it is enough to create a role and a database in PostgreSQL before installation or reconfiguration of ADBM:
>
> ```
> CREATE ROLE <user_name> LOGIN PASSWORD '<user_password>' SUPERUSER;
> ALTER ROLE <user_name> SET search_path TO public, adbm;
> CREATE DATABASE <database_name> WITH owner <user_name>;
> ```

3. Provide the ADBM access to the external PostgreSQL database by adding the following record to the *pg_hba.conf* file on the PostgreSQL server.

Without SSL

```
host    <database_name>  <user_name>       <adbm_address>      trust
```

where:

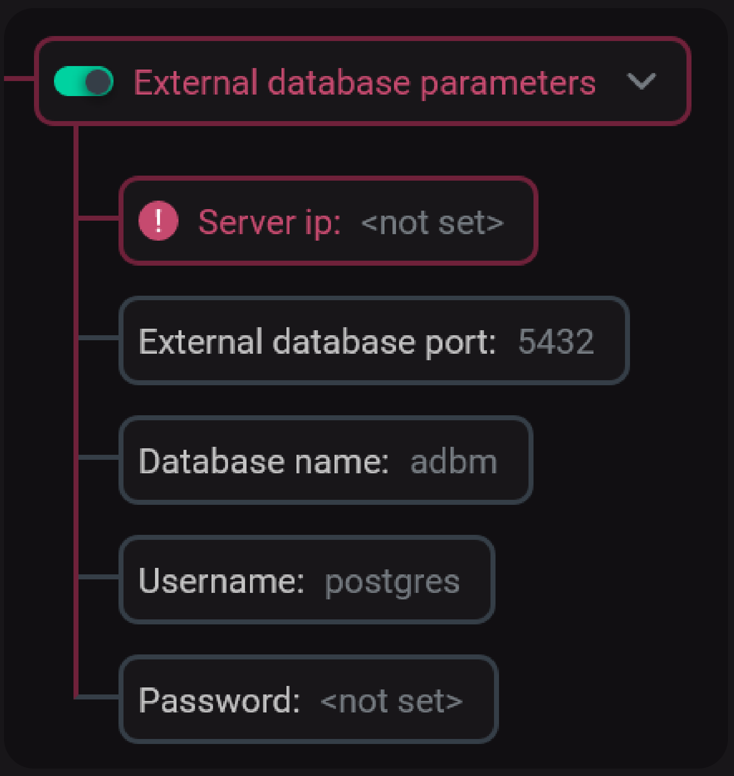
- `<database_name>` — a name of the created database.
- `<user_name>` — a name of the created user.
- `<adbm_address>` — an IP address of ADBM with a subnet number.

With SSL

```
hostssl     all  all        0.0.0.0/0       md5
```

For more information on how to configure PostgreSQL for using SSL, see SSL encryption.

4. On the ADBM service configuration page, activate the ***External database parameters*** toggle and fill in all parameters in the list that opens. The assignment of parameters is described in the Configuration parameters → ADBM → External database parameters section.



External database parameters

Server ip: <not set>

External database port: 5432

Database name: adbm

Username: postgres

Password: <not set>

Parameters for connecting to the external PostgreSQL DB

5. Click ***Save*** to save the ADBM configuration.

6. Install the ADBM service using the Install service action or apply the ***Reconfigure & Restart*** action to the ADBM service if it is already installed.

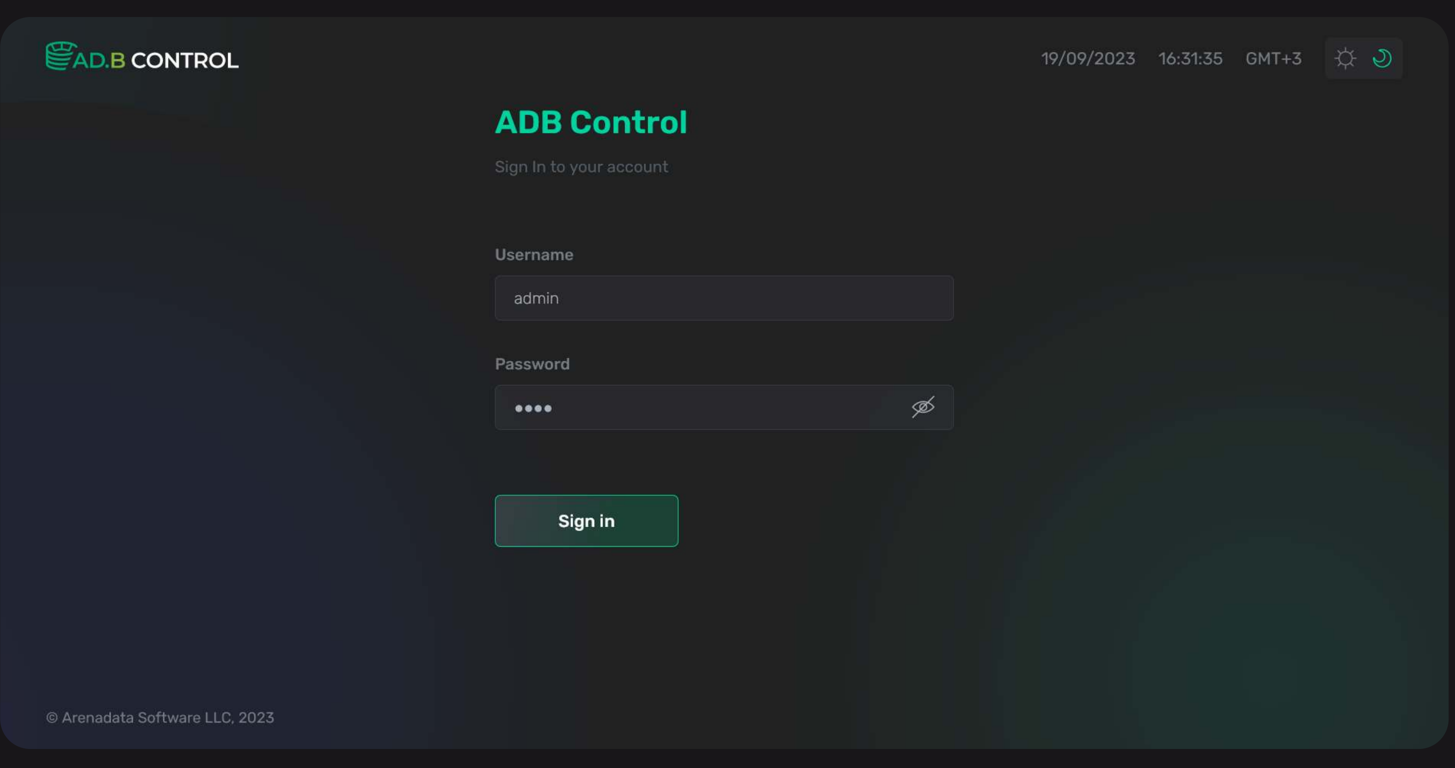
Connect to ADBM

Daria Barysheva

To Table of Contents

Currently, you can work with ADBM via the Arenadata DB Control (ADB Control) web interface. To connect to ADBM, follow the steps:

1. In the browser bar, enter the IP address of the host where ADB Control is deployed. To specify the port number, use the value of the Listening port parameter from the ***UI component parameters*** section defined when configuring the ADB Control service during the ADB cluster installation (`81` by default).
2. In the window that opens, enter the default user credentials and click ***Sign in***:
   - ***Username:*** admin
   - ***Password:*** 1234

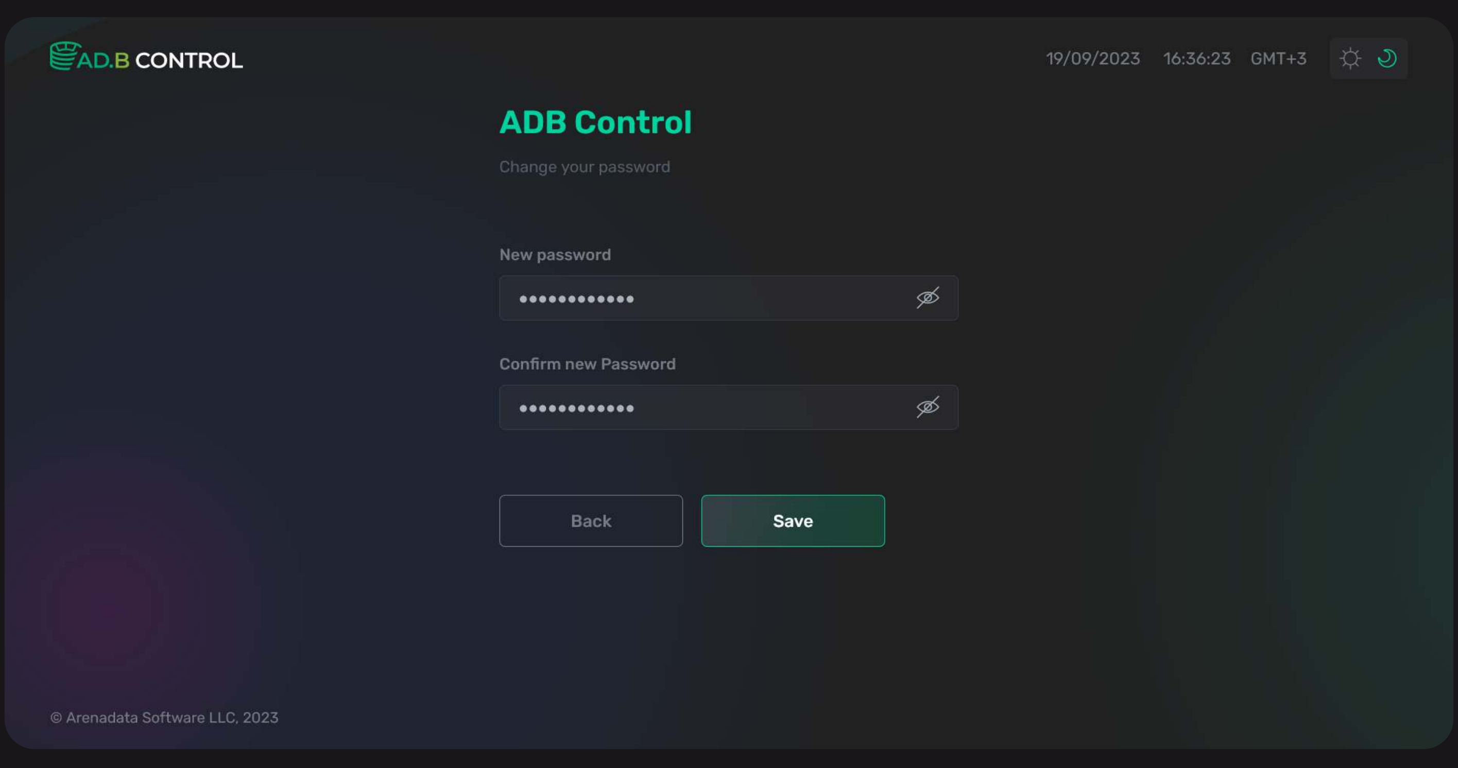

AD.B CONTROL
19/09/2023 16:31:35 GMT+3
ADB Control
Sign In to your account
Sign in
© Arenadata Software LLC, 2023

Username: admin

Password: ••••

Log in to ADB Control

3. When you first log in, you should change a password for the `admin` user. Enter a new password in the ***New password*** field and confirm it in the ***Confirm new Password*** field. Click ***Save***. Note that your password must be greater than or equal to 12 characters by default. Any characters other than spaces are allowed to enter.

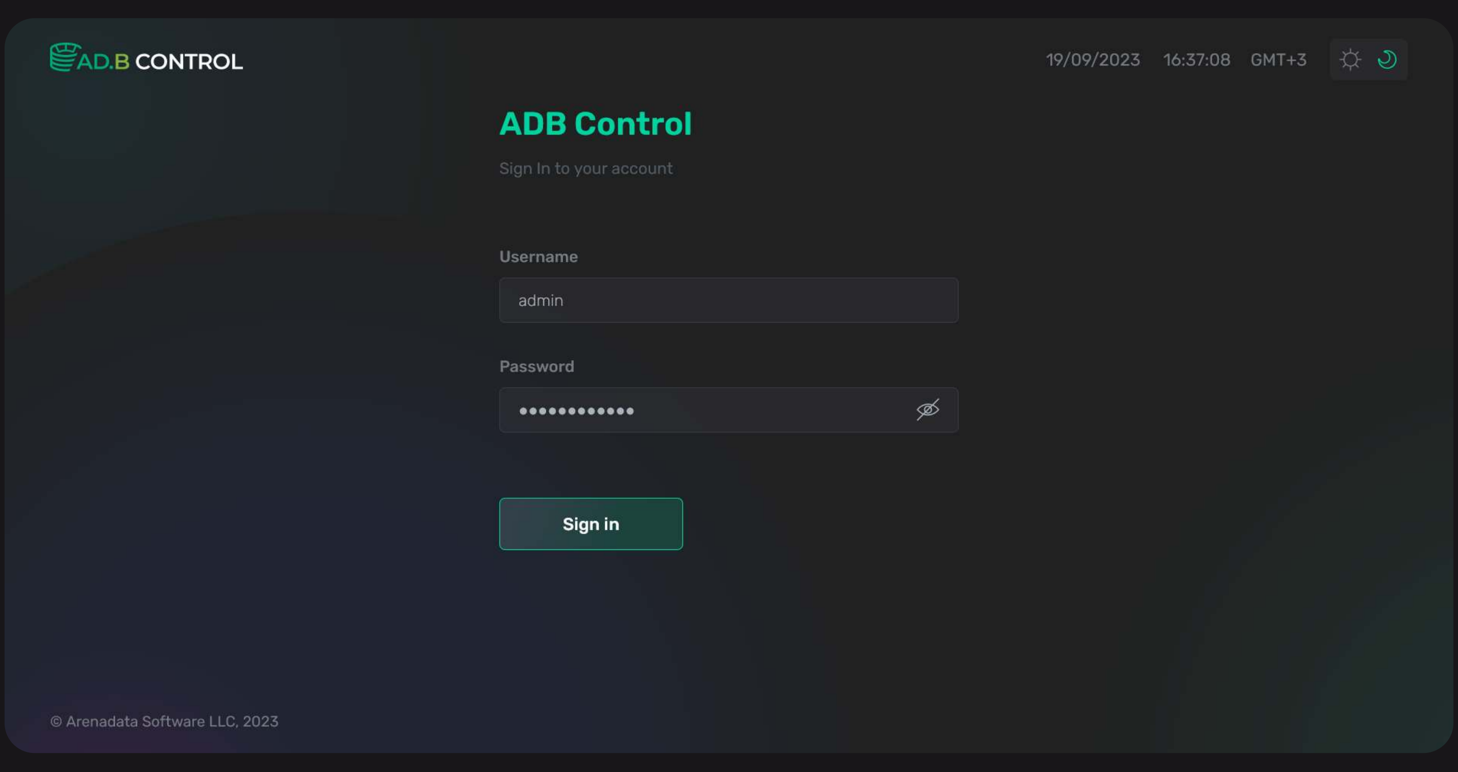

AD.B CONTROL
19/09/2023 16:36:23 GMT+3
ADB Control
Change your password
Back
Save
© Arenadata Software LLC, 2023

New password: ••••••••••••

Confirm new Password: ••••••••••••

Change a password

4. After saving a password, re-enter your login and a new password. Note that when a certain number of unsuccessful login attempts is reached, the user will be blocked — permanently or temporarily, depending on the settings.

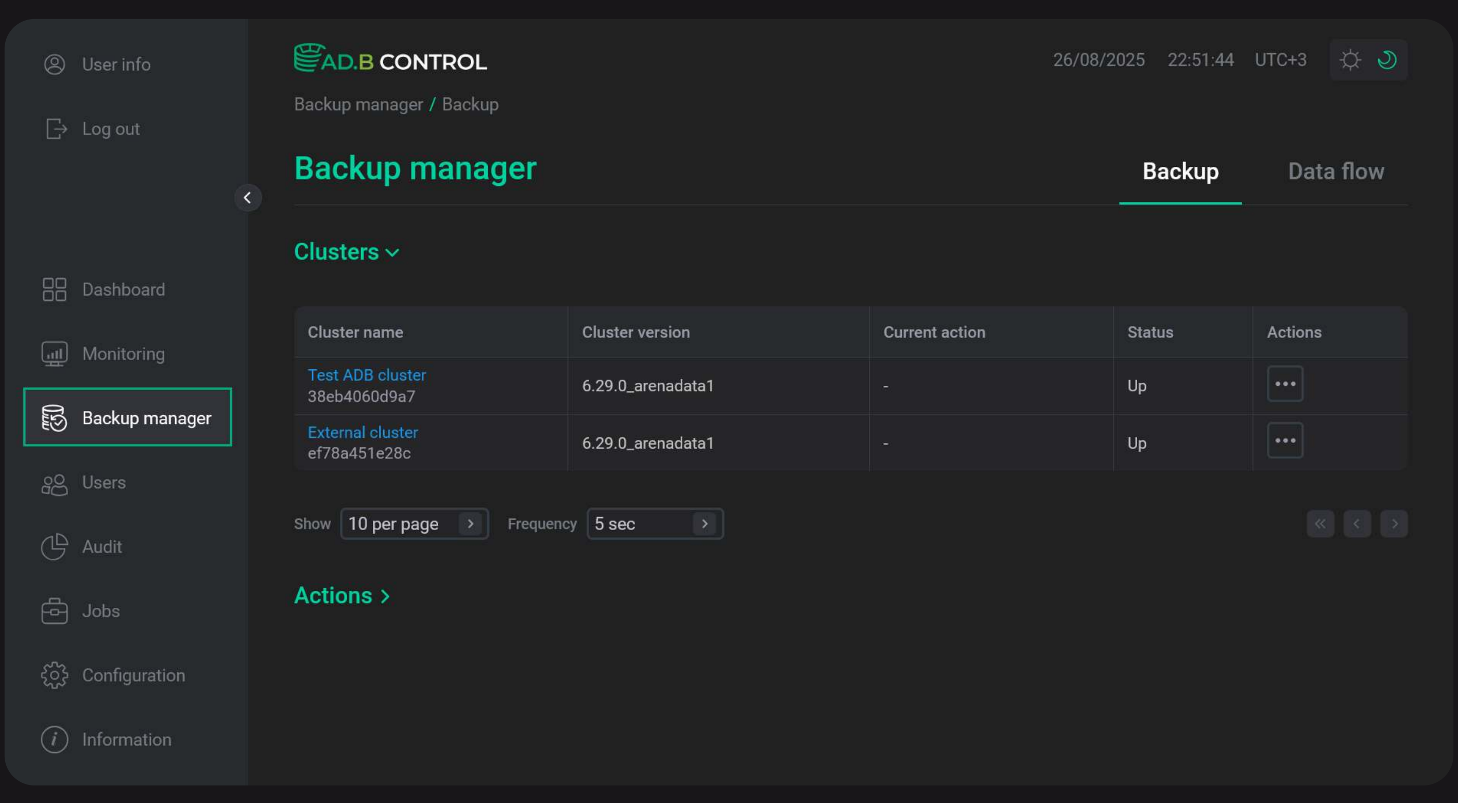

AD.B CONTROL
19/09/2023 16:37:08 GMT+3
ADB Control
Sign In to your account
Sign in
© Arenadata Software LLC, 2023

Username: admin

Password: ••••••••••••

Log in to ADB Control with a new password

5. On the next page, select the ***Backup manager*** item in the left navigation menu.

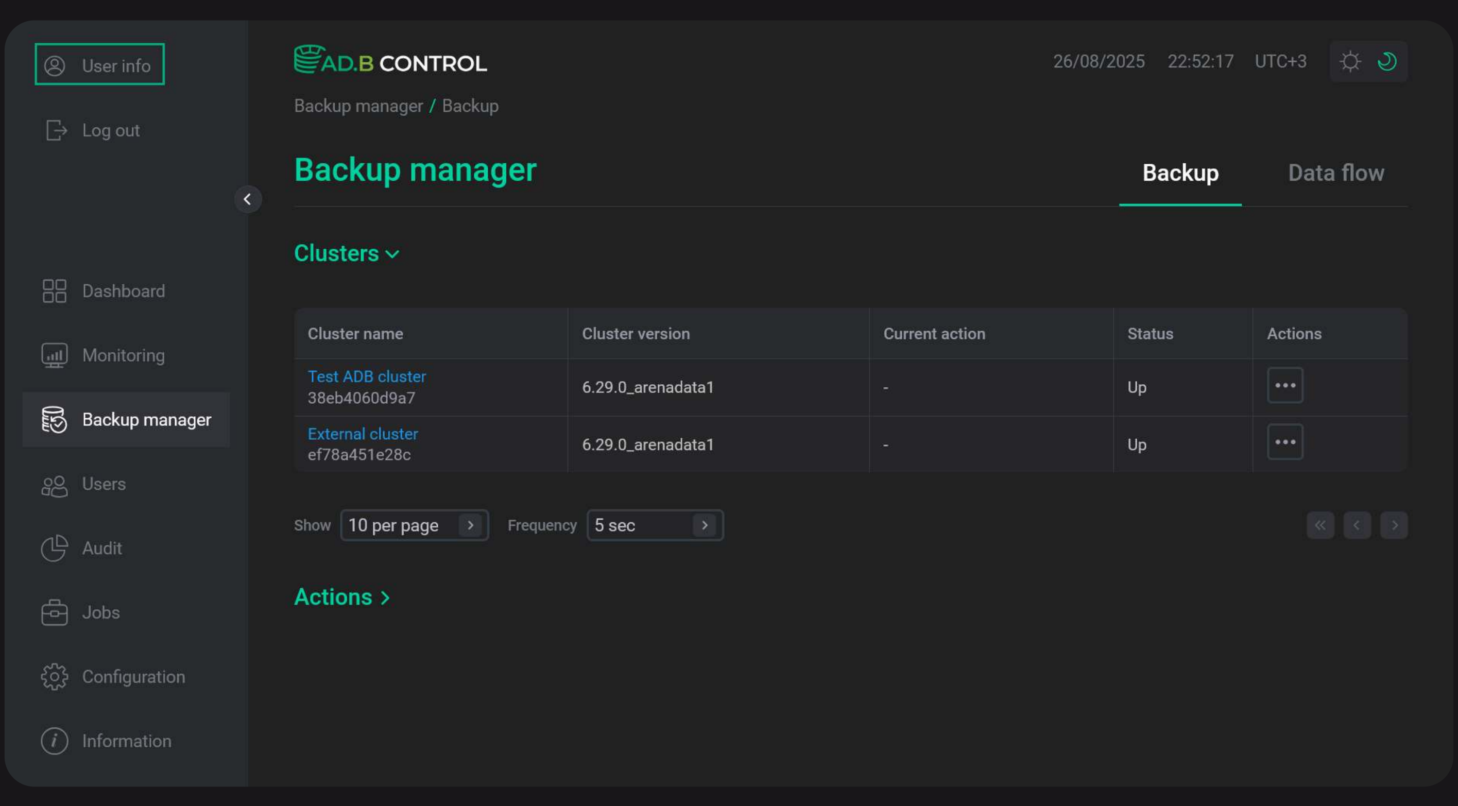

User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/08/2025 22:51:44 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show: 10 per page Frequency: 5 sec

The "Backup manager" page

6. If you later need to get information about the current user, select the ***User info*** tab in the left navigation menu.

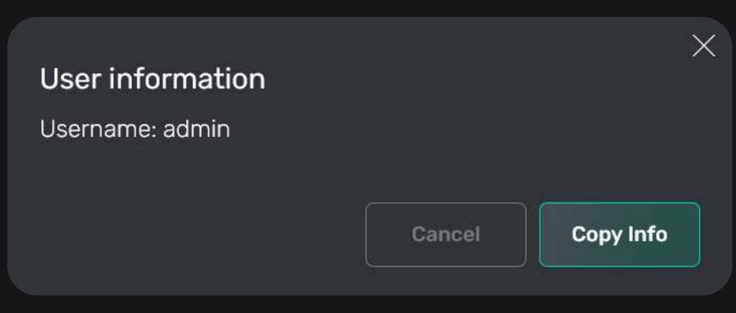

User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/08/2025 22:52:17 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show: 10 per page Frequency: 5 sec

View user information

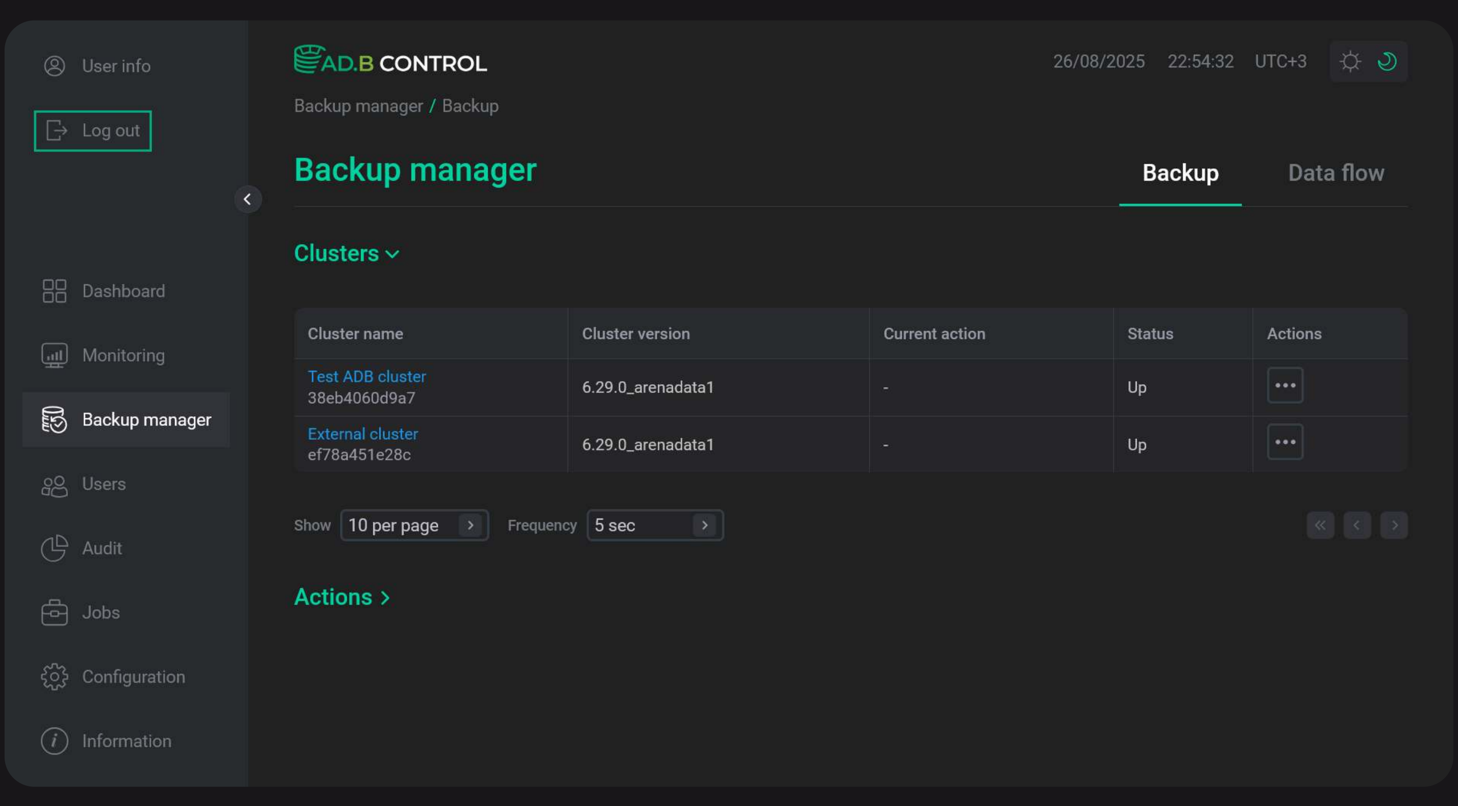
The modal window that opens displays the current user name, which can be copied by clicking ***Copy info***.


User information
Cancel
Copy Info

Username: admin

Copy user information

7. To log out from ADB Control, click ***Log out*** in the left navigation menu.


User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/08/2025 22:54:32 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show: 10 per page Frequency: 5 sec

Log out from ADB Control

- All actions described in the subsequent sections of the ADBM documentation are performed on the ***Backup manager*** page via the ADB Control UI.
- The ***Backup manager*** page is available to users with the `Administrator`, `ADBM Administrator`, or `Owner` role.
- To get information on how to add new users and configure access rights for them, refer to the following articles:
  - Basic authentication
  - LDAP authentication
  - Authorization
- You can change security policies (password length limits, authorization attempt limit, blocking rules) on the ***Configuration*** → ***Security*** page. For more information, see Configure security policies.

Manage configurations in ADBM

Daria Barysheva, Anton Vasilev

Contents

To Table of Contents

- Backup configuration
  - Overview
  - Add a new configuration
  - Edit a configuration
  - Draft configurations
- Schedule
- Storages
  - Add a new storage
  - Monitor storages

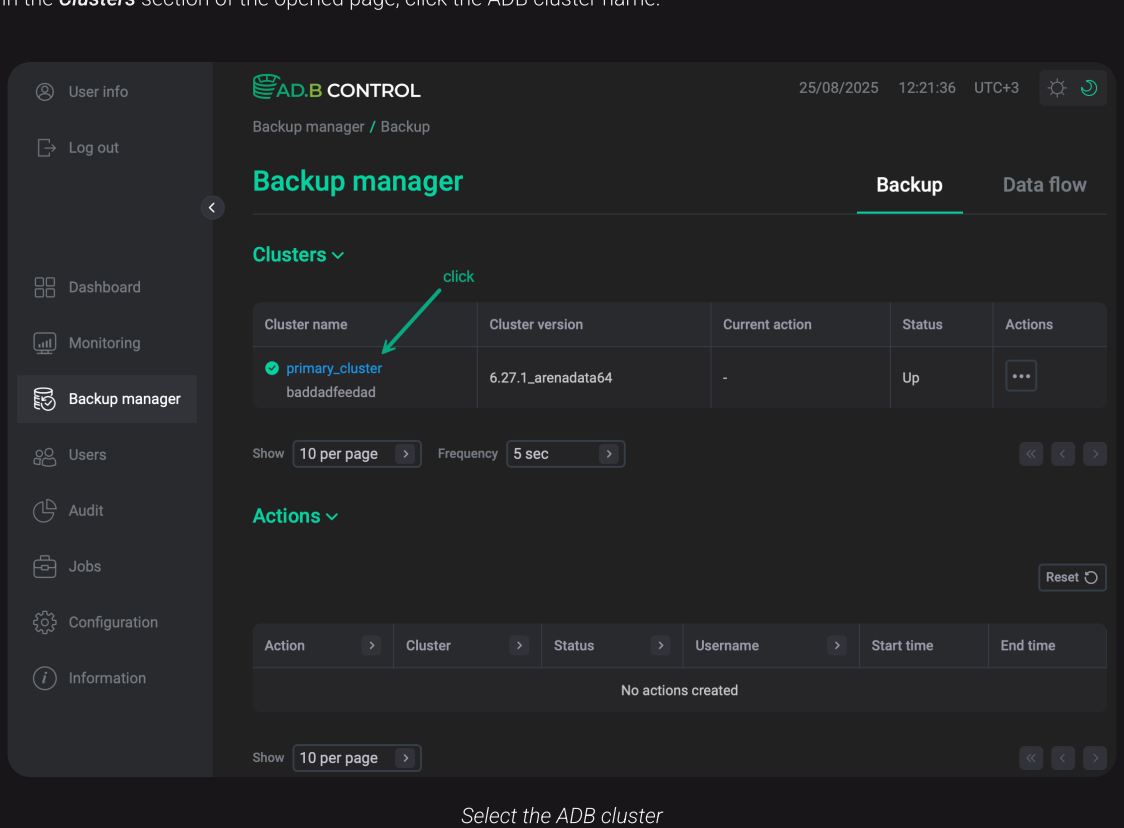
Backup configuration

Overview

In ADBM, a **backup configuration** means a set of options that are used to manage binary database backups. These options include paths for saving backups, log levels, timeouts, compression settings, etc. ADBM supports several configuration versions. But there can be only one active version, which is marked as `current` in the user interface.

Possible operations with configurations are described below. All operations are performed on the ***Configuration → Backup configuration*** tab. To open this tab, follow the steps:

1. Open the Backup manager page via the ADB Control web interface.
2. In the table that is located in the ***Clusters*** section of the opened page, click the ADB cluster name.

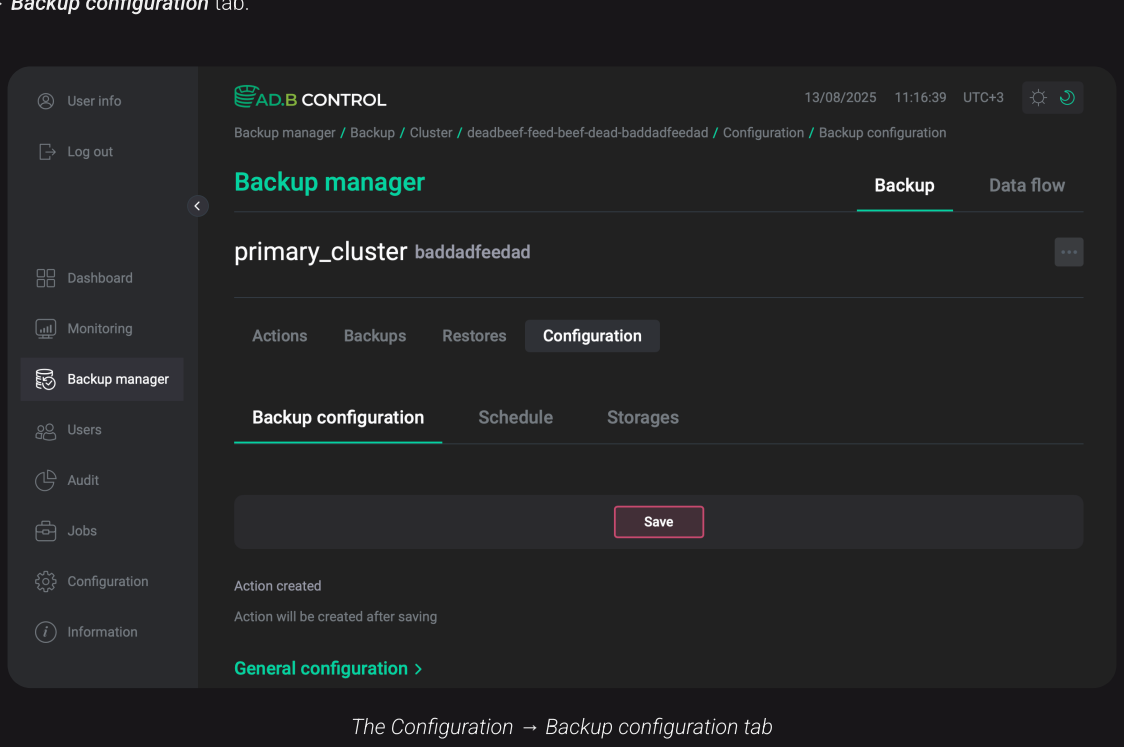


AD.B CONTROL
25/08/2025 12:21:36 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
click
Cluster name
Cluster version
Current action
Status
Actions
primary_cluster
baddadfeedad
6.27.1_arenadata64
-
Up
Actions
Reset
Action
Cluster
Status
Username
Start time
End time
No actions created
User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information

Select the ADB cluster

> **NOTE**
>
> Starting with ADBM 2.5.0, a unique cluster identifier is displayed next to each cluster name.

3. Select the ***Configuration → Backup configuration*** tab.



AD.B CONTROL
13/08/2025 11:16:39 UTC+3
Backup manager / Backup / Cluster / deadbeef-feed-beef-dead-baddadfeedad / Configuration / Backup configuration
Backup manager
Backup
Data flow
primary_cluster baddadfeedad
Actions
Backups
Restores
Configuration
Backup configuration
Schedule
Storages
Save
Action created
Action will be created after saving
General configuration

The Configuration → Backup configuration tab

Initially, there are no configurations. To start working with backups in ADBM, add a new configuration.

> **Limitations**
>
> - ADBM does not support topology of ADB clusters that consist of a single segment host with mirroring enabled. For these clusters, an error occurs when saving the configuration. Without this restriction, there would be duplicates in the `pgbackrest` configuration for each stanza.
> - If the cluster topology has changed before the configuration is edited (via the ***Expand*** action in ADCM), the configuration will not apply as long as there is at least one valid backup in ADBM. This is due to the fact that the configuration applied on the cluster with the changed topology automatically invalidates all previous backups. Therefore, in such cases, it is necessary to explicitly remove all existing backups and then edit the configuration.

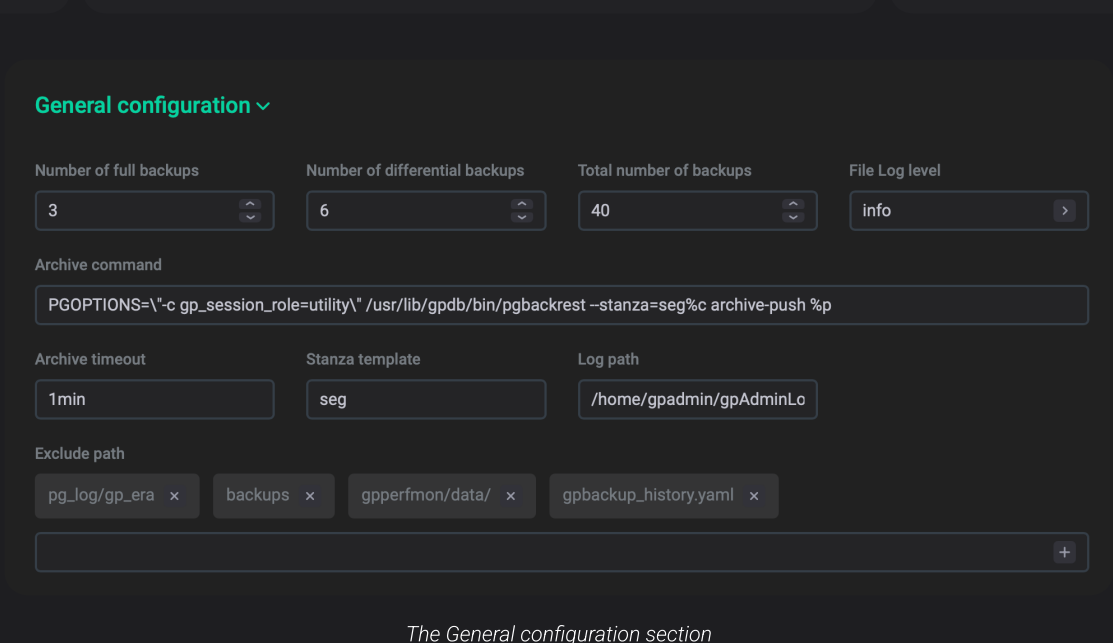
Add a new configuration

> **IMPORTANT**
>
> - Remember that enabling any configuration (even without configured schedules) applies `archive_command` to your cluster and starts the WAL generating process.
> - The repository that is used in configuration settings may be the same for multiple backup configurations. Every configuration can have a list of backups taken with that particular configuration. Thus, there may be sets of backups associated with different configurations but actually stored in the same repository. ADBM marks backups for deletion by cleanup policies considering combination of the repository type, path, and endpoint (in case of S3). So, in case you use different endpoints for the same S3 repository, the ***Cleanup*** action will treat them as different repositories and delete backups that should be kept.

1. Open the ***Configuration → Backup configuration*** tab as described above.
2. Fill in the parameters that are located in the ***General configuration***, ***Timeout***, ***Storage***, ***Workspace***, and ***Compression*** sections. When filling the fields, pay attention to the following:
   - Mandatory parameters are highlighted in red.
   - For those parameters that specify time values, you can use the following units:
     - `ms` — milliseconds;
     - `sec` — seconds;
     - `min` — minutes;
     - `hr` — hours;
     - `d` — days;
     - `w` — weeks.
   - For those parameters that specify data amount, you can use the following units:
     - `b` — bytes;
     - `kb` — kilobytes;
     - `mb` — megabytes;
     - `gb` — gigabytes.
   - For those parameters that specify time or data amount, use the `<value><unit>` format (without spaces). For example, `1min`, `2hr`. If you need a complex value, use a space (e.g. `30min 30sec`).
   - By clicking the ? icon (which appears to the right of some fields during their filling), you can view hints: available measurement units, minimum and maximum values, and so on.

General configuration

Parameter	Description	Default value
Number of full backups	A maximum number of full backups. After this limit is reached, the oldest extra full backups will be removed during the next automatic or manual ***Cleanup*** action. For more details, see Cleanup	3
Number of differential backups	A maximum number of differential backups. After this limit is reached, the oldest extra differential backups will be removed during the next automatic or manual ***Cleanup*** action. For more details, see Cleanup	6
Total number of backups	A total number of backups allowed for the current cluster, including backups in the `Stopped` and `Failed` statuses that are not deleted by the ***Cleanup*** action policies. The total amount of backups will be checked before making a new backup (whether it hits the specified limit)	40
File Log level	A level for file logging. Possible values: - `off` — no logs; - `error`; - `warn`; - `info`; - `detail`; - `debug`; - `trace`.	info
Archive command	The command that is to be executed when write-ahead log (WAL) files are transferred to the archive directory. If you change the ***Archive command*** field value, then the ADB cluster is automatically restarted when the configuration is applied (to change the value of the `archive_command` GUC). The stanza configuration name prefix for segments that is used in the ***Archive command*** should be equal to the ***Stanza template*** value (see below): `--stanza=<Stanza template>%c`	`PGOPTIONS=\"-c gp_session_role=utility\" /usr/lib/gpdb/bin/pgbackrest --stanza=seg%c archive-push %p`
Archive timeout	Sets maximum time to wait for each WAL segment (required for backup consistency) to reach the archive repository	1min
Stanza template	A stanza configuration name prefix for segments. Should be equal to the prefix that is used in the ***Archive command*** value (see above): `--stanza=<Stanza template>%c`	seg
Log path	A path where log files are to be stored	/home/gpadmin/gpAdminLogs
Exclude path	Defines the paths and files that should be excluded from backups	- pg_log/gp_era - backups - gpperfmon/data/ - gpbackup_history.yaml



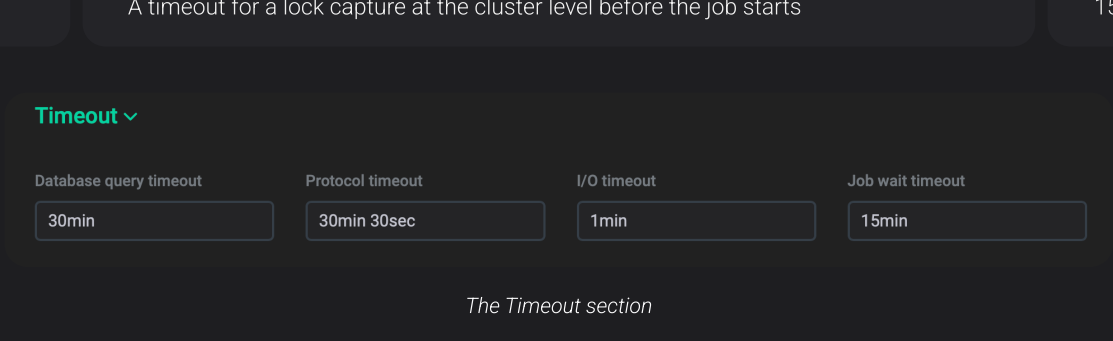
General configuration

Number of full backups: 3
Number of differential backups: 6
Total number of backups: 40
File Log level: info
Archive command: PGOPTIONS=\"-c gp_session_role=utility\" /usr/lib/gpdb/bin/pgbackrest --stanza=seg%c archive-push %p
Archive timeout: 1min
Stanza template: seg
Log path: /home/gpadmin/gpAdminLo
Exclude path: pg_log/gp_era, backups, gpperfmon/data/, gpbackup_history.yaml

The General configuration section

Timeout

Parameter	Description	Default value
Database query timeout	Sets the timeout for queries against the database including the backup start/stop functions which can take a significant amount of time. The timeout should be kept high unless you know that these functions will return quickly. The ***Database query timeout*** value should be less than the ***Protocol timeout*** value	30min
Protocol timeout	Sets the timeout that the local or the remote process can wait for a new message to be received on the protocol layer. The timeout prevents processes from waiting indefinitely. The ***Protocol timeout*** value should be greater than the ***Database query timeout*** value	30min 30sec
I/O timeout	A timeout that is used for connections and read/write operations. The entire read/write operation does not need to complete within this timeout but some progress should be made, even if it is only a single byte	1min
Job wait timeout	A timeout for a lock capture at the cluster level before the job starts	15min



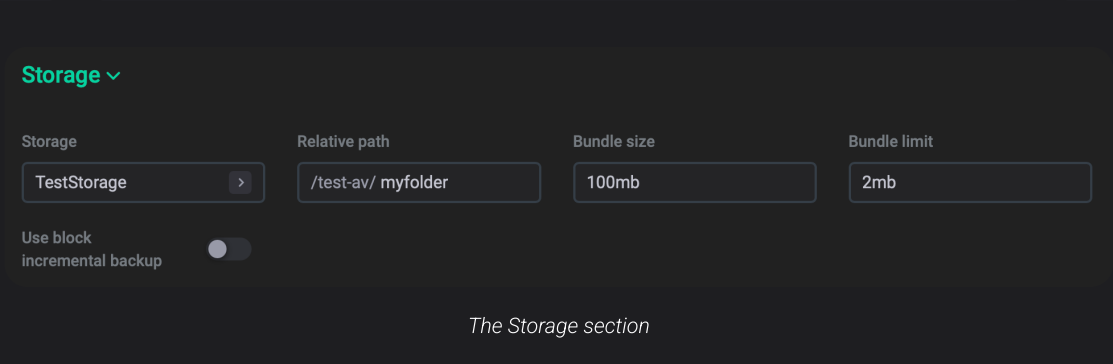
Timeout

Database query timeout: 30min
Protocol timeout: 30min 30sec
I/O timeout: 1min
Job wait timeout: 15min

The Timeout section

Storage

Parameter	Description	Default value
Storage	One of the storages that you added on the Storages tab	—
Relative path	A path to a repository for backups relative to the ***Repository path*** of the selected storage. You can use the same storage for multiple configurations and assign each of them its own separate repository within that storage	—
Bundle size	A target size for file bundles. A **bundle** means grouping single files into one file before the backup launch. It is not recommended to set the option too high because backup retries will need to redo the entire bundle	100mb
Bundle limit	A limit for the files that are included in bundles (see ***Bundle size*** above). The files larger than this size are stored separately. Bundled files cannot be reused when a backup is resumed, so this option, in fact, controls the files that can be resumed	2mb
Use block incremental backup	A switch that indicates whether to make more granular backups by splitting files into blocks that can be backed up independently	false



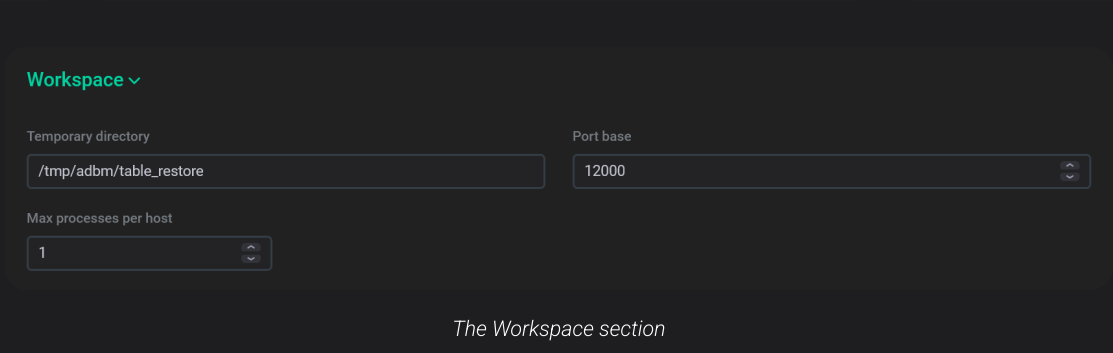
Storage

Storage: TestStorage
Relative path: /test-av/ myfolder
Bundle size: 100mb
Bundle limit: 2mb
Use block incremental backup

The Storage section

Workspace

Parameter	Description	Default value
Temporary directory	The working directory that is used during the Partial restore action for temporary data allocation. To reduce the impact on cluster performance during the table restore process, it is recommended to configure this directory on a disk different from the cluster disks. Since all operations during the table restore do not require random disk access, the HDD disk is sufficient	/tmp/adbm/table_restore
Port base	The base port number that is used during the Partial restore action. The actual port number of each segment is equal to `<Port base> + <dbId>` where `<dbId>` is the `gp_segment_configuration.dbid` value corresponding to the segment. Allowed values: `[1025,65035]`	12000
Max processes per host	The maximum number of processes that can work in parallel on one segment host during the Partial restore action. Allowed values: `[1,100]`	1



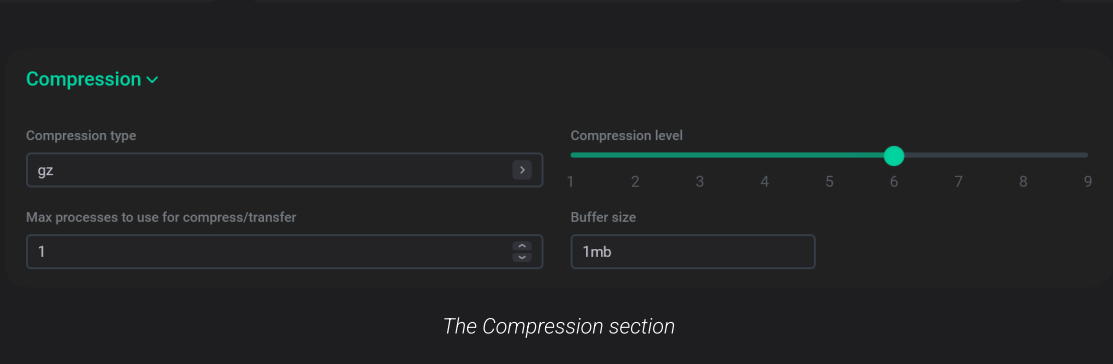
Workspace

Temporary directory: /tmp/adbm/table_restore
Port base: 12000
Max processes per host: 1

The Workspace section

Compression

Parameter	Description	Default value
Compression type	A compression type. Possible values: - `none` — no compression; - `bz2`; - `gz`; - `lz4`; - `zst`.	gz
Compression level	A compression level. Possible values: `1` — `9`	6
Max processes to use for compress/transfer	The maximum number of processes to be used on every segment for compression/transferring	1
Buffer size	A buffer size for I/O operations	1mb

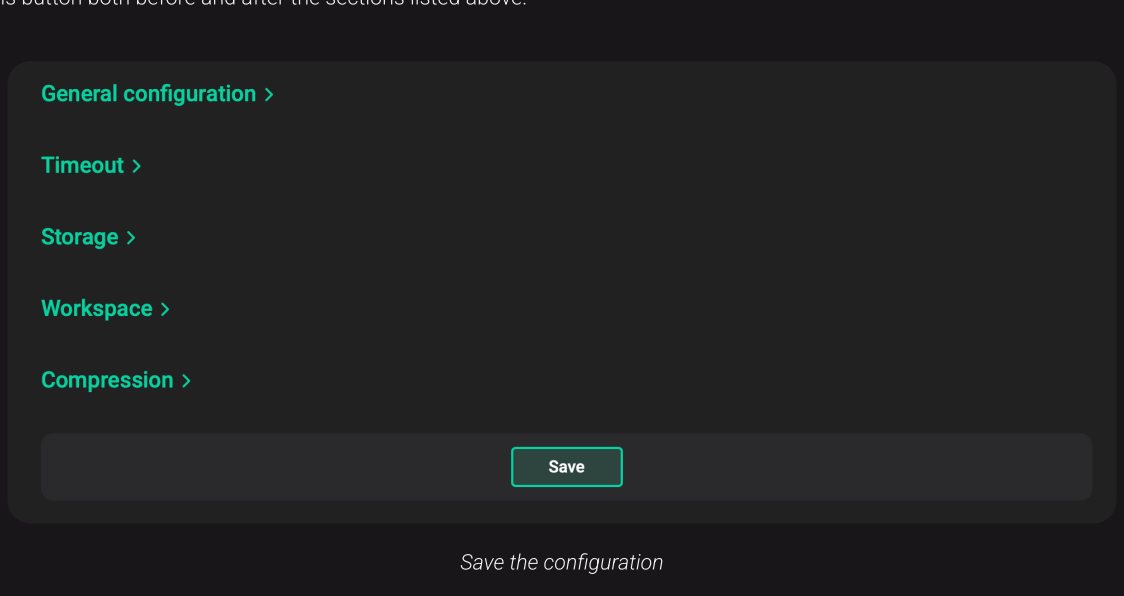


Compression

Compression type: gz
Compression level: 6
Max processes to use for compress/transfer: 1
Buffer size: 1mb

The Compression section

3. Click ***Save***. You can find this button both before and after the sections listed above.



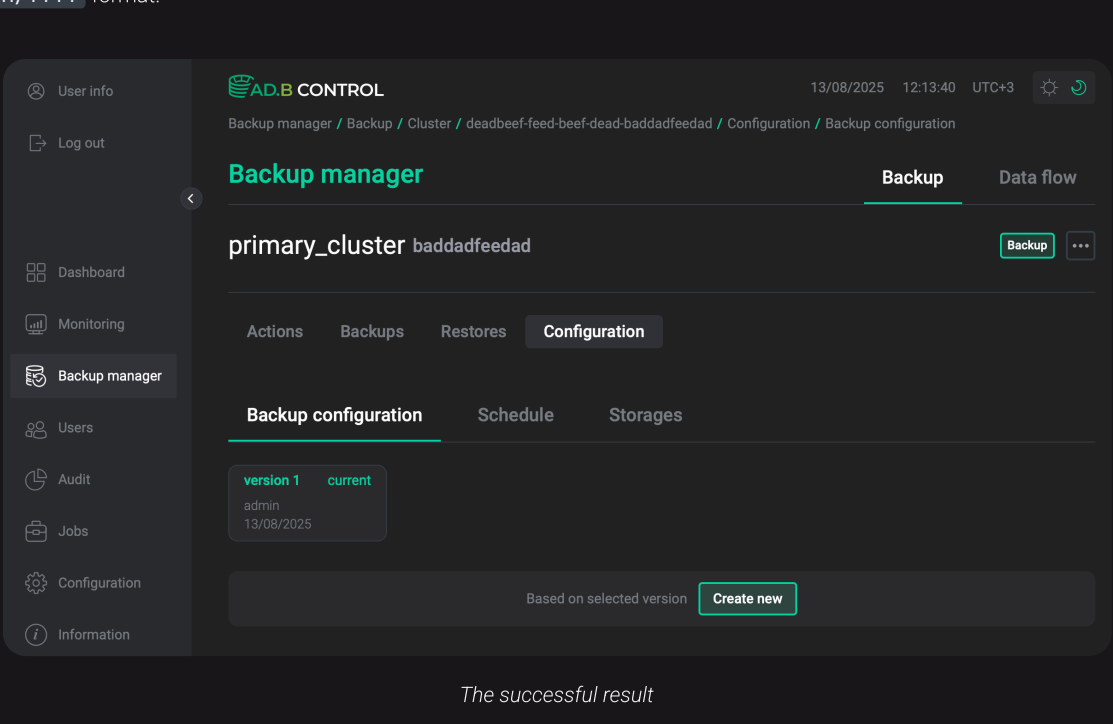
General configuration
Timeout
Storage
Workspace
Compression
Save

Save the configuration

4. As a result, the ***Apply backup configuration*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
   Starting with ADBM 2.6.5, you can open the ***Actions*** tab and view the action details by clicking the ***Apply backup configuration*** link, which is available in the ***Action created*** field after you save the configuration.

Until all necessary actions are completed successfully, the configuration will have the draft status. . As a successful result of all actions, the configuration acquires the `current` status. Along with status, the following information about the configuration is available:

- The version number (starting with `1`).
- The user who created a configuration version.
- The creation date in the `DD/MM/YYYY` format.



AD.B CONTROL
13/08/2025 12:13:40 UTC+3
Backup manager
primary_cluster baddadfeedad
Backup
Actions
Backups
Restores
Configuration
Backup configuration
Schedule
Storages
version 1
current
admin
13/08/2025
Based on selected version
Create new

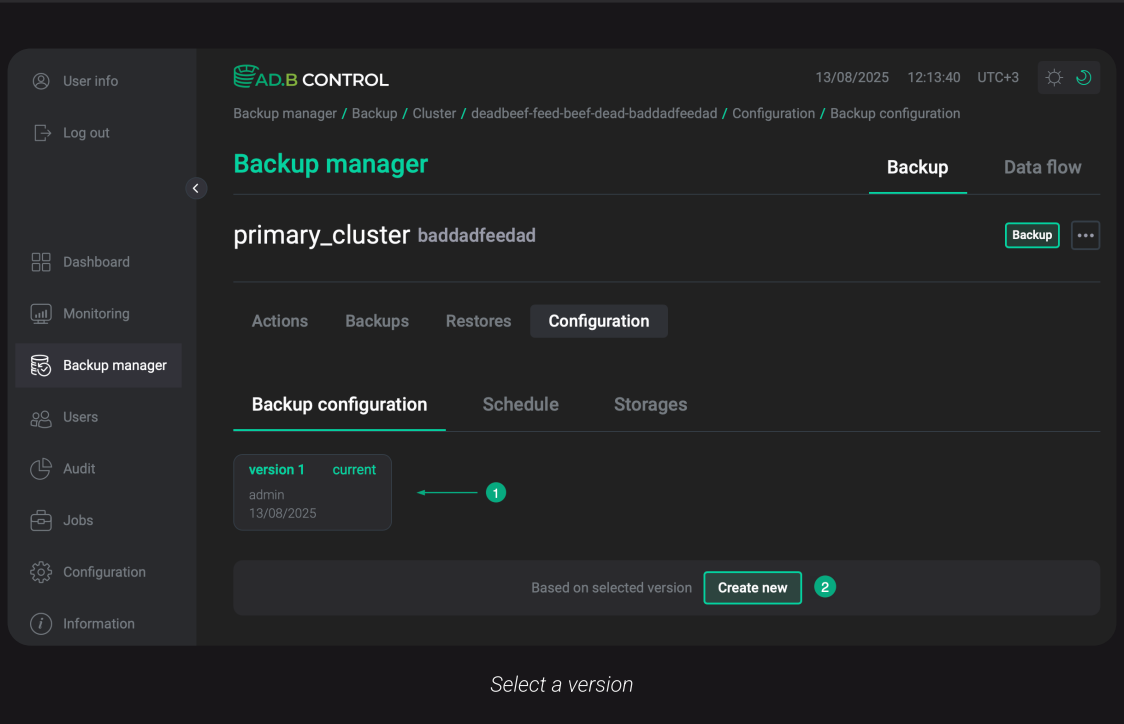
The successful result

Edit a configuration

After you created the configuration, you cannot edit it. But it is possible to add a new configuration version based on one of the previously created. To do this, perform the following steps:

1. Open the ***Configuration → Backup configuration*** tab as described above.
2. In the list of available configuration versions, click the version on base of which you want to create a new one. All parameters of the selected version are displayed under the list with available versions (in the read-only mode).

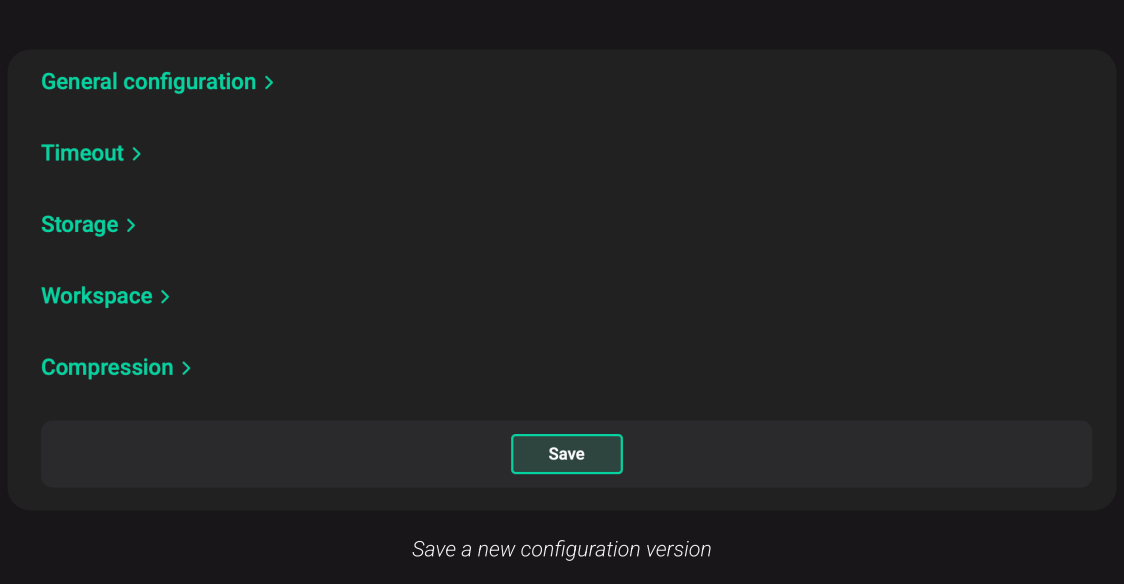
> **NOTE**
>
> You can create a configuration based on any previous version, not just the latest.



AD.B CONTROL
13/08/2025 12:13:40 UTC+3
Backup manager
primary_cluster baddadfeedad
Backup configuration
Schedule
Storages
version 1
current
admin
13/08/2025
1
Based on selected version
Create new
2

Select a version

3. Click ***Create new***.
4. Apply necessary changes to the parameters that are located in the ***General configuration***, ***Timeout***, ***Storage***, ***Workspace***, and ***Compression*** sections. The parameters are the same as described above.
5. Click ***Save*** to add a new configuration version. You can find this button both before and after the sections with parameters.

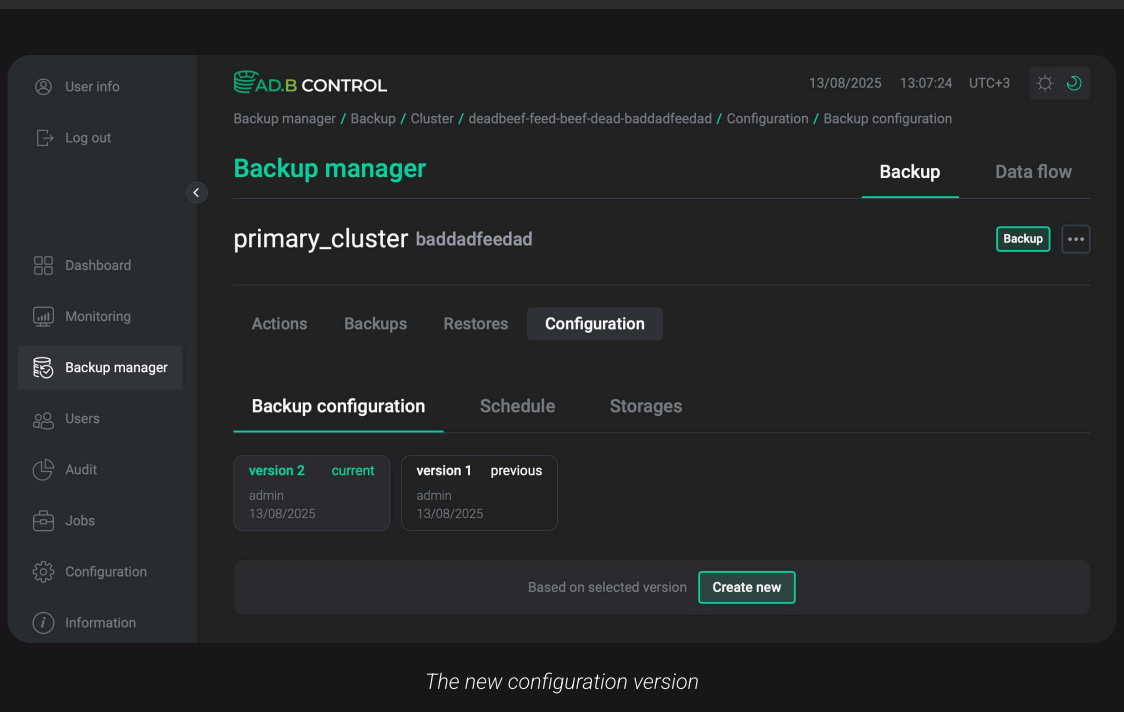


General configuration
Timeout
Storage
Workspace
Compression
Save

Save a new configuration version

6. As a result, the ***Apply backup configuration*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
7. After all necessary actions are completed successfully, the new configuration version becomes available on the ***Configuration*** tab and has the `current` status. The previous configuration acquires the `previous` status.

> **NOTE**
>
> - ADBM uses the latest configuration version you added.
> - Inactive configuration versions that have no related backups will be removed during the next automatic or manual Cleanup action.



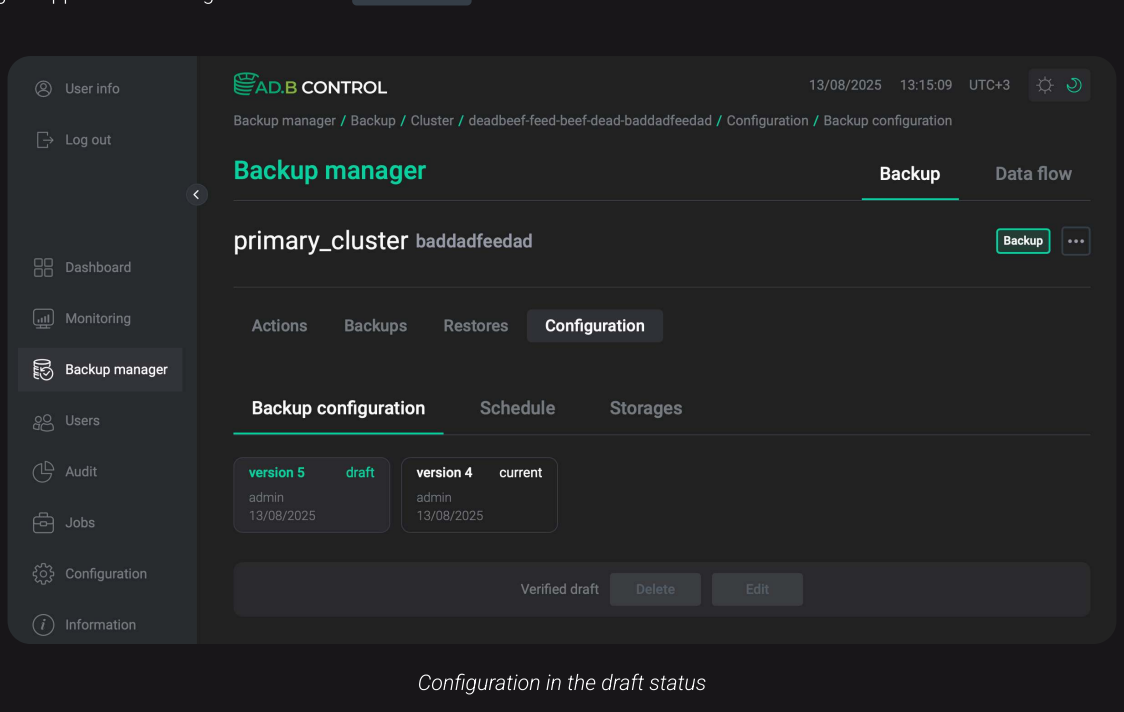
AD.B CONTROL
13/08/2025 13:07:24 UTC+3
Backup manager
primary_cluster baddadfeedad
Backup configuration
Schedule
Storages
version 2
current
admin
13/08/2025
version 1
previous
admin
13/08/2025
Based on selected version
Create new

The new configuration version

Draft configurations

Starting with the version 1.6.3, ADBM supports an initial status `draft` for backup configurations. This status is assigned to all configurations when attempting to save them until all actions are successfully applied and the status is changed to `current`. However, if a new configuration cannot be validated at the moment of its creation, it is saved as a draft. Thus, you can avoid re-entering all information during the next applying attempt. Configuration drafts can be modified (and re-applied) or removed by clicking ***Edit*** and ***Delete***, respectively.

Starting with ADBM 1.7.3, the `verified` status is also available for backup configurations. This status means that validation of the backup configuration is successfully completed, but some errors occurred during its application. Configurations in the `verified` status can be edited or removed like drafts.

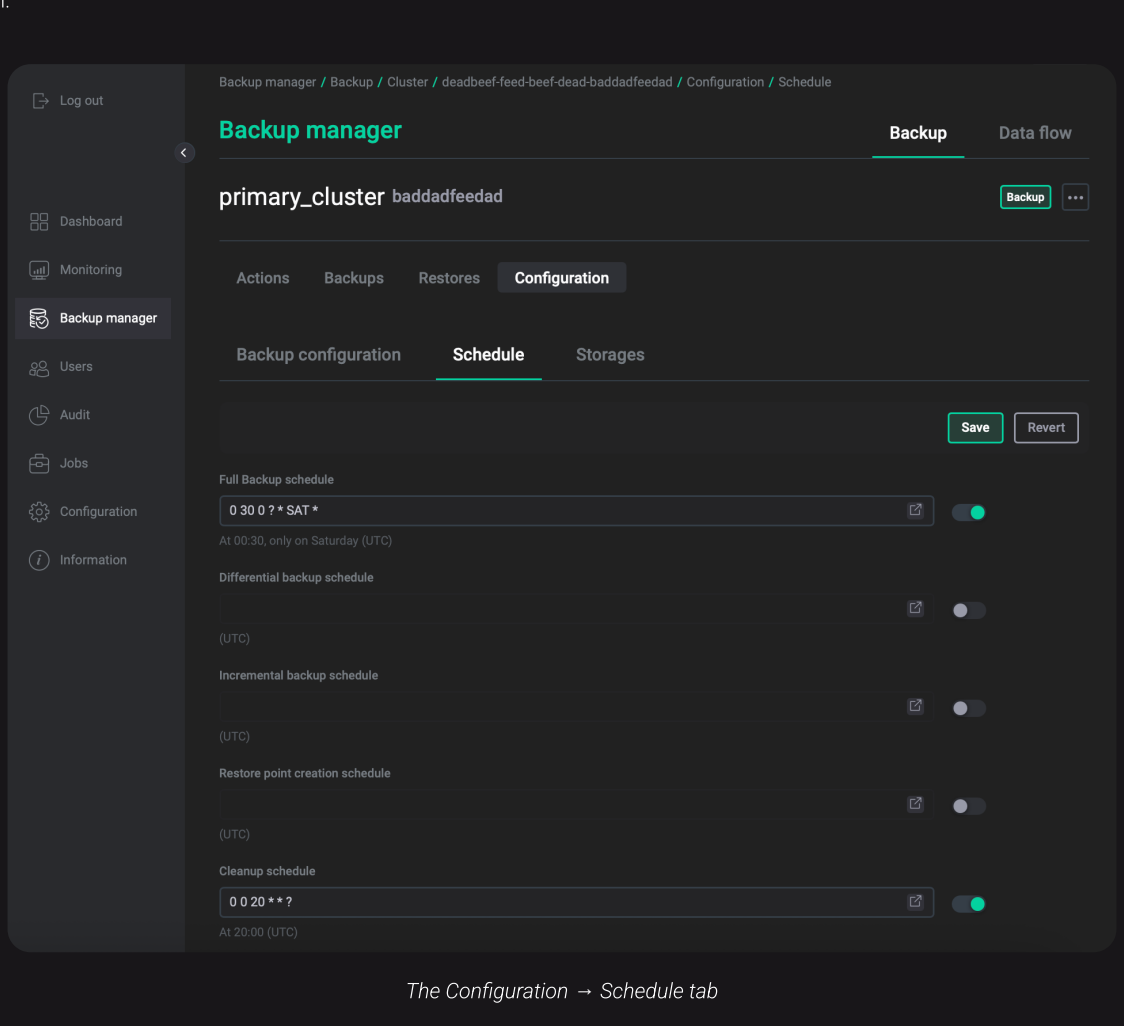


AD.B CONTROL
13/08/2025 13:15:09 UTC+3
Backup manager
primary_cluster baddadfeedad
Backup configuration
Schedule
Storages
version 5
draft
admin
13/08/2025
version 4
current
admin
13/08/2025
Verified draft
Delete
Edit

Configuration in the draft status

Schedule

The ***Configuration → Schedule*** tab allows you to select a schedule for various operations related to backups, such as creating various types of backups and restore points, data cleanups, etc. The tab is available starting with ADBM 2.5.0, before that schedules were part of a backup configuration and can be filled in along with other settings in the ***Configuration → General configuration*** section.



Backup manager / Backup / Cluster / deadbeef-feed-beef-dead-baddadfeedad / Configuration / Schedule
Backup manager
primary_cluster baddadfeedad
Backup configuration
Schedule
Storages
Save
Revert

Full Backup schedule: 0 30 0 ? * SAT *
At 00:30, only on Saturday (UTC)
Differential backup schedule
Incremental backup schedule
Restore point creation schedule
Cleanup schedule: 0 0 20 * * ?
At 20:00 (UTC)

The Configuration → Schedule tab

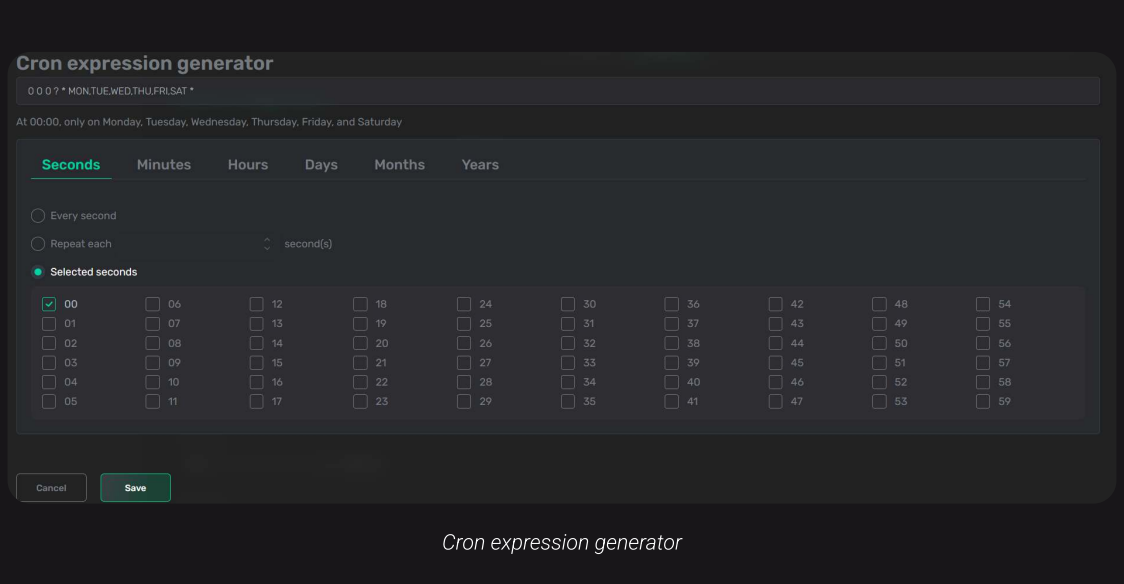
Below is the description of schedules that can be filled in on the ***Configuration → Schedule*** tab.

The "Configuration → Schedule" tab fields

Parameter	Description
Full Backup schedule	A schedule for automatic creation of full backups. If you are going to create backups only manually, do not activate the switcher that is located next to the field
Differential backup schedule	A schedule for automatic creation of differential backups. If you are going to create backups only manually, do not activate the switcher that is located next to the field
Incremental backup schedule	A schedule for automatic creation of incremental backups. If you are going to create backups only manually, do not activate the switcher that is located next to the field
Restore point creation schedule	A schedule for automatic creation of restore points. If you are going to create restore points only manually, do not activate the switcher that is located next to the field
Cleanup schedule	A schedule for deleting extra backups (see ***Number of full backups*** and ***Number of differential backups*** above) and backups with the `failed` and `invalid` statuses. If you are going to run the cleanup action only manually, do not activate the switcher that is located next to the field

To fill in any of the schedules mentioned above, activate the corresponding switch and enter the `cron` expression in the field that becomes available.

To switch to the alternative schedule fill mode, click the icon. As a result, the ***Cron expression generator*** window opens where you can set a schedule by selecting seconds, minutes, hours, days, months, and years. The selected values are automatically transformed into the `cron` expression that will be used as the field value. You can see it in the top part of the window along with its description. To save the current expression and return to the ***Configuration → Schedule*** main form, click ***Save***.



Cron expression generator
Seconds
Minutes
Hours
Days
Months
Years
Cancel
Save

0 0 0 ? * MON,TUE,WED,THU,FRI,SAT *
At 00:00:00, only on Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday

Cron expression generator

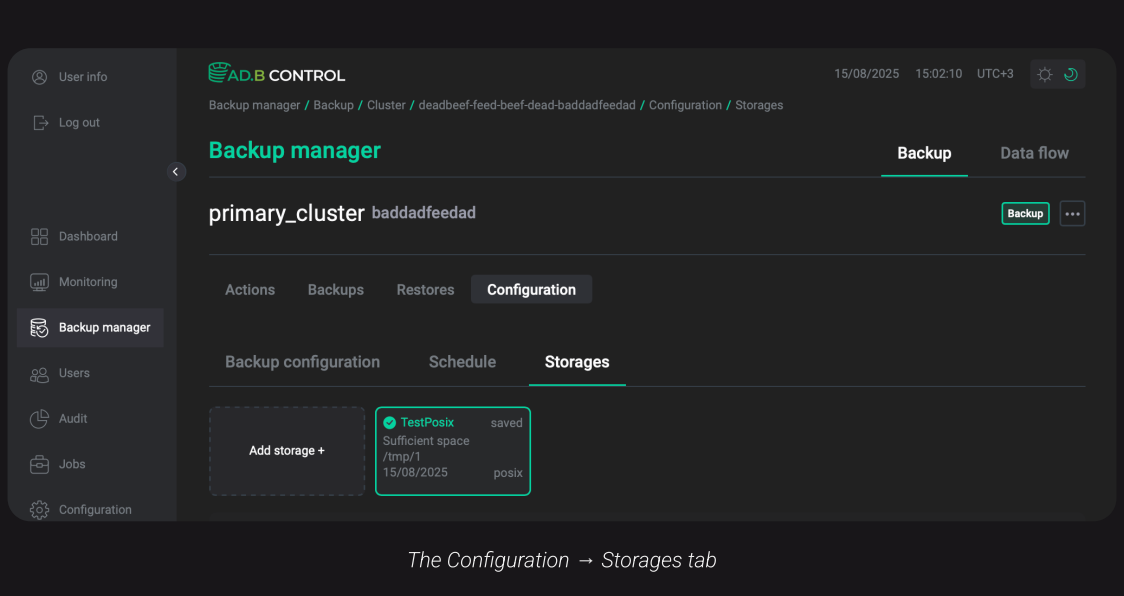
To save changes made on the ***Configuration → Schedule*** tab, click ***Save***. To undo changes, click ***Revert***.

Storages

In ADBM, a backup **storage** is a location where your data backup is stored. Starting with ADBM 2.6.5, it's a separate entity that you can configure, edit, and share among different backup configurations. To use a storage for backup, you can select it in the ***Storage*** section of your backup configuration.

Add a new storage

1. Open the ***Configuration → Storages*** tab and click ***Add storage***.



AD.B CONTROL
15/08/2025 15:02:10 UTC+3
Backup manager / Backup / Cluster / deadbeef-feed-beef-dead-baddadfeedad / Configuration / Storages
Backup manager
primary_cluster baddadfeedad
Backup configuration
Schedule
Storages
Add storage +
TestPosix
saved
Sufficient space
/tmp/1
15/08/2025
posix

The Configuration → Storages tab

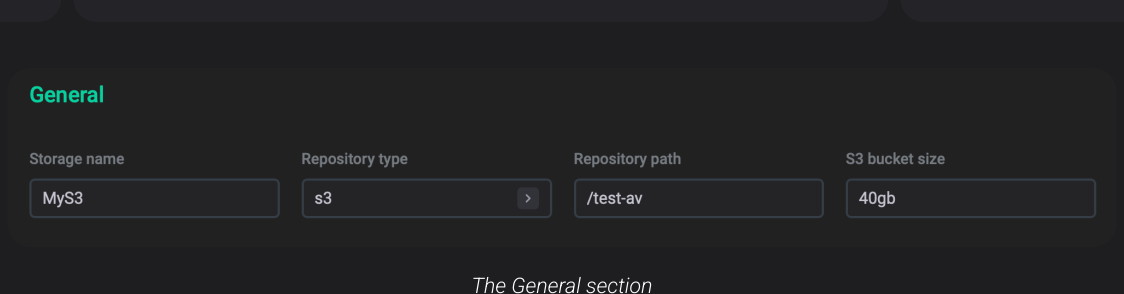
2. Fill in the parameters that are located in the ***Alert settings for storage*** and ***General*** sections described below. For S3 storages, you also need to fill in the parameters of the ***s3 configuration*** section.
3. Click ***Save***.

Alert settings for storage

Parameter	Description	Default value
Almost Full	The storage usage threshold (in percent) at which the storage configuration will have a yellow indicator and the `Almost Full` status	80
Critical	The storage usage threshold (in percent) at which the storage configuration will have a red indicator and the `Critical` status	90

General

Parameter	Description	Default value
Storage name	A unique name for the storage by which you can identify it when selecting a storage in a backup configuration	—
Repository type	A repository type. Possible values: - `posix` - `s3` - `cifs`	posix
Repository path	The repository where backups and archived WAL segments are to be stored. The repository directory should be pre-created and available from coordinator and segment hosts. You can use the NFS shared directory, which is connected to all cluster segments. Do not use local directories on segments if mirroring is switched on — in that case, the hosts with mirror segments won't have the required stanzas. Ensure that you set up the same repository path for complementary primary and mirror segments	—
S3 bucket size	The amount of storage space that you want to allocate for backups in the bucket. Displayed if `s3` is selected in the ***Repository type*** field of the ***General*** section. The parameter is required to calculate the S3 storage utilization metrics and status	—



General

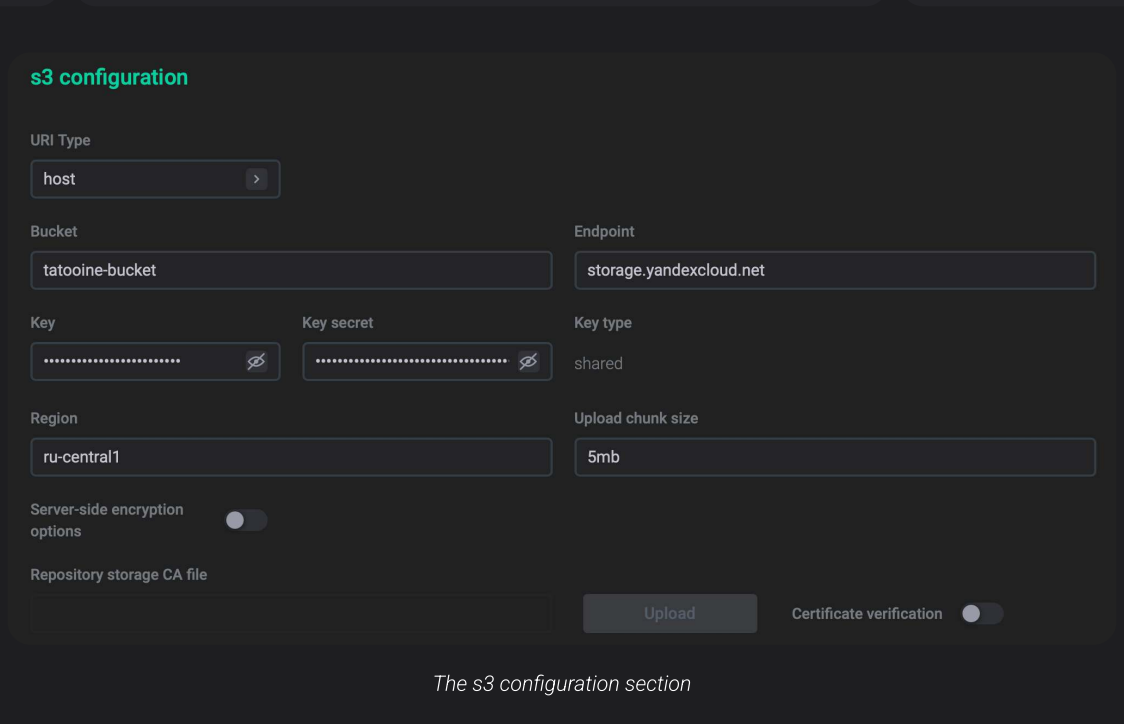
Storage name: MyS3
Repository type: s3
Repository path: /test-av
S3 bucket size: 40gb

The General section

s3 configuration

The following parameters are available if `s3` is selected in the ***Repository type*** field.

Parameter	Description	Default value
URI Type	An endpoint type (see ***Endpoint*** below). Displayed if `s3` is selected in the ***Repository type*** field. Possible values: - `host` - `path`	`host`
Bucket	A bucket name in the S3 storage. A bucket name cannot contain dots	—
Endpoint	An endpoint of the S3 storage	—
Key	A key to access the S3 storage	—
Key secret	A secret key to access the S3 storage	—
Region	A region of the S3 storage	—
Upload chunk size	Size of the uploaded data chunk (**part size** in S3 terms). Limitations shown in the tooltip (`64kb-1tb`) come from `pgbackrest`. Note that S3 storages may have their own restrictions	5mb
Server-side encryption options	Indicates whether to perform data encryption on the server side. After this switch is activated, the ***Type*** and ***Key*** fields become visible and mandatory (see below)	false
Type	A type of the server-side data encryption: - `kms-key-id` — using the specified AWS key management service key. - `sse-customer-key` — using the specified customer key. This field is mandatory if the ***Server-side encryption options*** switch is activated. You should also specify a key value in the ***Key*** field (see below)	—
Key	A key that is used for data encryption on the server side. The key type is specified in the ***Type*** field (see above). This field is mandatory if the ***Server-side encryption options*** switch is activated. The field becomes enabled after you select a value from the ***Type*** drop-down list	—
Repository storage CA file	Certificate Authority (CA) PEM file to access the S3 storage. To select the file, click ***Upload***	—
Certificate verification	Defines whether to verify a repository storage certificate. If you activate this switch, the ***Repository storage CA file*** becomes enabled	true



s3 configuration

URI Type: host
Bucket: tatooine-bucket
Endpoint: storage.yandexcloud.net
Key type: shared
Region: ru-central1
Upload chunk size: 5mb
Server-side encryption options
Repository storage CA file
Upload
Certificate verification

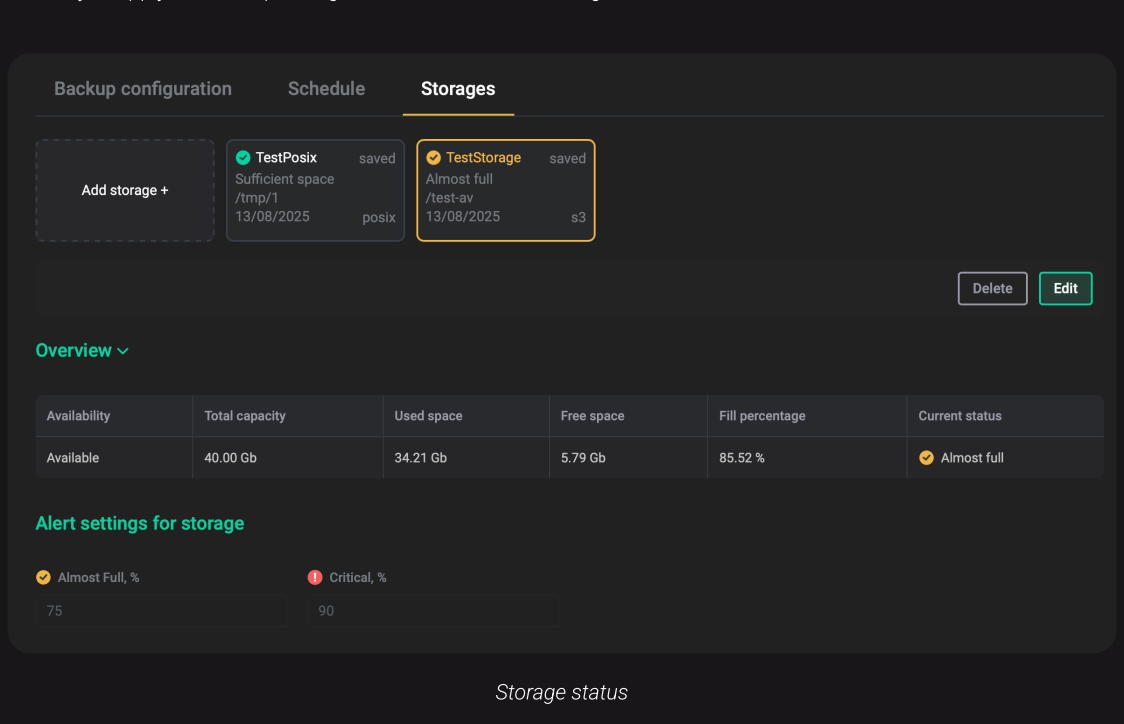
The s3 configuration section

Monitor storages

To monitor the storage status and space utilization, use the ***Overview*** section, which is available after applying the configuration with this storage. The ***Overview*** section allows you to monitor the following parameters:

- ***Availability***. Shows whether the storage is available. If ADBM cannot access the storage, it won't perform a backup.
- Space utilization: ***Total capacity***, ***Used space***, ***Free space***, and ***Fill percentage***.
  The fill percentage shows the ratio between the total capacity and used space. For POSIX and CIFS storages, the total capacity is calculated automatically. For S3 storages, you should manually specify the total capacity using the ***S3 bucket size*** field. If no value is set, the S3 storage utilization metrics and status won't be available.
- ***Current status***. Depending on the fill percentage value, each storage may have one of the following statuses:
  - `Sufficient space` — the fill percentage value is below the percentage configured under Alert settings for storage, in the ***Almost Full*** limit.
  - `Almost Full` — the fill percentage value is above the ***Almost Full*** limit, but below the ***Critical*** limit.
  - `Critical` — the fill percentage value is above the ***Critical*** limit. To avoid storage overflow during a backup, you can use the Fail on low space flag, which ensures the backup won't proceed if the storage is in the `Critical` state.

These metrics are first calculated when you apply the backup configuration that uses this storage. Afterward, the metrics and the status are checked and updated every minute.



Backup configuration
Schedule
Storages
Add storage +
TestPosix
saved
Sufficient space
/tmp/1
13/08/2025
posix
TestStorage
saved
Almost full
/test-av
13/08/2025
s3
Delete
Edit
Overview
Alert settings for storage

Availability	Total capacity	Used space	Free space	Fill percentage	Current status
Available	40.00 Gb	34.21 Gb	5.79 Gb	85.52 %	Almost full

Almost Full, %: 75
Critical, %: 90

Storage status

Run actions in ADBM

Daria Barysheva

Contents

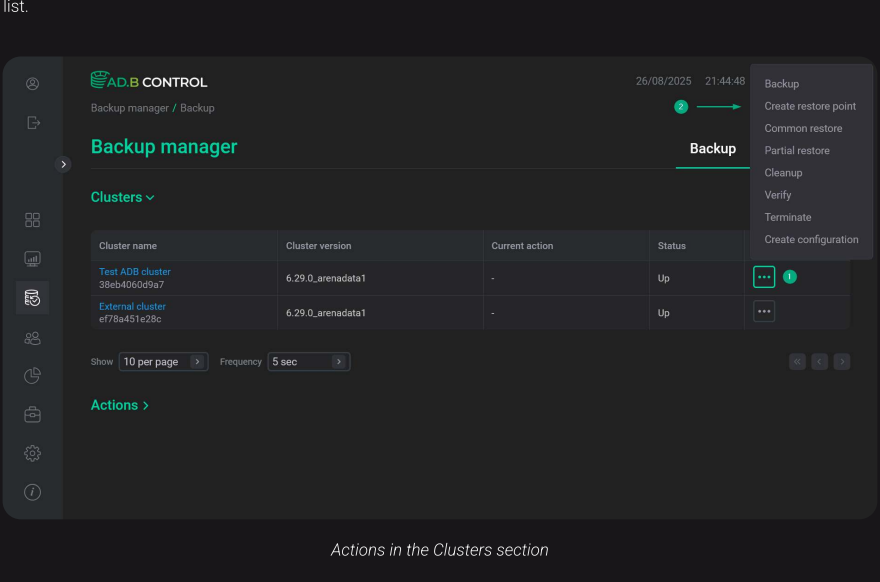
To Table of Contents

- Overview
- Backup
- Create restore point
- Common restore
- Partial restore
- Cleanup
- Verify
- Terminate
- Resume
- Create configuration

Overview

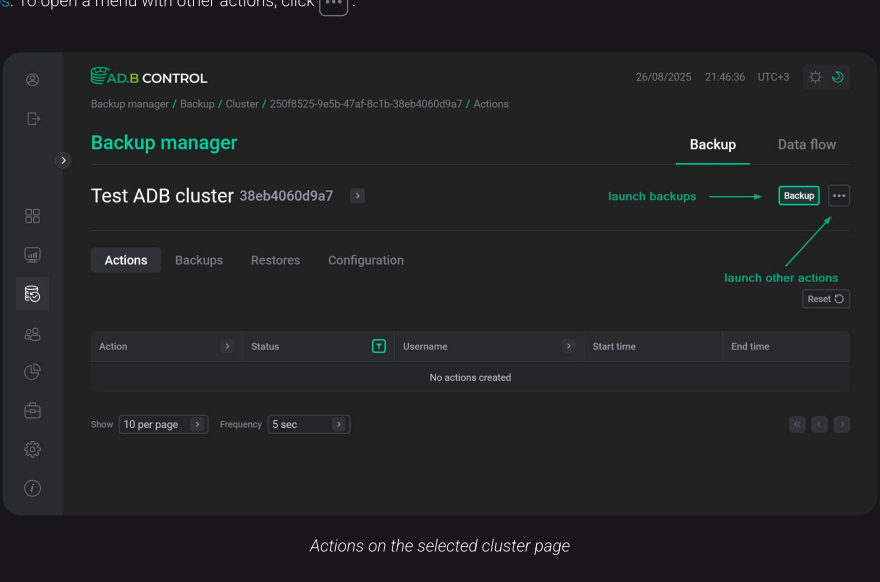
After you created a configuration, backups are launched and cleaned up according to the schedules that you defined. However, ADBM also allows you to manage backups manually via the cluster actions that are described below. To run the actions, follow the steps:

1. Open the Backup manager page via the ADB Control web interface.
2. In the **Clusters** section of the page that opens, find a cluster for which you want to run actions and click the ⋯ icon in the row that corresponds to that cluster. Possible actions are listed in the drop-down list.



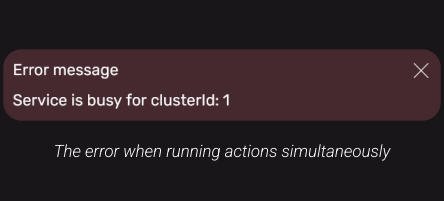
Actions in the Clusters section

3. If you want to work with a specified cluster, click the cluster name in the table that is located in the **Clusters** section. The selected cluster page opens. The **Backup** button is designed to launch backups. To open a menu with other actions, click ⋯.



Actions on the selected cluster page

Running actions is an exclusive operation. You cannot run another action until the previous one is completed (Terminate is exception). If you try to run several actions simultaneously, you get the following error.



Error message

Service is busy for clusterId: 1

The error when running actions simultaneously

> **IMPORTANT**
>
> Starting with ADBM 2.3.2, some of the actions mentioned below may be hidden in the menu if they are not allowed at the current time. In particular, after a cluster upgrade that requires verification or recreation of the backup configuration, only two actions are displayed in the drop-down menu: Verify and Create configuration. Access to other actions is possible only after you launch one of the allowed actions.

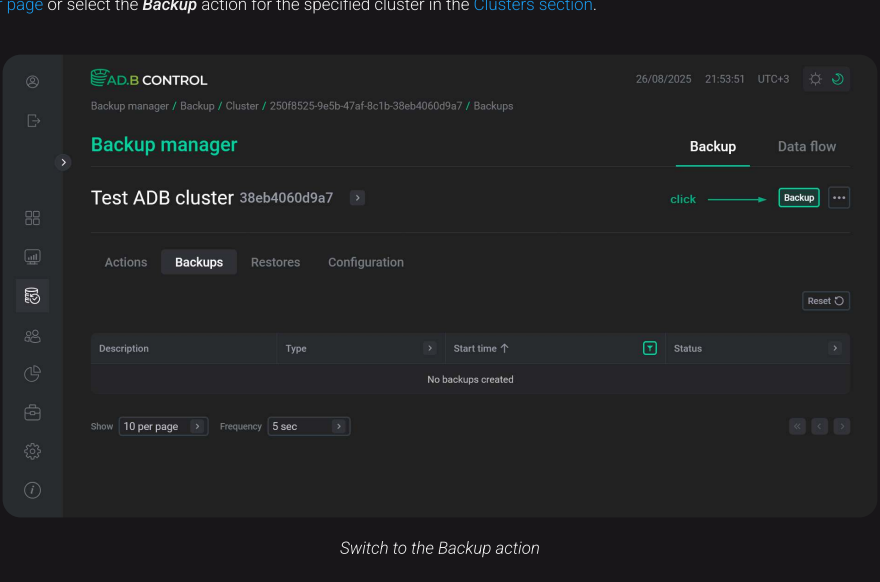
Backup

Database backups are launched automatically on the schedules that are specified by the following parameters of the current configuration version (see the Configuration → Schedule tab):

- ***Full Backup schedule*** — full backups;
- ***Differential backup schedule*** — differential backups;
- ***Incremental backup schedule*** — incremental backups.

If you need to get an additional backup of any type without waiting for the next auto launch, do the following:

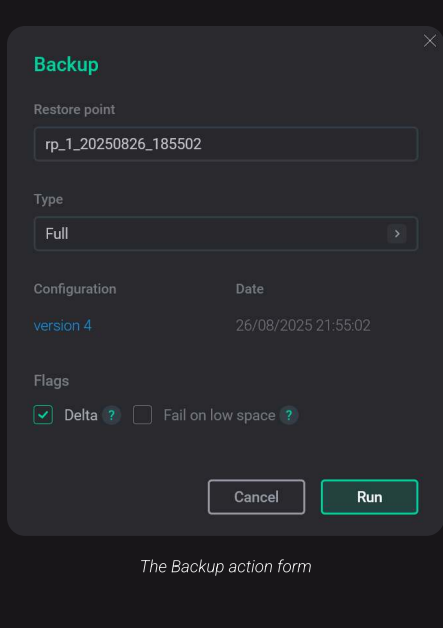
1. Click ***Backup*** on the cluster page or select the ***Backup*** action for the specified cluster in the Clusters section.



Switch to the Backup action

The window that opens contains the following fields:

- ***Restore point*** — an automatically generated name for a new restore point.
- ***Type*** — a backup type. Possible values:
  - `Full` — full;
  - `Incr` — incremental;
  - `Diff` — differential.
- ***Configuration*** — the configuration version that will be applied. You can click it to check or edit the configuration parameters in the list of configurations.
- ***Date*** — a current timestamp.
- ***Delta*** — a flag that indicates the need to use checksums during the `Incr` or `Diff` backup creation. It does not affect `Full` backups.
- ***Fail on low space*** — a flag that indicates that the backup won't run if the storage has the Critical status.



Backup

Restore point: rp_1_20250826_185502

Type: Full

Configuration: version 4

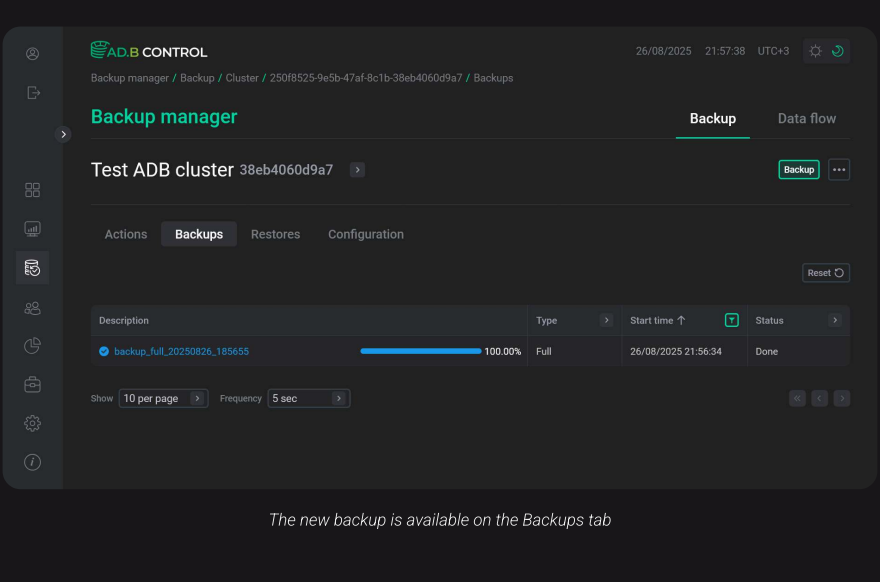
Date: 26/08/2025 21:55:02

Flags: ☑ Delta ☐ Fail on low space

Cancel Run

The Backup action form

2. Select the backup type in the ***Type*** field and change the ***Restore point*** value if necessary.
3. Click ***Run***.
4. As a result, the ***Backup*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
5. As a successful result of all actions, the new backup becomes available in the list of backups on the ***Backups*** tab.



The new backup is available on the Backups tab

Starting with ADBM 2.4.0, there is a progress bar next to the backup name in the ***Description*** column, which displays the backup execution progress (as a percentage of the estimated duration for all cluster segments). Note that backup progress is calculated only for the physical cluster reservation and does not include other actions.

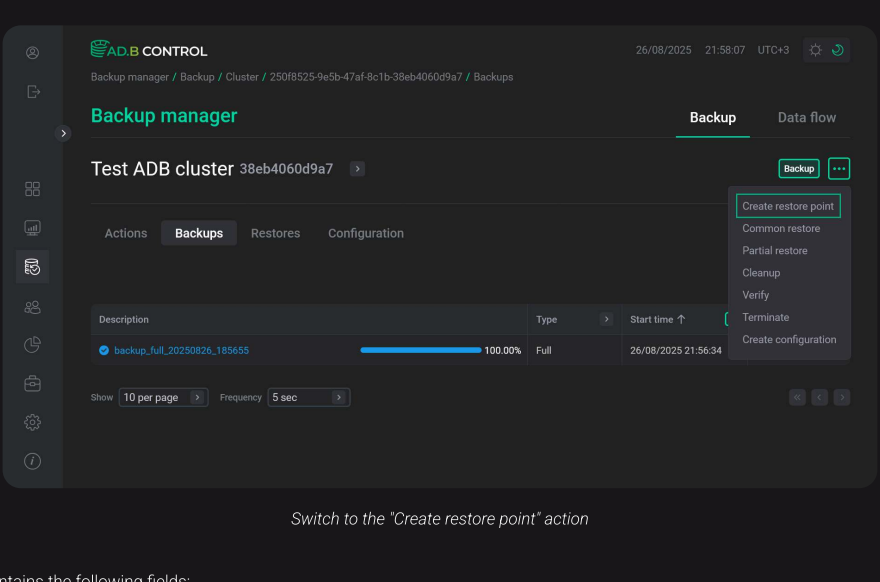
Create restore point

> **NOTE**
>
> The ***Create restore point*** action is available if there is at least one backup in the `Done` status.

Restore points are automatically created according to the schedule that is defined by the ***Restore point creation schedule*** parameter of the current configuration version (see the Configuration → Schedule tab).

If you need to create a restore point without waiting for the next auto launch, follow the steps:

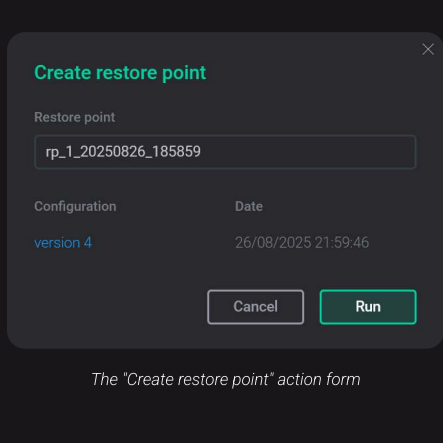
1. Select the ***Create restore point*** action on the cluster page or in the Clusters section.



Switch to the "Create restore point" action

The window that opens contains the following fields:

- ***Restore point*** — an automatically generated name for a new restore point.
- ***Configuration*** — the configuration version that will be applied. You can click it to check or edit the configuration parameters in the list of configurations.
- ***Date*** — a current timestamp.



Create restore point

Restore point: rp_1_20250826_185859

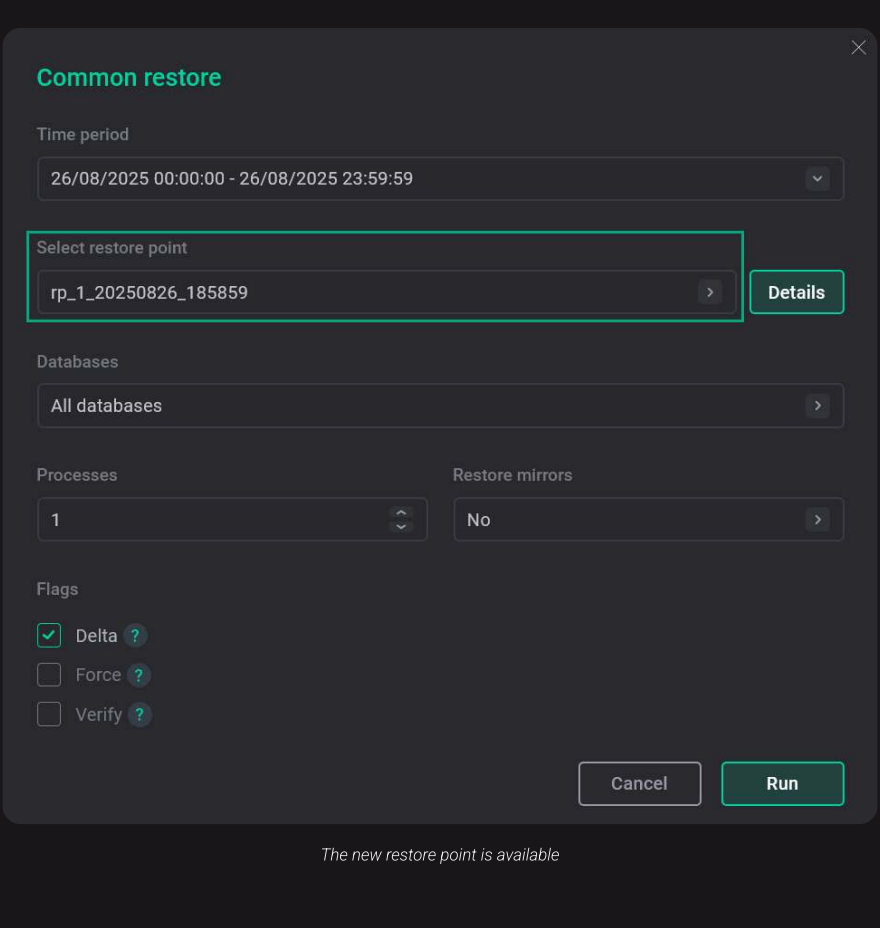
Configuration: version 4

Date: 26/08/2025 21:59:46

Cancel Run

The "Create restore point" action form

2. Change the restore point name in the ***Restore point*** field if necessary.
3. Click ***Run***.
4. As a result, the ***Create restore point*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
5. As a successful result of all actions, a new restore point becomes available. You can check it by running the Common restore, Partial restore, or Cleanup action.



Common restore

Time period: 26/08/2025 00:00:00 - 26/08/2025 23:59:59

Select restore point: rp_1_20250826_185859

Details

Databases: All databases

Processes: 1

Restore mirrors: No

Flags: ☑ Delta ☐ Force ☐ Verify

Cancel Run

The new restore point is available

Common restore

The ***Common restore*** action allows you to restore databases at the moment of one of existing restore points. This is the only action that can be applied to both a running and a stopped ADB cluster. The action logic depends on the cluster status:

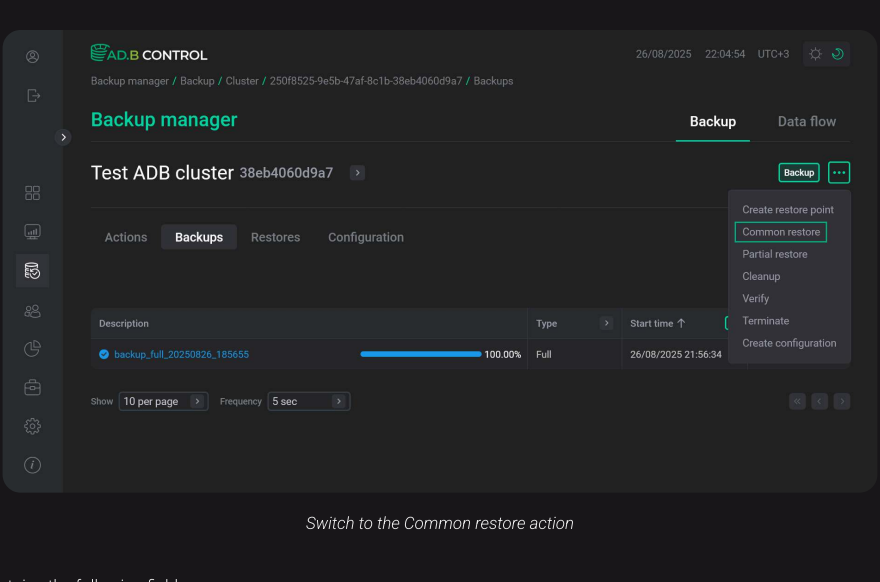
- **ADB cluster is running (marked as Up in the ADBM UI).** In this case, before the action is started, additional checks are automatically performed to ensure that the selected backup can be applied to the cluster:
  - The topology that is saved in the backup is compared with the topology obtained from the ADB cluster (from the system table `gp_segment_configuration`).
  - The backup metadata is also checked (see Verify below).

  If the validation is successful, the cluster stops automatically. In addition, if any error is fixed during the ***Common restore*** action, you can run Restore retry — this is a restart of the last restore process without any additional checks.
- **ADB cluster is stopped (marked as Down in the ADBM UI).** In this case, any validation is skipped. However, it is still possible to start the restore process. This method is recommended only in case of fatal problems when the cluster does not even start.

> **NOTE**
>
> Starting with ADBM 2.5.0, you can also restore data in a stopped cluster using a data flow with the `Copy` type.

To restore databases, follow the steps:

1. Select the ***Common restore*** action on the cluster page or in the Clusters section.



Switch to the Common restore action

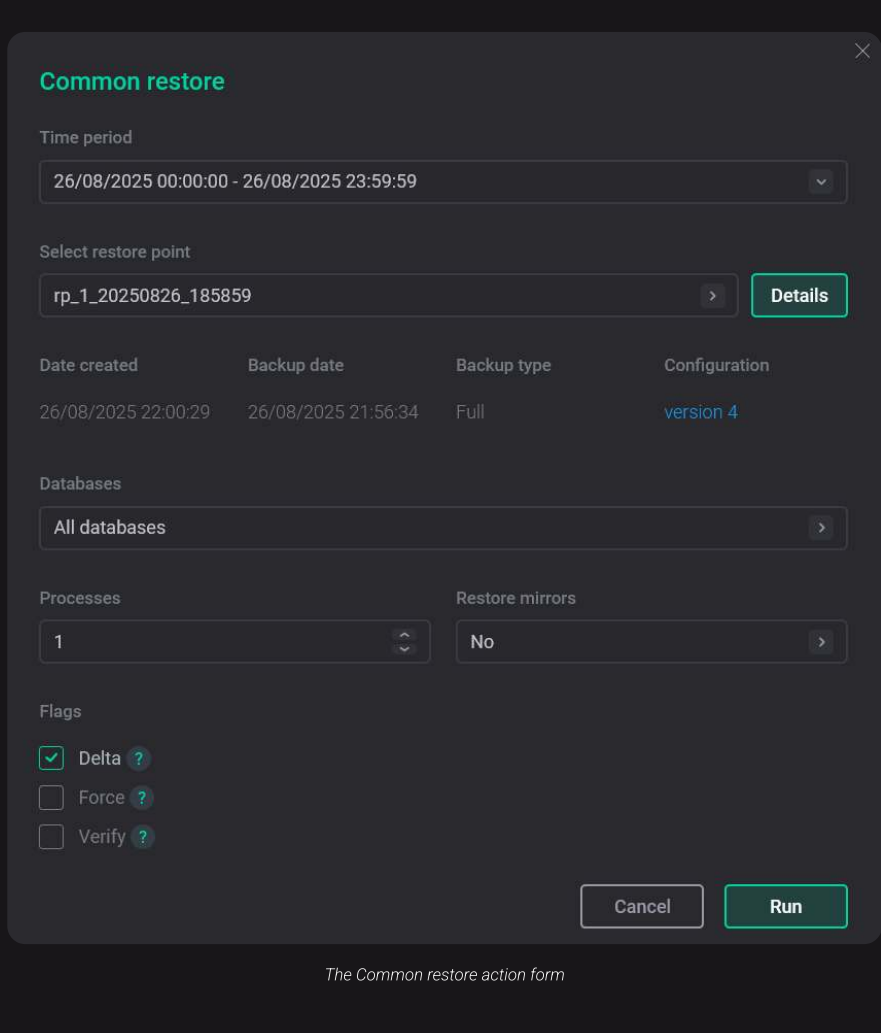
The window that opens contains the following fields:

- ***Time period*** — the timestamp range that is used to search restore points. The default value is the current date.
- ***Select restore point*** — the restore point at the moment of which you want to restore databases.
- ***Databases*** — the databases that you can restore for the selected restore point.
- ***Processes*** — a maximum number of processes to restore one segment.
- ***Restore mirrors*** — the drop-down list, which value indicates whether to restore data for mirrors and standby if they exist in the cluster:
  - `No` — do not restore mirrors.
  - `Parallel` — restore mirrors concurrently with primary segments.
  - `Sequential` — restore mirrors after all primary segments are restored.
- ***Flags***:
  - ***Delta*** — a flag that indicates the need to use checksums during the data restore from the `Incr` or `Diff` backup. It does not affect `Full` backups.
  - ***Force*** — a flag that indicates the need to completely overwrite data and tablespace paths in PostgreSQL.
  - ***Verify*** — a flag that indicates the need to check backup data before data restore.

> **IMPORTANT**
>
> - If the ***Delta*** flag is set and the ***Force*** flag is unset, checksums will be verified when restoring files with data, and only files that have been updated since the backup creation will be overwritten.
> - If both flags ***Delta*** and ***Force*** are set, data files will be compared based on their sizes and modification dates. Although the restore process in such case is quicker, it provides fewer assurances of integrity than the checksum check.

After you click ***Details***, you can also see additional fields:

- ***Date created*** — a creation timestamp of the selected restore point.
- ***Backup date*** — a creation timestamp of the latest backup that was completed before the selected restore point.
- ***Backup type*** — a type of the latest backup that was created before the selected restore point.
- ***Configuration*** — the configuration version that was applied to the selected restore point. You can click it to view the configuration parameters in the list of configurations.



Common restore

Time period: 26/08/2025 00:00:00 - 26/08/2025 23:59:59

Select restore point: rp_1_20250826_185859

Details

Date created	Backup date	Backup type	Configuration
26/08/2025 22:00:29	26/08/2025 21:56:34	Full	version 4

Databases: All databases

Processes: 1

Restore mirrors: No

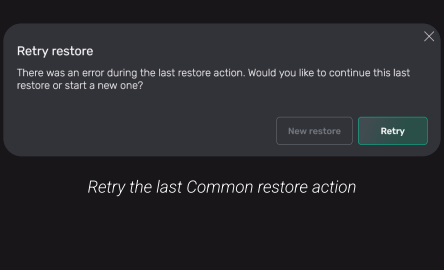
Flags: ☑ Delta ☐ Force ☐ Verify

Cancel Run

The Common restore action form

2. Fill in the necessary fields:
   - Select a restore point in the ***Select restore point*** drop-down-list. Use the ***Time period*** filter if necessary.
   - In the ***Databases*** drop-down-list, select the databases that you want to restore.
   - Select the ***Restore mirrors*** list value.
   - If necessary, set the ***Delta*** and ***Force*** flags.
3. Click ***Run***.
4. As a result, the ***Restore*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
5. As a successful result of all actions, the selected databases should be restored at the moment of the specified restore point.

For running ADB clusters (with the `Up` status), if any error occurs when running the ***Common restore*** action — you can retry the latest restore process. To do this, select the ***Common restore*** action again (see step 1 above) and click ***Retry*** in the window that opens. If you want to run a new ***Common restore*** action from scratch (with all checks), click ***New restore***.



Retry restore

There was an error during the last restore action. Would you like to continue this last restore or start a new one?

New restore Retry

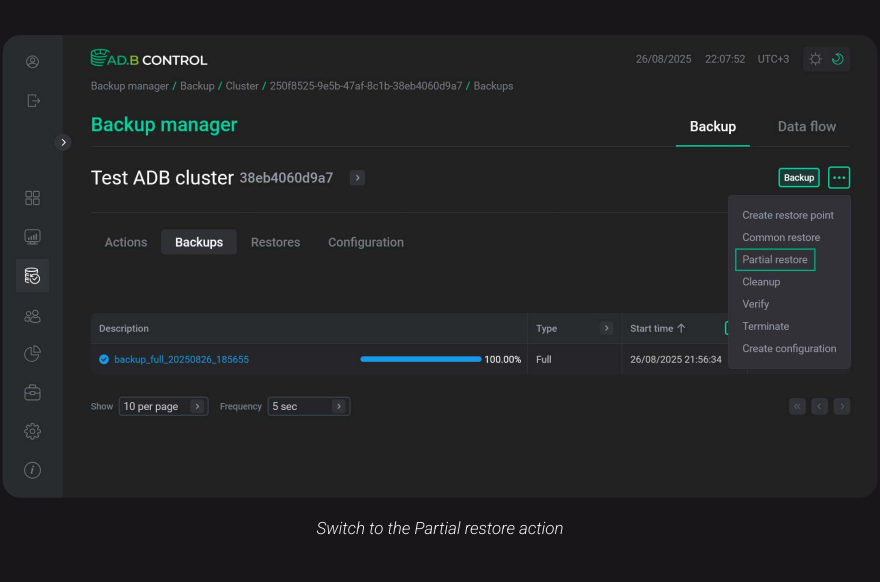
Retry the last Common restore action

Partial restore

> **NOTE**
>
> The ***Partial restore*** action is available starting with ADBM 2.2.2.

The ***Partial restore*** action allows you to restore specific tables at the moment of one of existing restore points. To restore tables, follow the steps:

1. Select the ***Partial restore*** action on the cluster page or in the Clusters section.



Switch to the Partial restore action

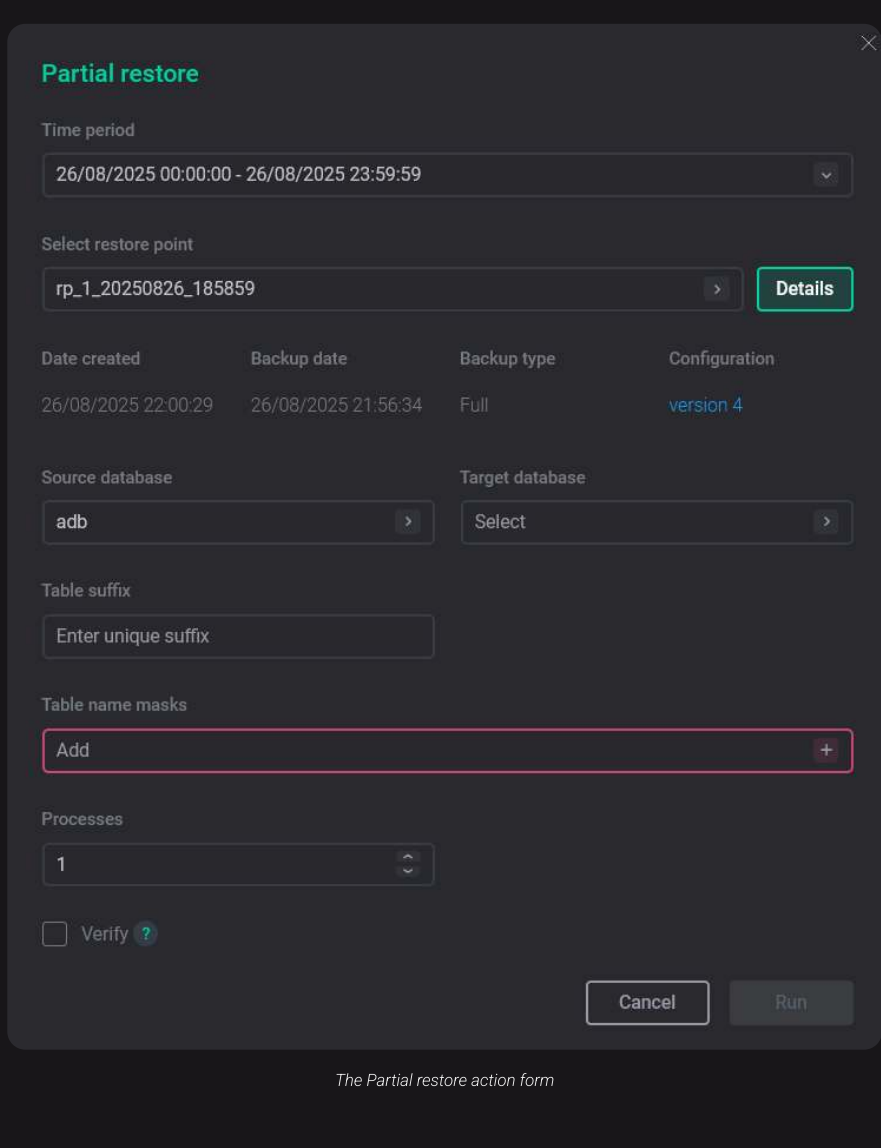
The window that opens contains the following fields:

- ***Time period*** — the timestamp range that is used to search restore points. The default value is the current date.
- ***Select restore point*** — the restore point at the moment of which you want to restore tables.
- ***Source database*** — the source database whose tables you can restore for the selected restore point.
- ***Target database*** — the target database where tables should be restored. If not specified, all tables will be restored in the source database.
- ***Table suffix*** — the suffix that will be added to all table names in the target database during the restore process. Maximum length is 32 symbols. If not specified, table names from the source database will be used. In the latter case, for successful data restore, it is important that the selected tables do not exist in the target database.
- ***Table name masks*** — an array of name masks for tables that should be restored. Each array element requires the following format: `<schema>/<table>`, where `<schema>` is a schema name template and `<table>` is a table name template. The `<table>` template may contain the `*` wildcards. Thus, each array element should match the following regular expression: `\w+(/[a-zA-Z_*]+)`. If `<schema>/` is absent, the `public` schema is used. To enter each mask, fill in the ***Add*** field and click `+`. To delete a mask, click `×`.
- ***Processes*** — a maximum number of parallel processes per host during the restore process. Values from the following range are supported: `[1,100]`. The default value comes from the ***Max processes per host*** field in the backup configuration (see the ***Workspace*** section in Manage configurations in ADBM).
- ***Verify*** — a flag that indicates the need to check backup data before data restore.

> **TIP**
>
> Since one process restores data of one segment on a host, it is recommended to set the maximum value equal to the number of primary segments on the host. If the value is higher, the rest of the processes will not be used, if the value is lower — the data restore process will take longer, yet consume less resources (due to the limit).

After you click ***Details***, you can also see additional fields:

- ***Date created*** — a creation timestamp of the selected restore point.
- ***Backup date*** — a creation timestamp of the latest backup that was completed before the selected restore point.
- ***Backup type*** — a type of the latest backup that was created before the selected restore point.
- ***Configuration*** — the configuration version that was applied to the selected restore point. You can click it to view the configuration parameters in the list of configurations.



Partial restore

Time period: 26/08/2025 00:00:00 - 26/08/2025 23:59:59

Select restore point: rp_1_20250826_185859

Details

Date created	Backup date	Backup type	Configuration
26/08/2025 22:00:29	26/08/2025 21:56:34	Full	version 4

Source database: adb

Target database: Select

Table suffix: Enter unique suffix

Table name masks: Add

Processes: 1

☐ Verify

Cancel Run

The Partial restore action form

2. Fill in the necessary fields:
   - Select a restore point in the ***Select restore point*** drop-down-list. Use the ***Time period*** filter if necessary.
   - In the ***Source databases*** drop-down-list, select the database whose tables you want to restore.
   - If necessary, fill in the ***Target database*** and ***Table suffix*** fields.
   - Add an array of table name templates in the ***Table name masks*** field.
   - Change the default value of the ***Processes*** field if needed.

> **CAUTION**
>
> - If the ***Table suffix*** value is empty and a table with the source name already exists in the target database, the following error occurs in one of subactions: `Couldn't create table <table_name>`. In that case, the table data restore fails (the process of restoring other tables still continues). To avoid such a problem, use a suffix in case of existence of the table being restored.
> - When choosing the ***Table suffix*** value, remember that the maximum length of any DB relation name (including partition names) in ADB is `64` characters. If the overall length of the name with the suffix exceeds the specified limit, the name will be trimmed, which could potentially lead to duplication of table names.
> - If a source database does not contain tables with names matching the ***Table name masks*** array elements (for example, if a non-existing schema is specified), the following error occurs in one of subactions: `No tables were found by provided filter`, and the restore status is changed to `Failed`. If at least one mask specified in the ***Table name masks*** field is valid, the tables corresponding to this mask are successfully restored and invalid masks are ignored. To avoid such situations, ensure that all schemas and tables are present in the source database before launching the ***Partial restore*** action.

3. Click ***Run***.
4. As a result, the ***Table restore*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
5. As a successful result of all actions, the selected tables should be restored at the moment of the specified restore point. If at least one table is restored with a failure, the ***Table restore*** action status changes to `Warning`, and the restore status of each table can be seen on the ***Restores*** page — in details of the selected data restore on the Tables tab.

Cleanup

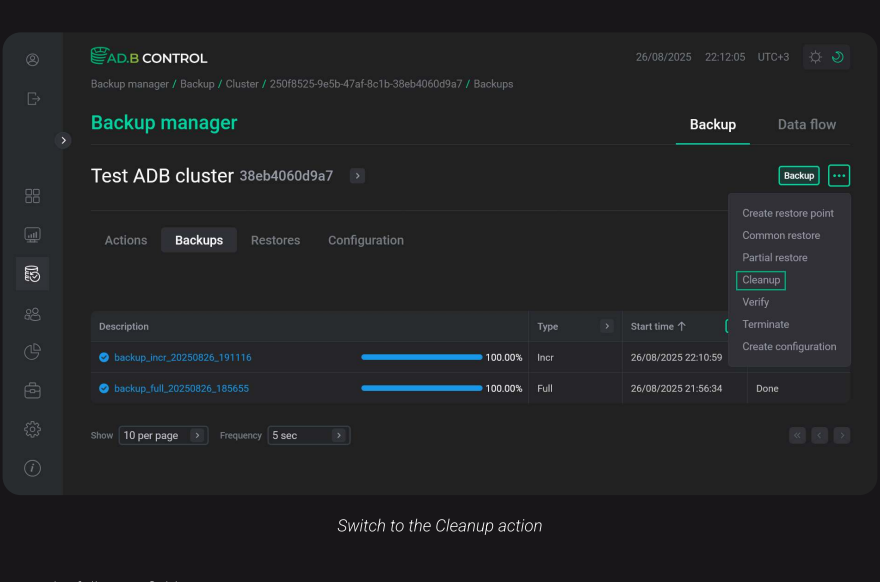
The ***Cleanup*** action is automatically executed according to the schedule that is defined by the ***Cleanup schedule*** parameter of the current configuration version (see the Configuration → Schedule tab). This action removes:

- The oldest extra full backups — if the number of full backups exceeds the ***Number of full backups*** parameter defined for the current configuration.
- The oldest extra differential backups — if the number of differential backups exceeds the ***Number of differential backups*** parameter defined for the current configuration.
- All backups in the following statuses: `Failed`, `Invalid`.
- Inactive configuration versions that have no related backups.

When you run the ***Cleanup*** action manually, it deletes all objects mentioned above but also allows you to remove restore points and backups related to the specified restore point. You define the restore point, and ADBM finds the latest full or differential backup that was completed before the selected point created. Then, all backups and restore points related to that backup are removed.

To run the ***Cleanup*** action manually, follow the steps:

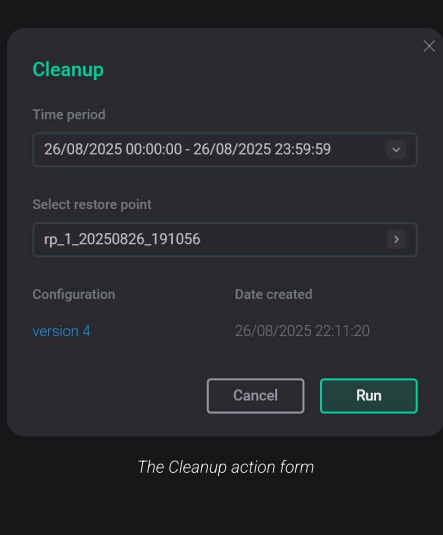
1. Select the ***Cleanup*** action on the cluster page or in the Clusters section.



Switch to the Cleanup action

The window that opens contains the following fields:

- ***Time period*** — the timestamp range that is used to search restore points. The default value is the current date.
- ***Select restore point*** — the restore point at the moment of which you want to remove a set of related backups and restore points.
- ***Configuration*** — the configuration version that was applied to the selected restore point. You can click it to view the configuration parameters in the list of configurations.
- ***Date created*** — a creation timestamp of the selected restore point.



Cleanup

Time period: 26/08/2025 00:00:00 - 26/08/2025 23:59:59

Select restore point: rp_1_20250826_191056

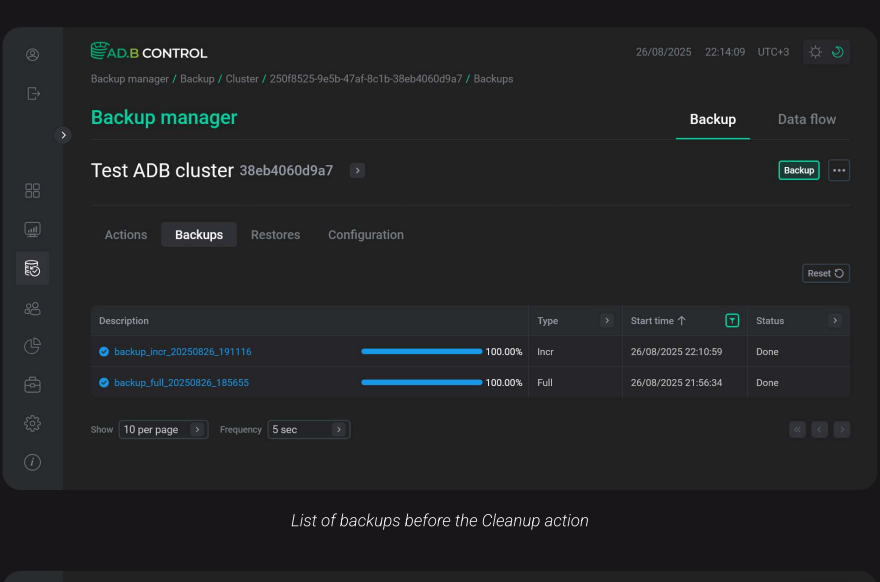
Configuration: version 4

Date created: 26/08/2025 22:11:20

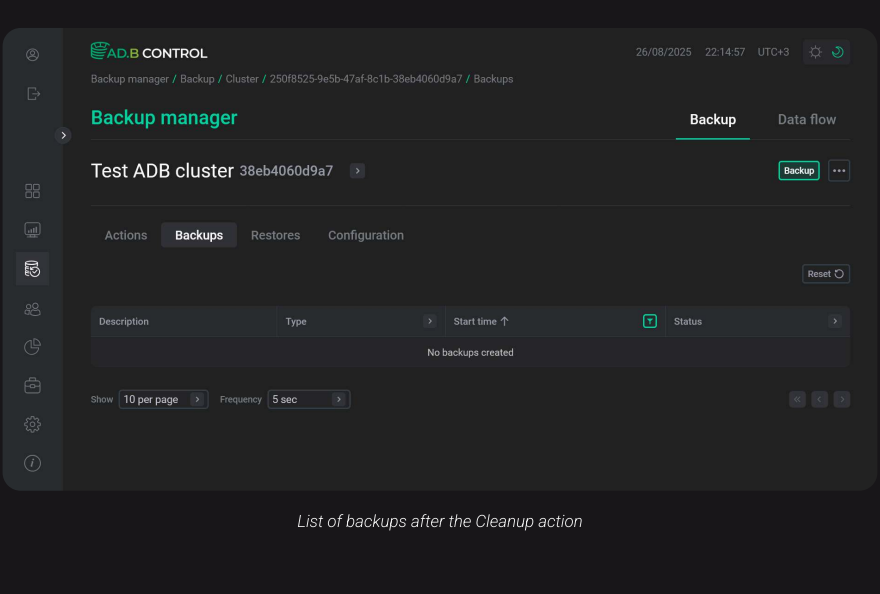
Cancel Run

The Cleanup action form

2. Select a restore point in the ***Select restore point*** drop-down-list. Use the ***Time period*** filter if necessary.
3. Click ***Run***.
4. As a result, the ***Start cleanup job for Verify backup*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
5. As a successful result of all actions, backups and restore points are removed according to the rules described above. You can check it in the list of backups on the ***Backups*** tab. For example, if you have two backups, full and incremental, and select the restore point that was created after the incremental backup — both backups will be deleted.

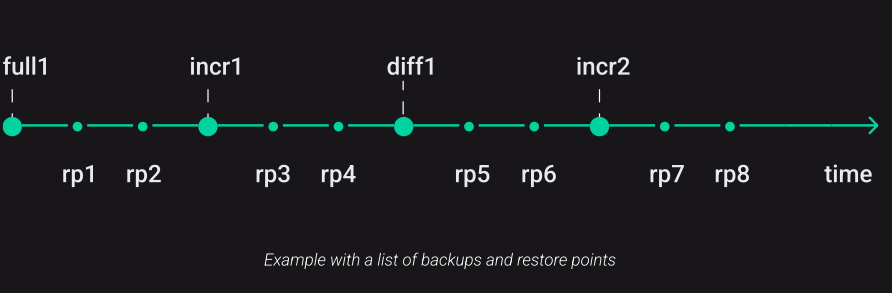


List of backups before the Cleanup action



List of backups after the Cleanup action

To better understand the logic of the ***Cleanup*** manual action, you can view the example listed below. The figure shows a timeline with a list of backups of different types (called `full*`, `diff*`, `incr*`) and restore points (called `rp*`). The table explains which objects will be deleted when any of restore points is selected in the Cleanup action form.



Example with a list of backups and restore points

Logic of the manual Cleanup action

Selected restore point	Objects to be removed
rp1	All backups and restore points for the specified period
rp2	All backups and restore points for the specified period
rp3	All backups and restore points for the specified period
rp4	All backups and restore points for the specified period
rp5	diff1, incr2, rp5 — rp8
rp6	diff1, incr2, rp5 — rp8
rp7	diff1, incr2, rp5 — rp8
rp8	diff1, incr2, rp5 — rp8

Verify

The ***Verify*** action performs the following:

- Compares the backup metadata that is stored in ADBM with the backup metadata that is being received from `pgbackrest` (i.e. actual backups).
- Checks whether the backups and archives in a repository are valid on the moment of the selected restore point. The following checks are made:
  - Existence of backup files.
  - Size of backup files.
  - Checksums of backup files.
  - Validity of WAL archives that belong to the backup.

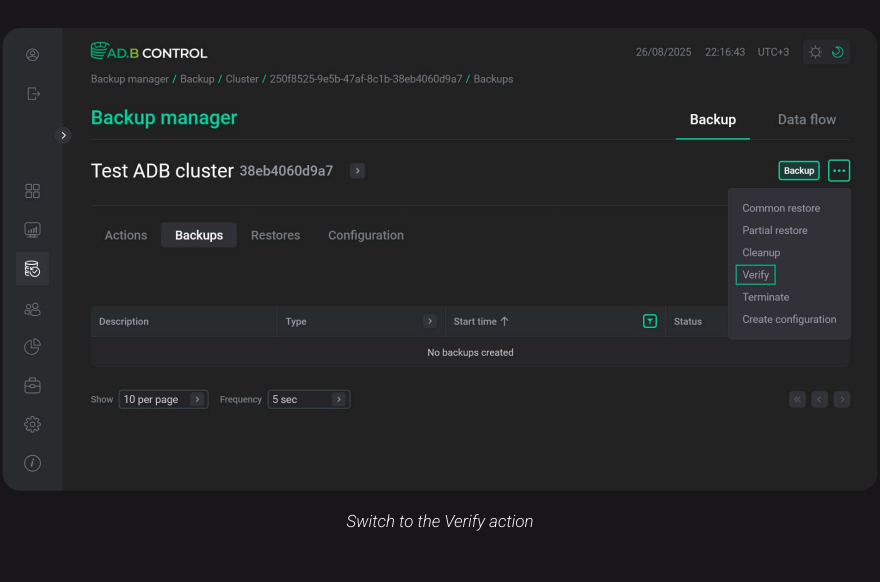
If the difference is fixed for any backups, they acquire the `Invalid` status as they cannot be used to restore databases. Such backups will be removed during the next Cleanup action launch.

Before comparing data and metadata, ADBM additionally checks existence of configuration files (*pgbackrest.conf*) for all timelines on all segment hosts. If any of these files are not present, the last available version of the timeline configuration is taken, and a new configuration file is created on the basis of this configuration. Then this file is written to the segment host for the selected timeline. This is necessary to ensure that all configurations are present before backup data and metadata are checked.

> **NOTE**
>
> - If the repository is inaccessible during the ***Verify*** action execution, backups are transferred to the `Unknown` status. After the repository becomes available, these backups can be marked as `Done` as a result of the ***Verify*** action.
> - Backup data is checked starting with ADBM 2.5.0 via the `pgbackrest verify` command. Before that, the ***Verify*** action checked only backup metadata using the `pgbackrest info` command.
> - It is recommended to set the immutable attribute on the local NFS mount directory to ensure correct detection of repository availability.

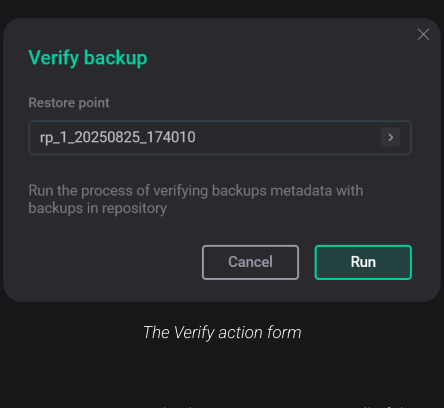
To run the ***Verify*** action, do the following:

1. Select the ***Verify*** action on the cluster page or in the Clusters section.



Switch to the Verify action

2. In the window that opens, select a restore point (on the moment of which creation you need to check backup data) and click ***Run***.



Verify backup

Restore point: rp_1_20250825_174010

Run the process of verifying backups metadata with backups in repository

Cancel Run

The Verify action form

3. As a result, the ***Verify backup*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
4. As a successful result of all actions, some of the backups can acquire the `Invalid` or `Unknown` status. You can search such backups using the ***Status*** filter on the Backups tab.

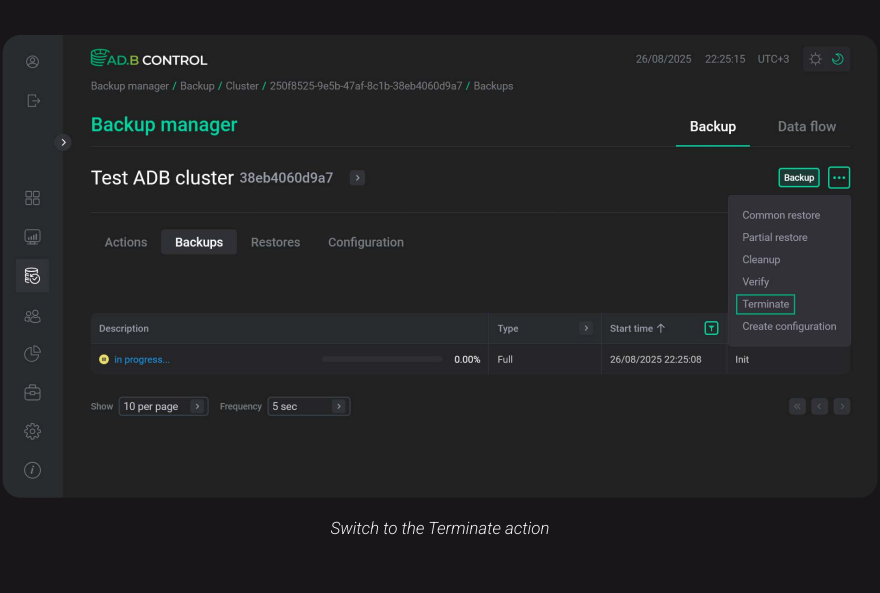
Terminate

The ***Terminate*** action allows you to terminate some of the currently running actions. This is the only action type for which a separate distributed lock is used (in etcd), which allows you to run ***Terminate*** in parallel with other types of actions.

> **IMPORTANT**
>
> - You can terminate actions of the following types:
>   - Backup
>   - Cleanup
> - After applying the ***Terminate*** action, you can resume the latest terminated backup via the Resume action.

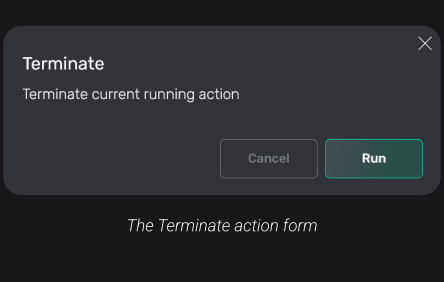
Below is an example how to terminate the ***Backup*** action:

1. Run the ***Backup*** action as shown above.
2. Select the ***Terminate*** action on the cluster page or in the Clusters section.



Switch to the Terminate action

3. In the window that opens, click ***Run***.



Terminate

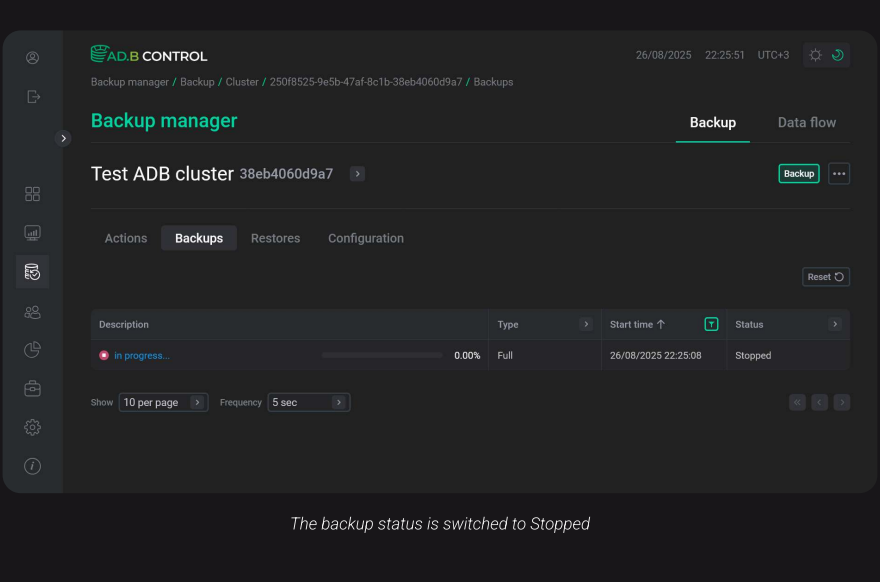
Terminate current running action

Cancel Run

The Terminate action form

4. As a result, the ***Terminate*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM). The current backup acquires the `Terminating` status. You can check it in the list of backups on the ***Backups*** tab.

   As a successful result of all actions, the backup acquires the `Stopped` status.



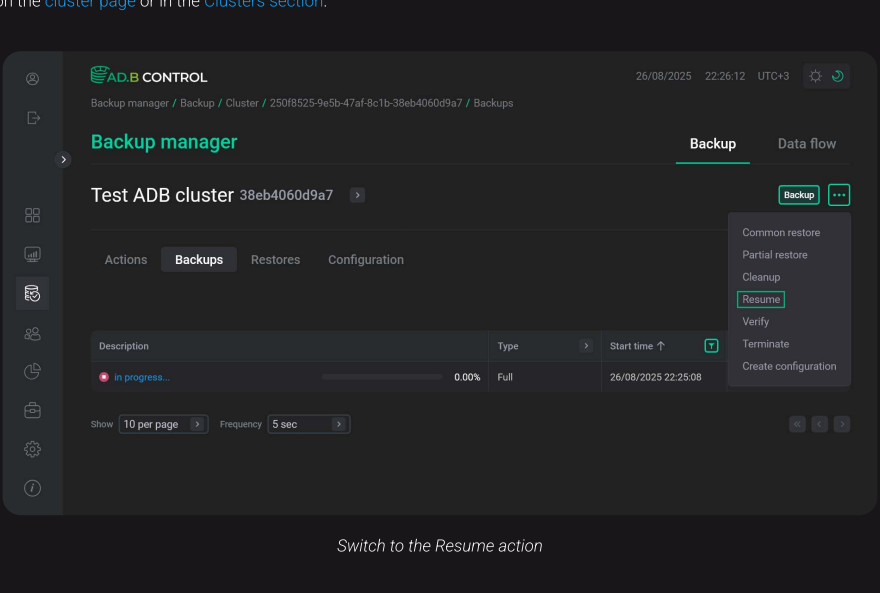
The backup status is switched to Stopped

Resume

> **IMPORTANT**
>
> - Starting with ADBM 2.1.2 the ***Resume*** action is available only for `Full` backups.
> - Bundled files cannot be reused when a backup is resumed. The number of files that can be reused depends on the Bundle limit value configured in the ***Storages*** section of the backup configuration: the higher the value, the fewer files are available for reuse when resuming a backup.

The ***Resume*** action allows you to resume the latest terminated backup launch. This action is available in the list of cluster actions only when backups with the `Stopped` status exist. Below is an example how to resume the backup that was terminated in the previous section:

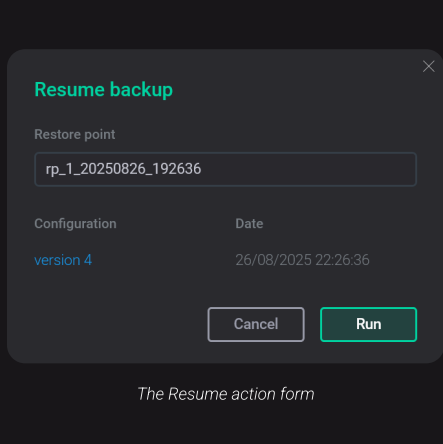
1. Select the ***Resume*** action on the cluster page or in the Clusters section.



Switch to the Resume action

The window that opens contains the following fields:

- ***Restore point*** — an automatically generated name for a new restore point.
- ***Configuration*** — the configuration version that will be applied. You can click it to check the configuration parameters in the list of configurations.
- ***Date*** — a current timestamp.



Resume backup

Restore point: rp_1_20250826_192636

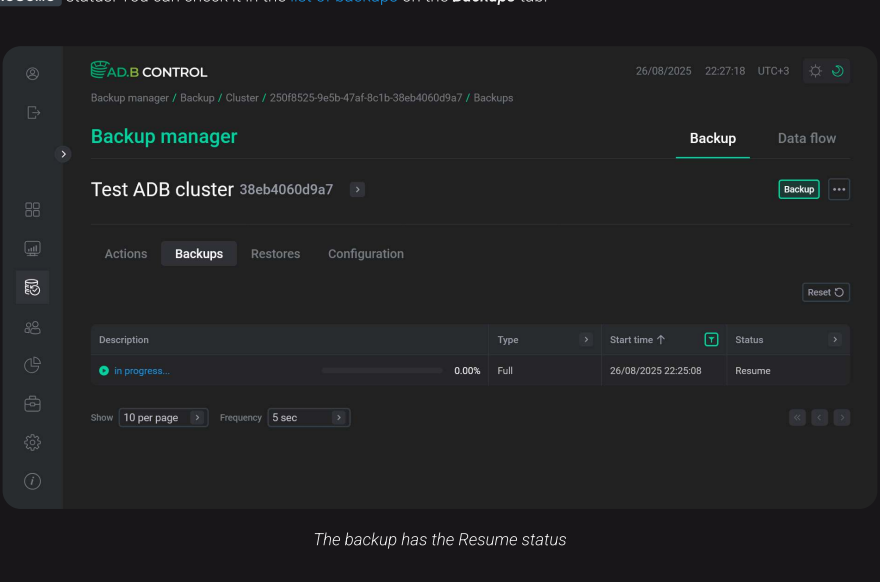
Configuration: version 4

Date: 26/08/2025 22:26:36

Cancel Run

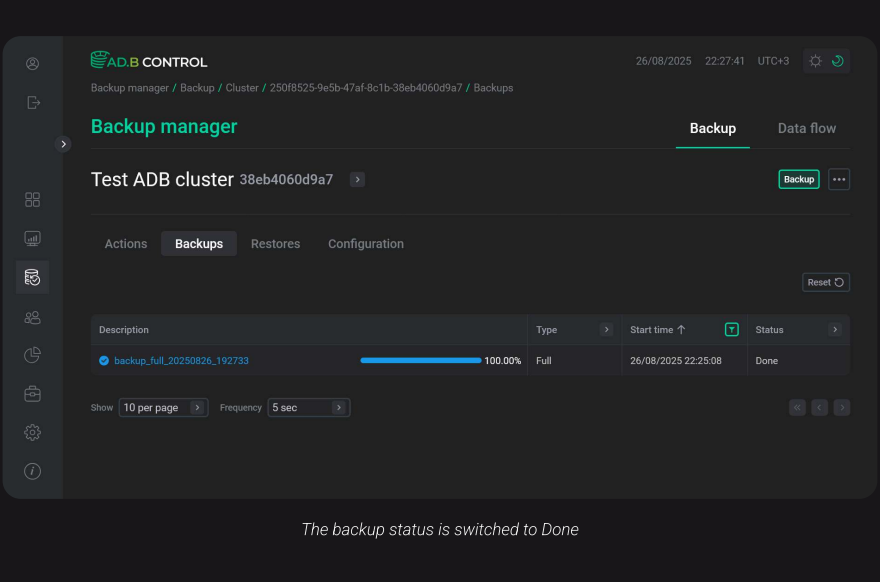
The Resume action form

2. Change the restore point name in the ***Restore point*** field if necessary.
3. Click ***Run***.
4. As a result, the ***Resume backup*** action runs. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
   The backup acquires the `Resume` status. You can check it in the list of backups on the ***Backups*** tab.



The backup has the Resume status

5. As a successful result of all actions, the backup acquires the `Done` status.

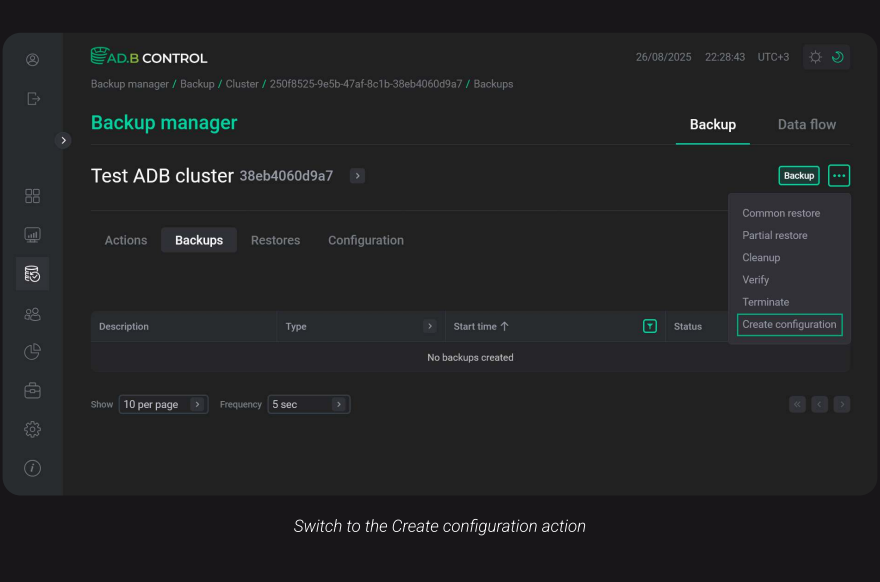


The backup status is switched to Done

Create configuration

> **NOTE**
>
> The ***Create configuration*** action is available starting with ADBM 2.3.2.

The ***Create configuration*** action redirects you to the ***Backup manager → Backup → Cluster → <Current ADB cluster name> → Configuration*** tab, where you can add a new backup configuration based on one of the previously created.



Switch to the Create configuration action

View backups in ADBM

Daria Barysheva

Contents

To Table of Contents

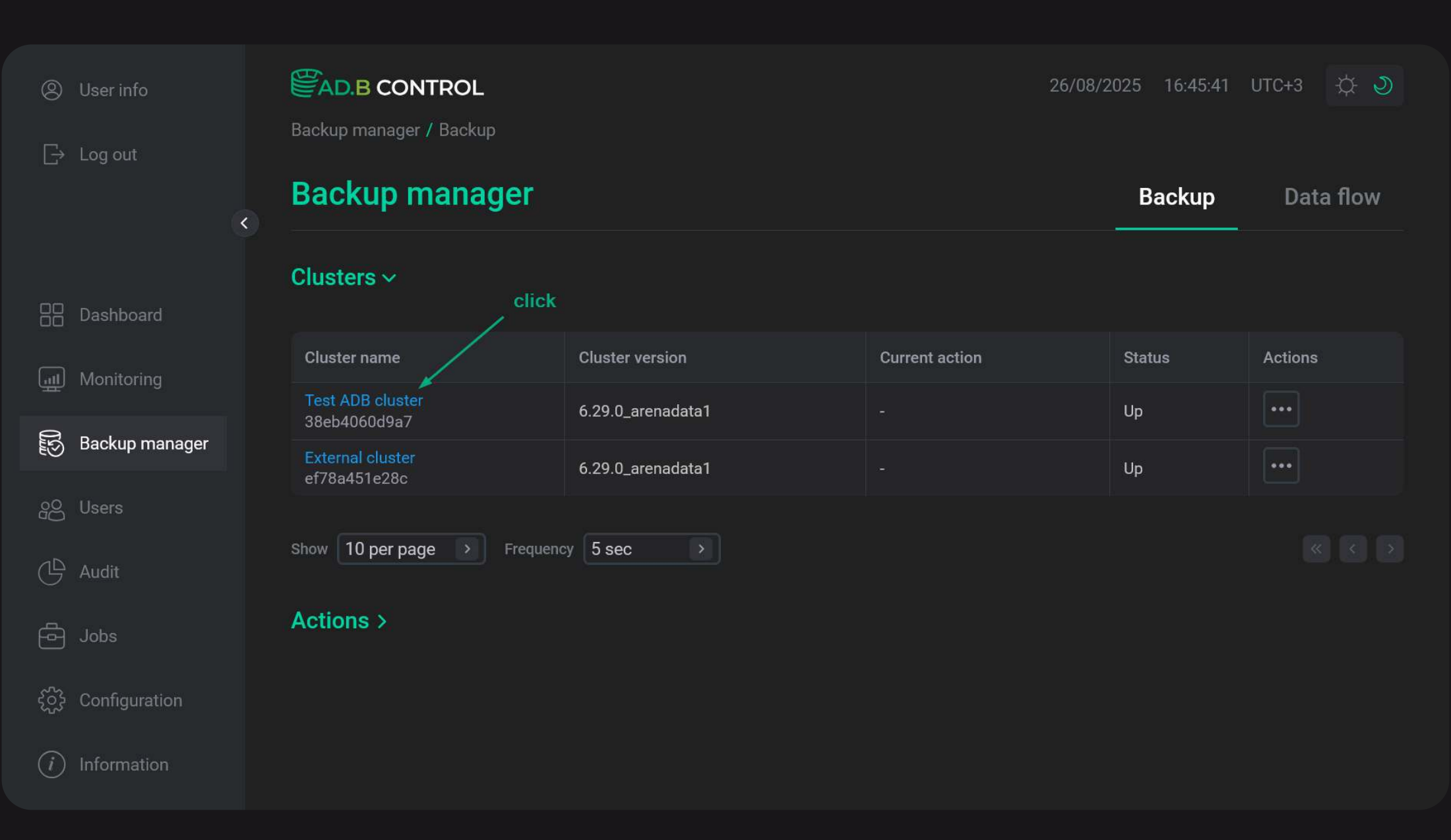
List of backups

Backup details

List of backups

Regardless of how backups are created (manually or on a schedule), all of them are displayed on the ***Backups*** tab. To open this tab, follow the steps:

1. Open the Backup manager page via the ADB Control web interface.
2. In the table that is located in the ***Clusters*** section of the opened page, click the ADB cluster name.

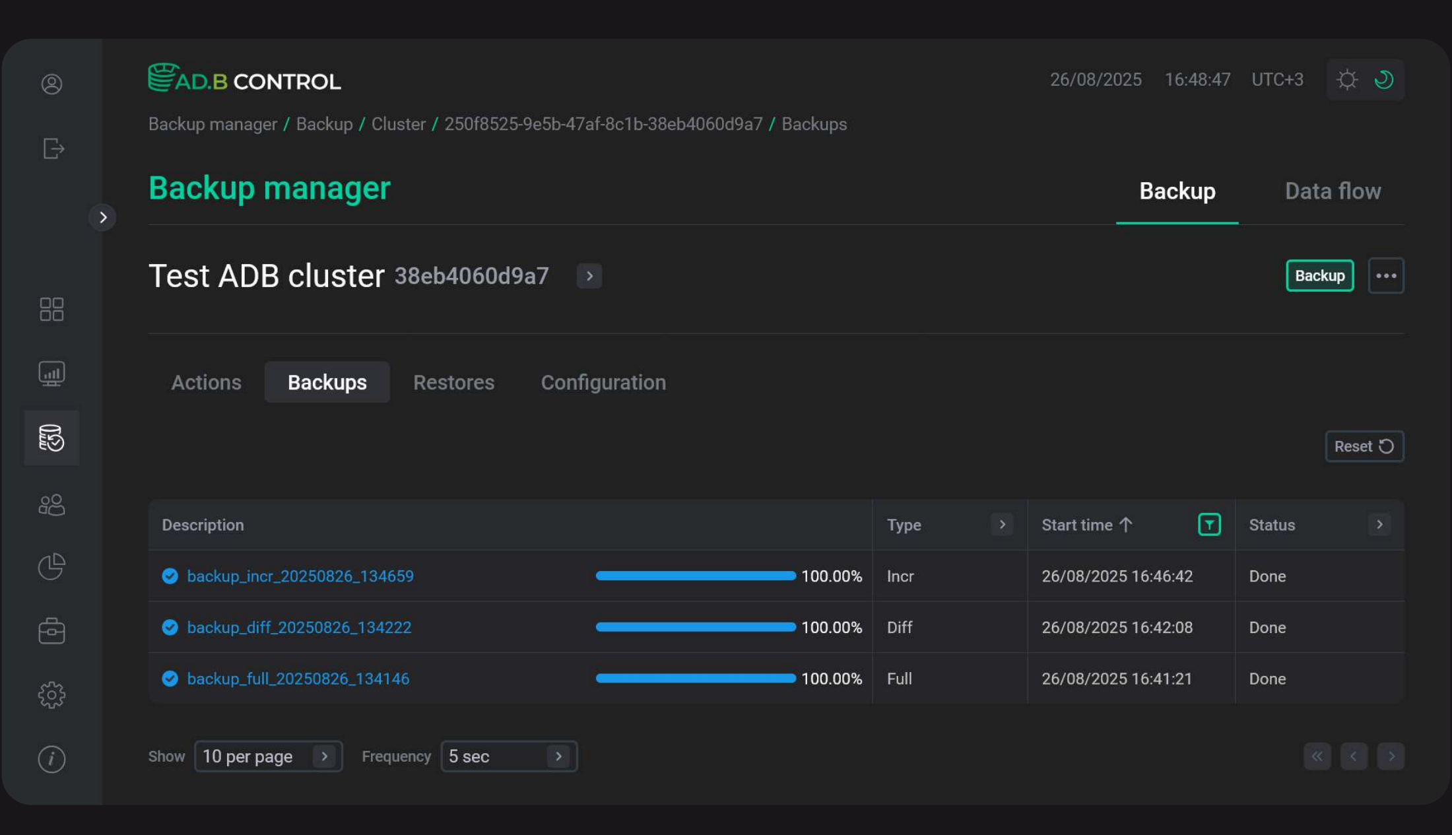


User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/08/2025 16:45:41 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
click
Show 10 per page
Frequency 5 sec
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Select the ADB cluster

3. Select the ***Backups*** tab.



AD.B CONTROL
26/08/2025 16:48:47 UTC+3
Backup manager / Backup / Cluster / 250f8525-9e5b-47af-8c1b-38eb4060d9a7 / Backups
Backup manager
Backup
Data flow
Test ADB cluster 38eb4060d9a7
Backup
Actions
Backups
Restores
Configuration
Reset
Show 10 per page
Frequency 5 sec

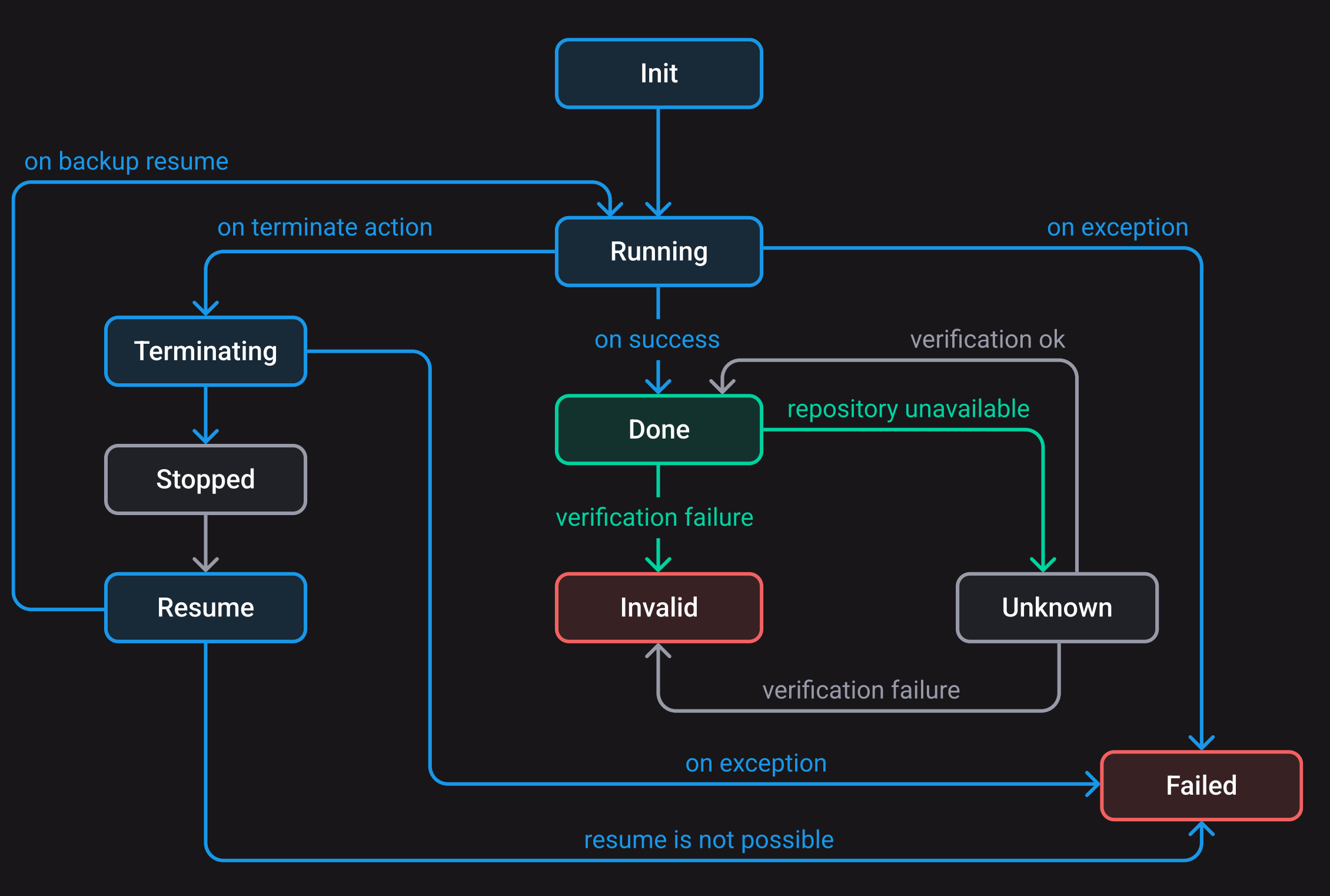
Description		Type	Start time	Status
backup_incr_20250826_134659	100.00%	Incr	26/08/2025 16:46:42	Done
backup_diff_20250826_134222	100.00%	Diff	26/08/2025 16:42:08	Done
backup_full_20250826_134146	100.00%	Full	26/08/2025 16:41:21	Done

The Backups tab

On the page that opens, there is a table with the following information on backups.

Backup list fields

Field	Description
Description	A backup name. Filled in automatically when performing a backup according to the `backup_<type>_<start_date>_<start_time>` template, where: - `<type>` — backup type (see ***Type*** below); - `<start_date>` — backup start date in the `YYYYMMDD` format; - `<start_time>` — backup start time in the `HHmmss` format. Starting with ADBM 2.4.0, there is a progress bar next to a backup name, which displays the backup execution progress (as a percentage of the estimated duration for all cluster segments). Note that backup progress is calculated only for the physical cluster reservation and does not include other actions
Type	The initial backup type (which was selected when creating a backup). Possible values: - `Full` — full backup; - `Diff` — differential backup; - `Incr` — incremental backup.
Start time	A backup start timestamp in the `DD/MM/YYYY HH:mm:ss` format
Status	A backup status. Possible values: - `Init` — the backup is being initialized. - `Running` — the backup is running. - `Terminating` — the backup is being terminated. - `Invalid` — after verification, the backup is considered incorrect. Such backups will be automatically dropped when the next cleanup procedure starts. - `Stopped` — the backup launch is stopped (after successful termination). - `Resume` — the backup is being resumed. - `Failed` — the backup launch is failed. - `Done` — the backup launch is successful. - `Unknown` — a repository that is specified in the backup configuration is inaccessible. For more information on possible transitions between statuses, see the diagram below



Init
on backup resume
on terminate action
Running
on exception
Terminating
on success
verification ok
Done
repository unavailable
Stopped
verification failure
Resume
Invalid
Unknown
verification failure
on exception
Failed
resume is not possible

Backup statuses

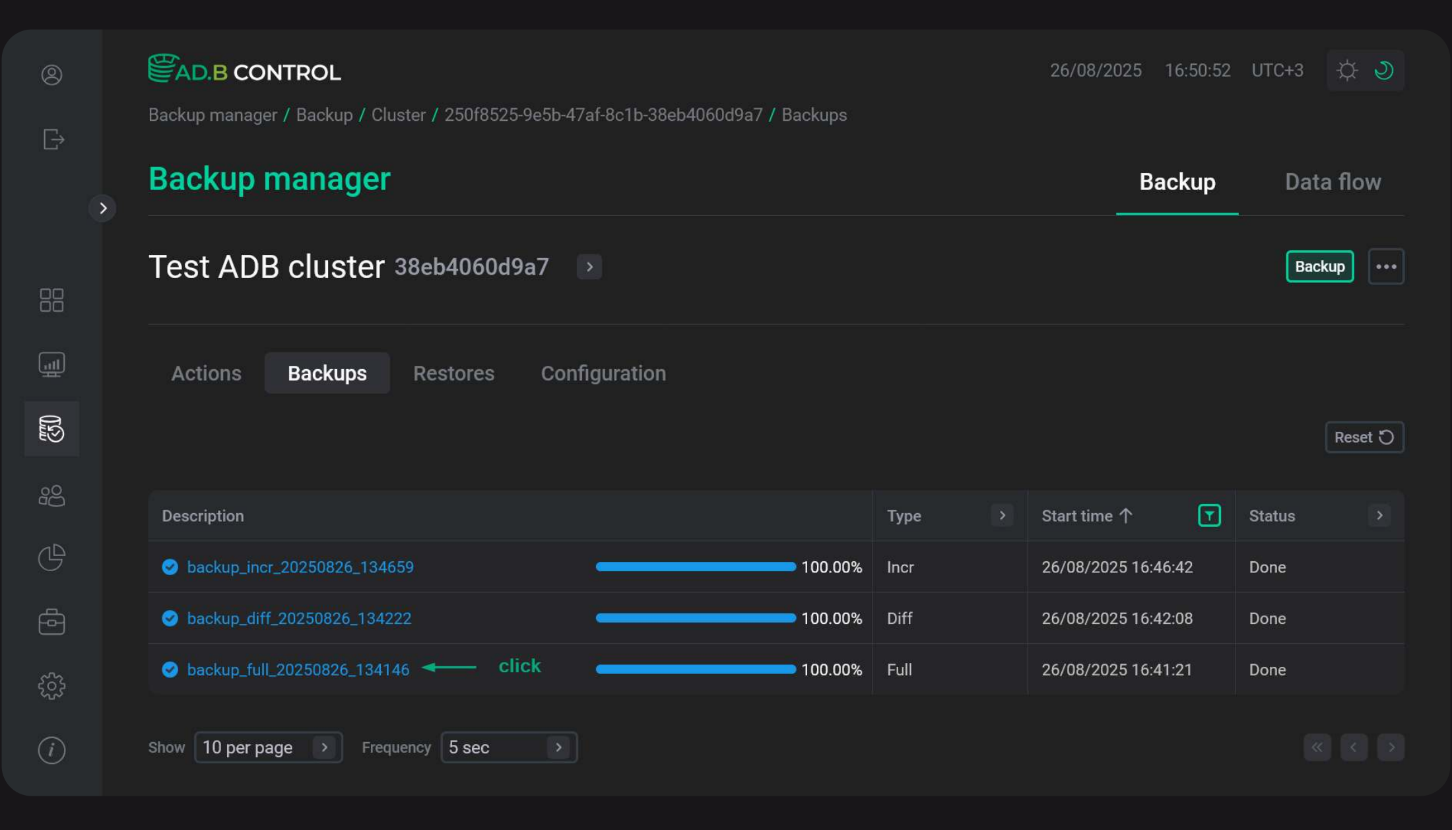
In the column headers of the table with a list of backups, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Type***), you can select a value from the drop-down list. For columns that show date and time (e.g. ***Start time***), the time range can be selected from the calendar.

The icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

> **IMPORTANT**
>
> By default, the table on the ***Backups*** tabs shows the current-day backups. To view all backups, use the time filters.

Backup details

To view the backup details, click the backup description (***Description***) in the table on the ***Backups*** tab.



AD.B CONTROL
26/08/2025 16:50:52 UTC+3
Backup manager / Backup / Cluster / 250f8525-9e5b-47af-8c1b-38eb4060d9a7 / Backups
Backup manager
Backup
Data flow
Test ADB cluster 38eb4060d9a7
Backup
Actions
Backups
Restores
Configuration
Reset
click
Show 10 per page
Frequency 5 sec

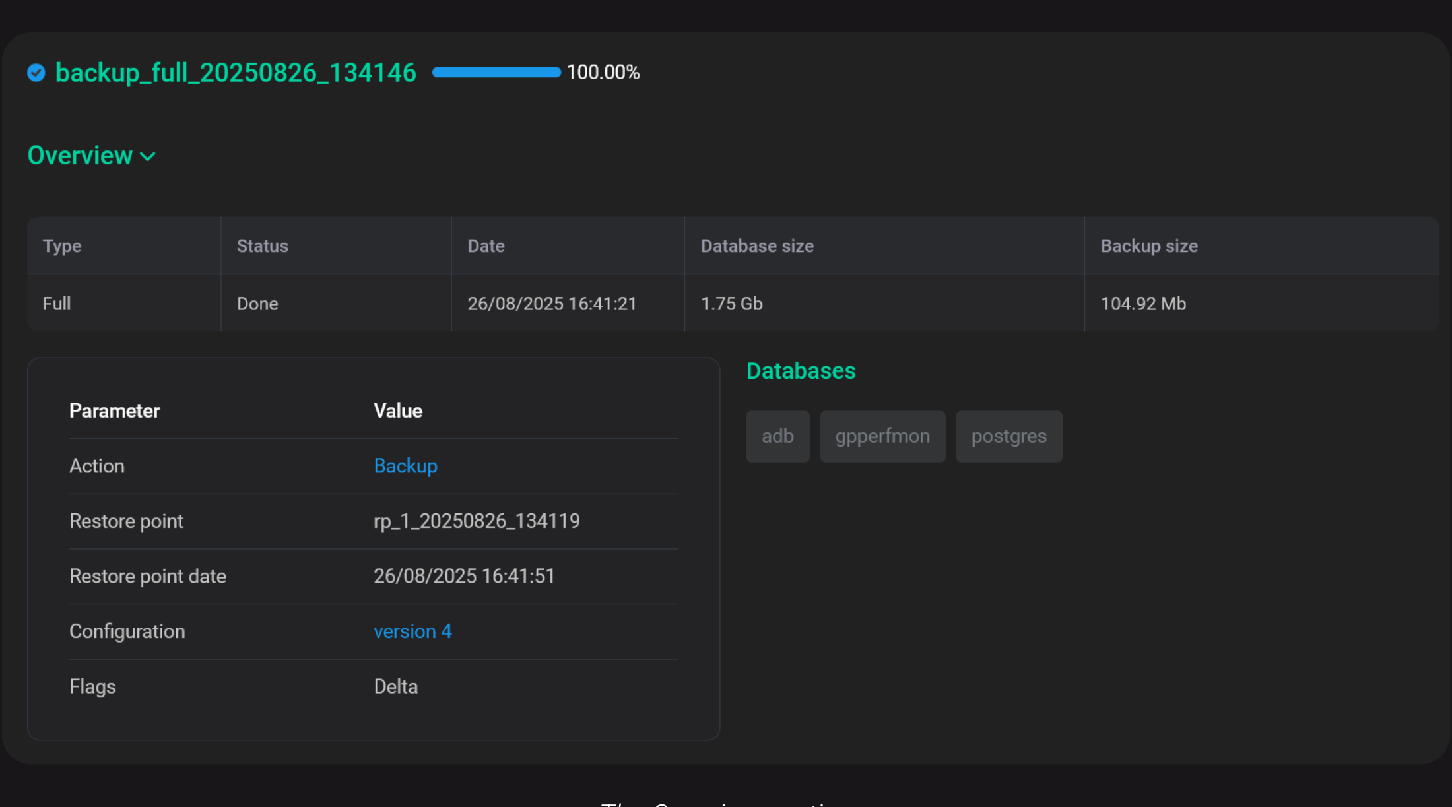
Description		Type	Start time	Status
backup_incr_20250826_134659	100.00%	Incr	26/08/2025 16:46:42	Done
backup_diff_20250826_134222	100.00%	Diff	26/08/2025 16:42:08	Done
backup_full_20250826_134146	100.00%	Full	26/08/2025 16:41:21	Done

Select a backup

The next page contains the following sections:

- ***Overview***
- ***Topology***
- ***Statuses***

The ***Overview*** section shows the general backup information. At the top of the section, you can see the backup name. Starting with ADBM 2.4.0, there is a progress bar next to the backup name, which displays the backup execution progress (as a percentage of the estimated duration for all cluster segments).



backup_full_20250826_134146 100.00%
Overview
Databases
adb
gpperfmon
postgres

Type	Status	Date	Database size	Backup size
Full	Done	26/08/2025 16:41:21	1.75 Gb	104.92 Mb

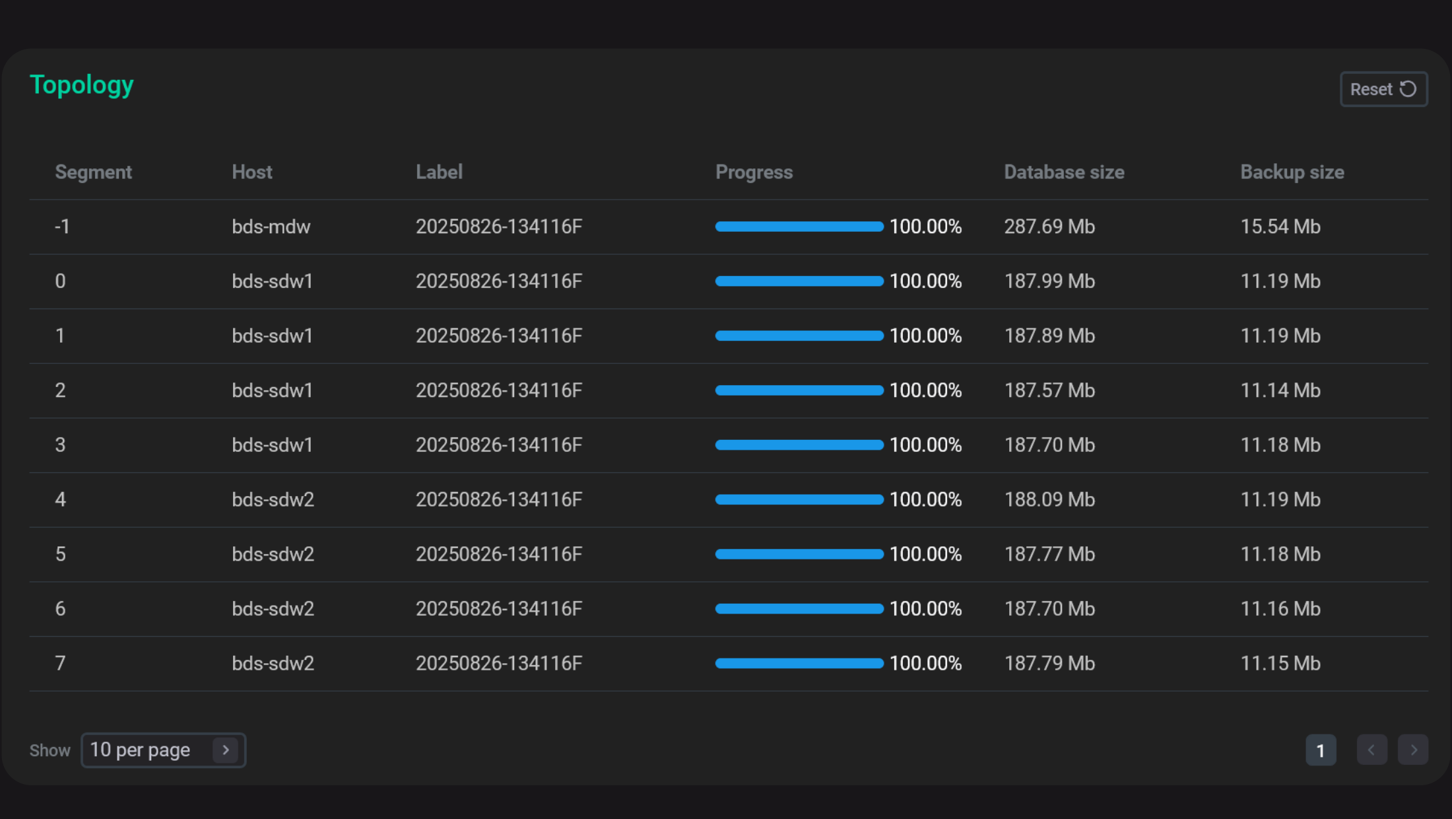
Parameter	Value
Action	Backup
Restore point	rp_1_20250826_134119
Restore point date	26/08/2025 16:41:51
Configuration	version 4
Flags	Delta

The Overview section

Fields of the Overview section

Field	Description
Type	A backup type. See ***Type*** above
Status	A backup status. See ***Status*** above
Date	A backup start timestamp in the `DD/MM/YYYY HH:mm:ss` format
Database size	The total volume of data files that have been backed up
Backup size	The total backup volume
Action	A name of the action within which the current backup was launched. You can click the name to view the detailed action information
Restore point	A name of the restore point created after the backup was completed
Restore point date	The restore point creation timestamp in the `DD/MM/YYYY HH:mm:ss` format
Configuration	A backup configuration version. To see the configuration details, click the version number
Flags	The flags that are selected for the backup operation (see ***Delta*** in the Backup description)
Databases	A list of databases that were in the DBMS at the time of the backup generation

The ***Topology*** section contains the backup topology information.



Topology
Reset
Show 10 per page
1

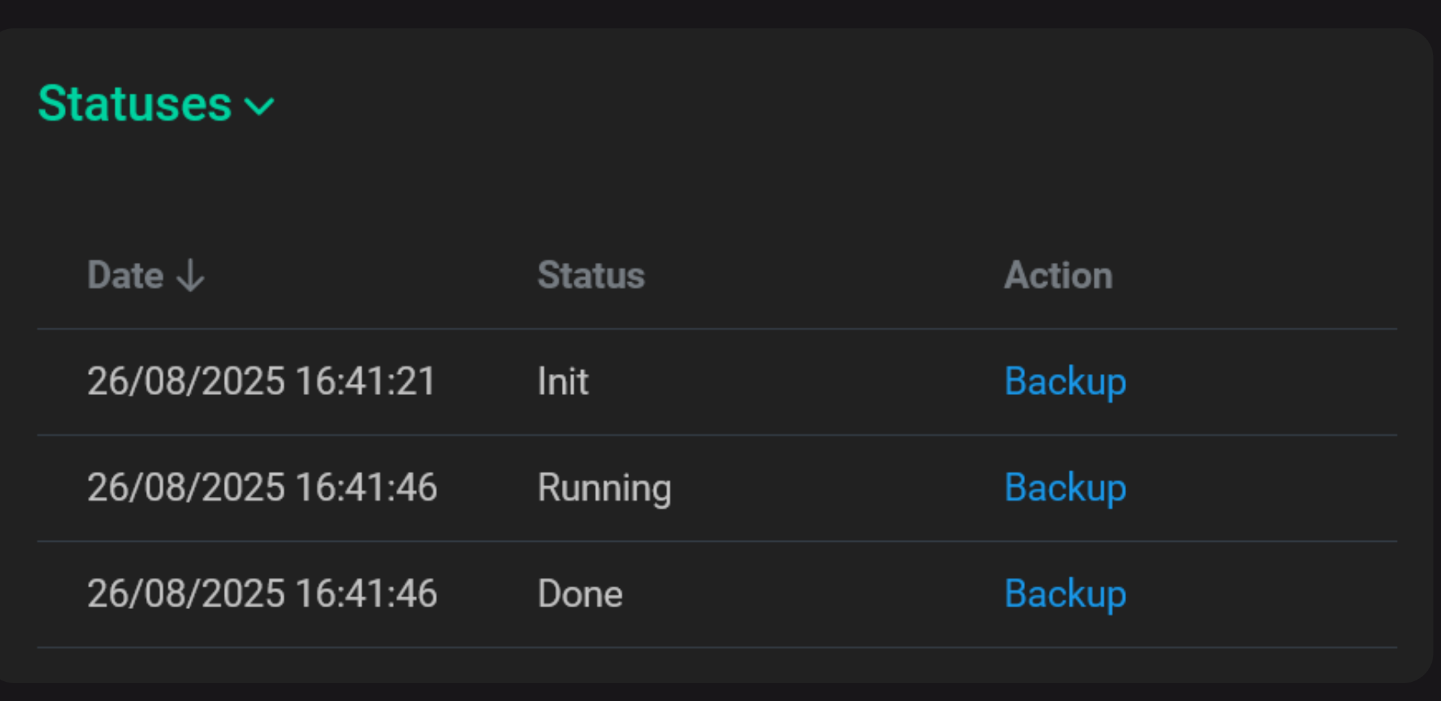
Segment	Host	Label	Progress	Database size	Backup size
-1	bds-mdw	20250826-134116F	100.00%	287.69 Mb	15.54 Mb
0	bds-sdw1	20250826-134116F	100.00%	187.99 Mb	11.19 Mb
1	bds-sdw1	20250826-134116F	100.00%	187.89 Mb	11.19 Mb
2	bds-sdw1	20250826-134116F	100.00%	187.57 Mb	11.14 Mb
3	bds-sdw1	20250826-134116F	100.00%	187.70 Mb	11.18 Mb
4	bds-sdw2	20250826-134116F	100.00%	188.09 Mb	11.19 Mb
5	bds-sdw2	20250826-134116F	100.00%	187.77 Mb	11.18 Mb
6	bds-sdw2	20250826-134116F	100.00%	187.70 Mb	11.16 Mb
7	bds-sdw2	20250826-134116F	100.00%	187.79 Mb	11.15 Mb

The Topology section

Fields of the Topology section

Field	Description
Segment	The content identifier of a segment (or coordinator) instance. Corresponds to `gp_segment_configuration.content`
Host	A host name
Label	A backup name according to the internal logic of `pgbackrest`
Progress	Displays the backup execution progress for the current segment (as a percentage of the estimated duration). Available starting with ADBM 2.4.0
Database size	The full volume of uncompressed database data on a segment
Backup size	The full volume of compressed database data on a segment

The ***Statuses*** section displays a history of changing the backup status.



Date ↓	Status	Action
26/08/2025 16:41:21	Init	Backup
26/08/2025 16:41:46	Running	Backup
26/08/2025 16:41:46	Done	Backup

The Statuses section

Fields of the Statuses section

Field	Description
Date	A timestamp of the backup status change in the `DD/MM/YYYY HH:mm:ss` format
Status	A backup status. See ***Status*** above
Action	A name of the action during which the backup status was changed. To view the action details, click the name

View restores in ADBM

Daria Barysheva

Contents

To Table of Contents

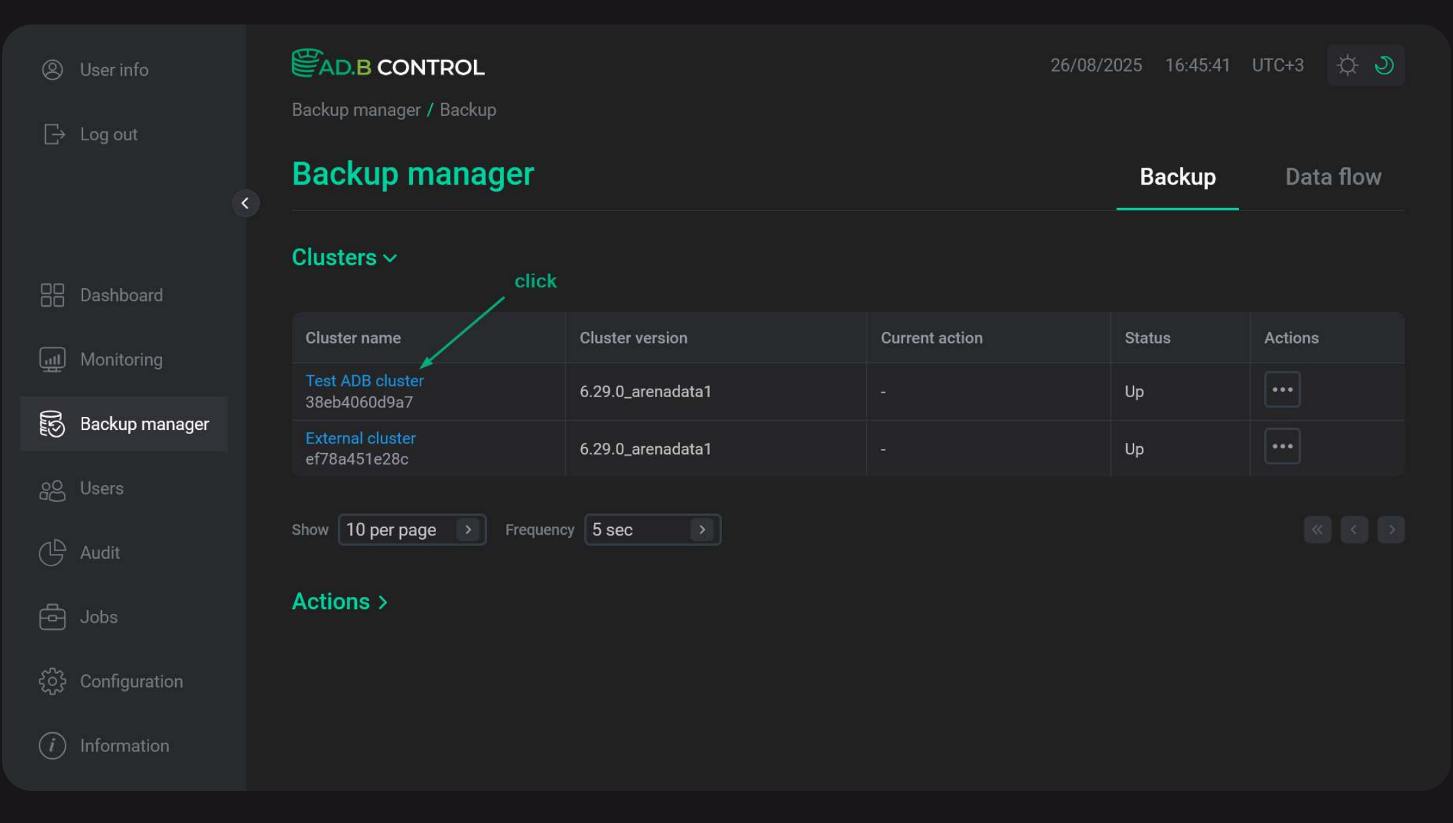
List of restores

Restore details

List of restores

All restore operations that are launched by users in ADBM (via the Common restore, Partial restore actions or Disaster recovery) are displayed on the ***Restores*** tab. To open this tab, follow the steps:

1. Open the Backup manager page via the ADB Control web interface.
2. In the table that is located in the ***Clusters*** section of the opened page, click the ADB cluster name.

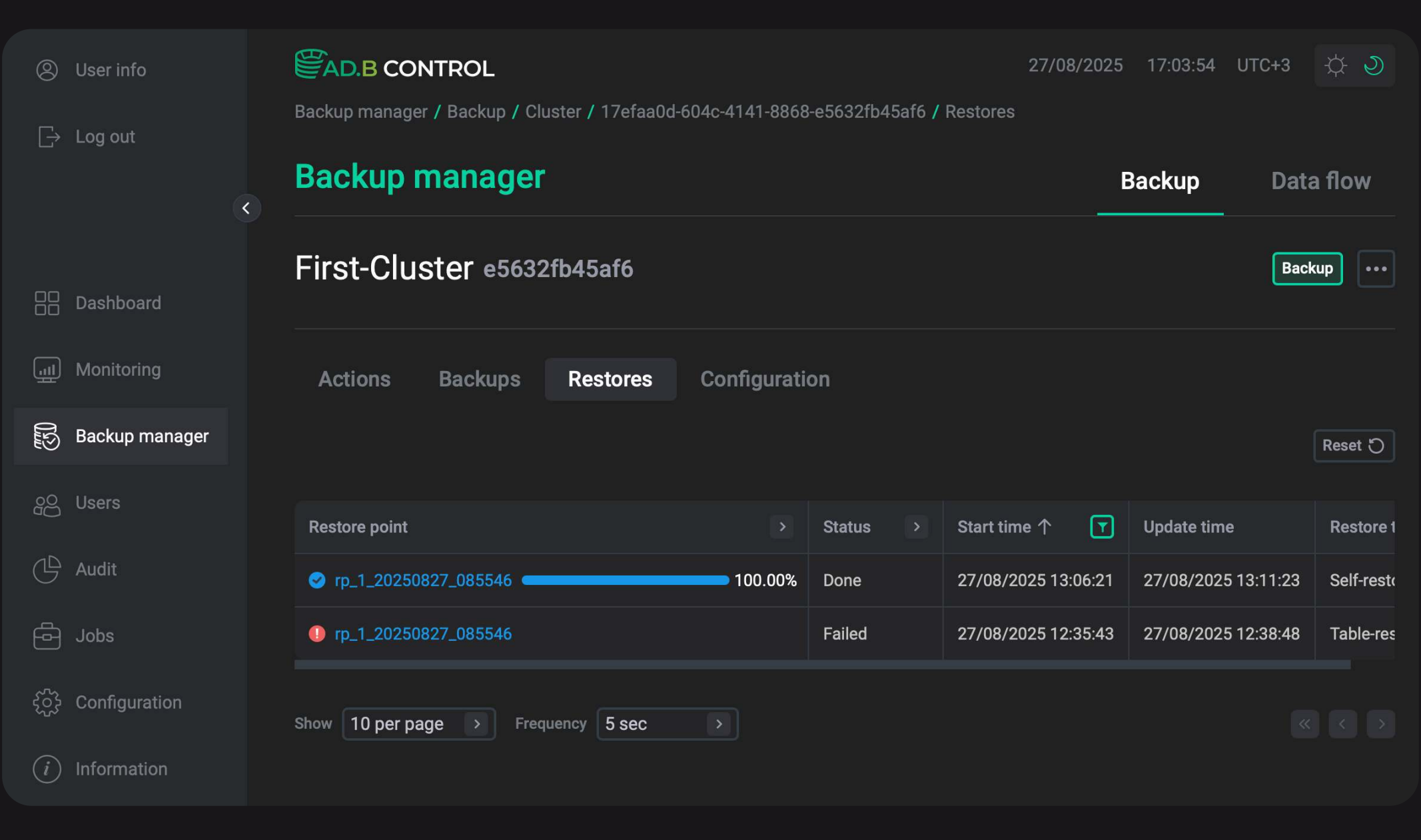


User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
26/08/2025 16:45:41 UTC+3
Backup manager / Backup
Backup manager
Backup
Data flow
Clusters
click
Show 10 per page
Frequency 5 sec
Actions

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Select the ADB cluster

3. Select the ***Restores*** tab.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
27/08/2025 17:03:54 UTC+3
Backup manager / Backup / Cluster / 17efaa0d-604c-4141-8868-e5632fb45af6 / Restores
Backup manager
Backup
Data flow
First-Cluster e5632fb45af6
Backup
Actions
Backups
Restores
Configuration
Reset
Show 10 per page
Frequency 5 sec

Restore point	Status	Start time	Update time	Restore t
rp_1_20250827_085546 100.00%	Done	27/08/2025 13:06:21	27/08/2025 13:11:23	Self-resto
rp_1_20250827_085546	Failed	27/08/2025 12:35:43	27/08/2025 12:38:48	Table-res

The Restores tab

On the page that opens, there is a table with the following information on the data restores.

Restore list fields

Field	Description
Restore point	A restore point that was selected to restore data. Starting with ADBM 2.5.0, there is a progress bar next to a restore point name, which displays the restore execution progress (as a percentage of the estimated duration for all cluster segments). This indicator is shown only for data restores with the `Self-restore` type
Status	A restore status. Possible values: ■ `Init` — the restore is being initialized; ■ `Running` — the restore is running; ■ `Stopped` — the restore is stopped; ■ `Failed` — the restore launch is failed; ■ `Done` — the restore launch is successful.
Start time	A start timestamp of the restore operation in the `DD/MM/YYYY HH:mm:ss` format
Update time	A timestamp of the last restore status update in the `DD/MM/YYYY HH:mm:ss` format
Restore type	A data restore type. Possible values: ■ `Self-restore` — a standard data restore operation that is launched via the Common restore action; ■ `Table-restore` — a table-level data restore operation that is launched via the Partial restore action; ■ `Copy` — a data restore operation that is launched during copying data between two clusters as a part of Disaster recovery with the `Copy` type (cold standby). ■ `Streaming` — a data restore operation that is launched during copying data between two clusters as a part of Disaster recovery with the `Streaming` type (warm standby).

In the column headers of the table with a list of restore operations, there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Restore point***), the search value should be entered. For columns that show date and time (e.g. ***Start time***), the time range can be selected from the calendar.

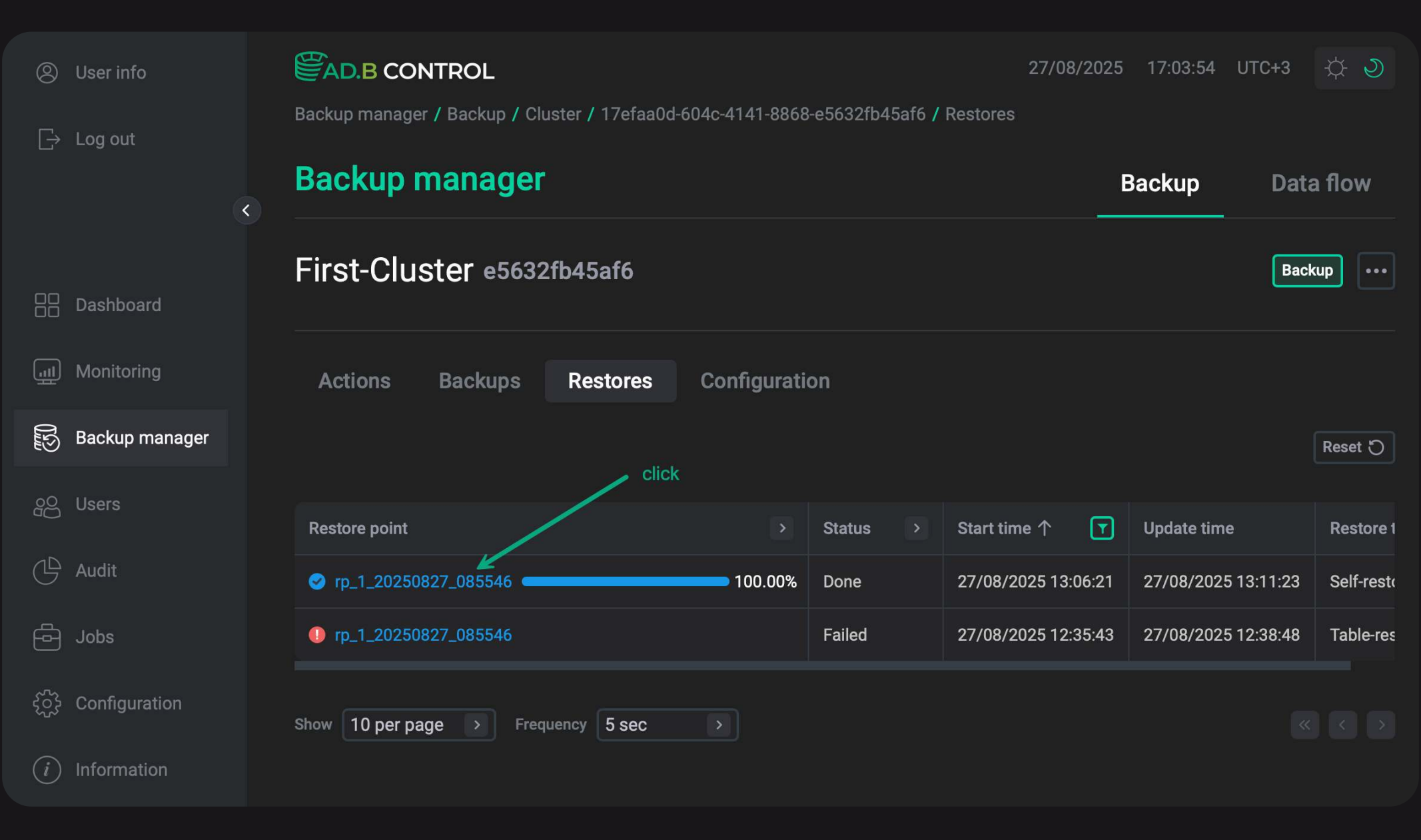
The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

IMPORTANT

By default, the table on the ***Restores*** tab shows the current-day restore operations. To view all restores, use the time filters.

Restore details

To view the restore details, click the restore point name (***Restore point***) in the table on the ***Restores*** tab.



User info
Log out
Dashboard
Monitoring
Backup manager
Users
Audit
Jobs
Configuration
Information
AD.B CONTROL
27/08/2025 17:03:54 UTC+3
Backup manager / Backup / Cluster / 17efaa0d-604c-4141-8868-e5632fb45af6 / Restores
Backup manager
Backup
Data flow
First-Cluster e5632fb45af6
Backup
Actions
Backups
Restores
Configuration
Reset
click
Show 10 per page
Frequency 5 sec

Restore point	Status	Start time	Update time	Restore t
rp_1_20250827_085546 100.00%	Done	27/08/2025 13:06:21	27/08/2025 13:11:23	Self-resto
rp_1_20250827_085546	Failed	27/08/2025 12:35:43	27/08/2025 12:38:48	Table-res

Select a restore operation

The next page displays the following information:

- The table with general restore information.
- The ***Parameters*** tab with additional information on parameters.
- The ***Tables*** tab with information on the restored tables.
- The ***Topology*** section with information on the cluster topology information.

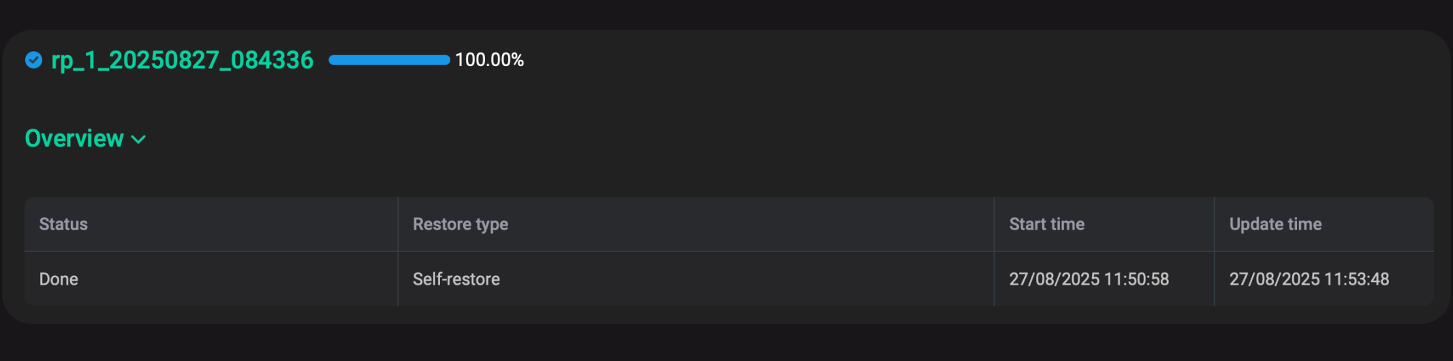
NOTE

- The ***Tables*** tab is displayed only for the Partial restore actions starting with ADBM 2.2.2.
- The ***Topology*** section is displayed starting with ADBM 2.5.0.

In the table, there are the following fields.

Fields of the table

Field	Description
Status	A restore status. See ***Status*** above
Restore type	A data restore type. See ***Restore type*** above
Start time	A start timestamp of the restore operation in the `DD/MM/YYYY HH:mm:ss` format
Update time	A timestamp of the last restore status update in the `DD/MM/YYYY HH:mm:ss` format



rp_1_20250827_084336 100.00%
Overview

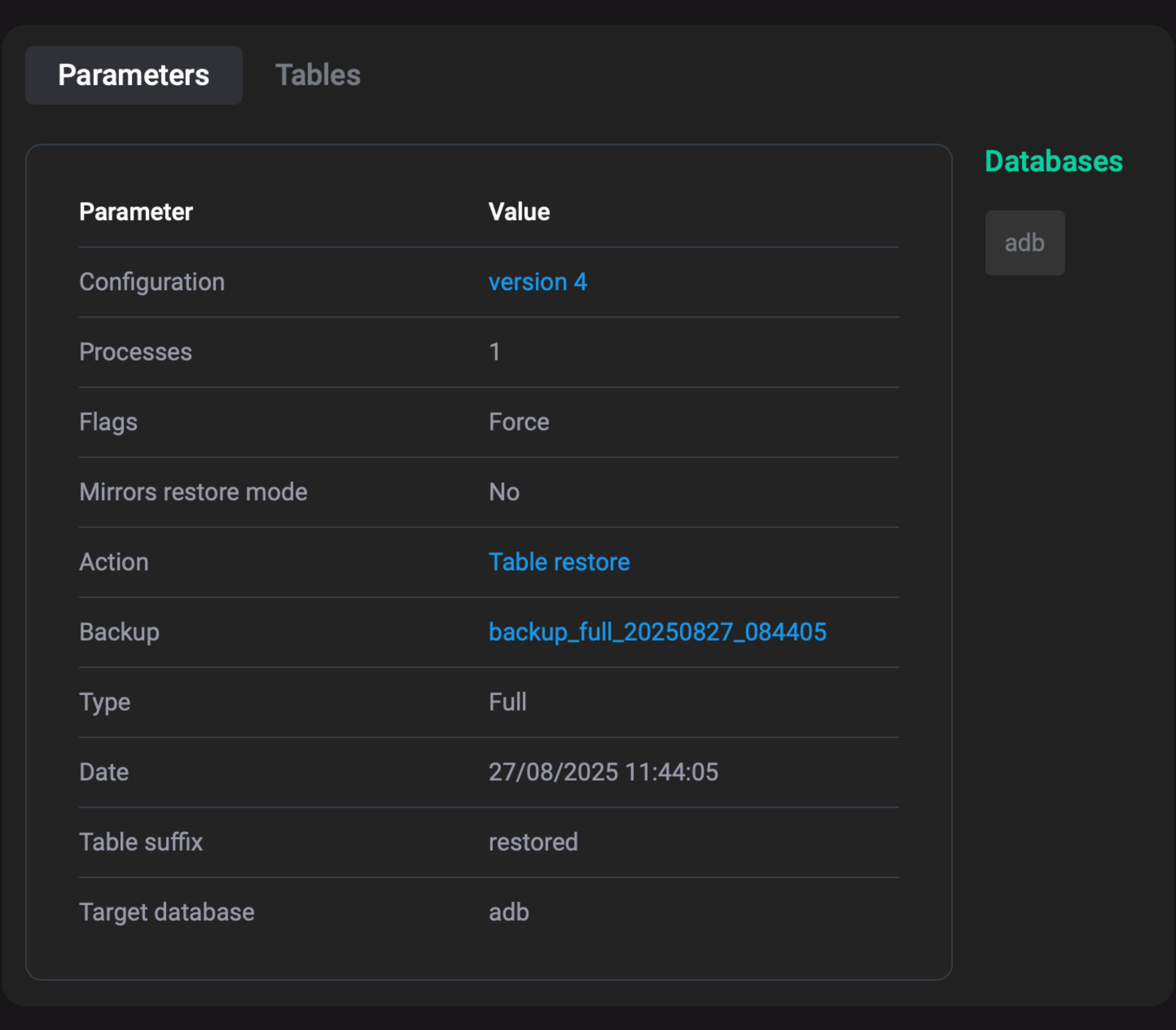
Status	Restore type	Start time	Update time
Done	Self-restore	27/08/2025 11:50:58	27/08/2025 11:53:48

Restore details: general table

On the ***Parameters*** tab, there are the following fields.

Fields of the Parameters tab

Field	Description
Configuration	A configuration version. To view the configuration details, click the version number
Processes	A maximum number of processes that is selected to restore one segment
Flags	The flags that are selected for the restore operation (see ***Delta***, ***Force***, ***Verify*** in the Common restore action and data flow launch descriptions)
Mirrors restore mode	The restore mode selected for mirrors during the data restore configuration (see ***Restore mirrors*** in the Common restore action and data flow launch descriptions)
Action	A name of the action within which the restore operation was launched. To view the action details, click the name
Backup	A name of the latest backup that was created before the selected restore point. To view the backup details, click the name
Type	A type of the latest backup that was created before the selected restore point
Date	A creation timestamp of the latest backup that was completed before the selected restore point
Table suffix	The suffix that is added to all table names in the target database during the restore process. Displayed only for the Partial restore actions starting with ADBM 2.2.2
Target database	The target database where tables are restored during the restore process. Displayed only for the Partial restore actions starting with ADBM 2.2.2
Databases	A list of source databases selected for the data restore (from their backups)



Parameters
Tables
Databases
adb

Parameter	Value
Configuration	version 4
Processes	1
Flags	Force
Mirrors restore mode	No
Action	Table restore
Backup	backup_full_20250827_084405
Type	Full
Date	27/08/2025 11:44:05
Table suffix	restored
Target database	adb

Restore details: Parameters

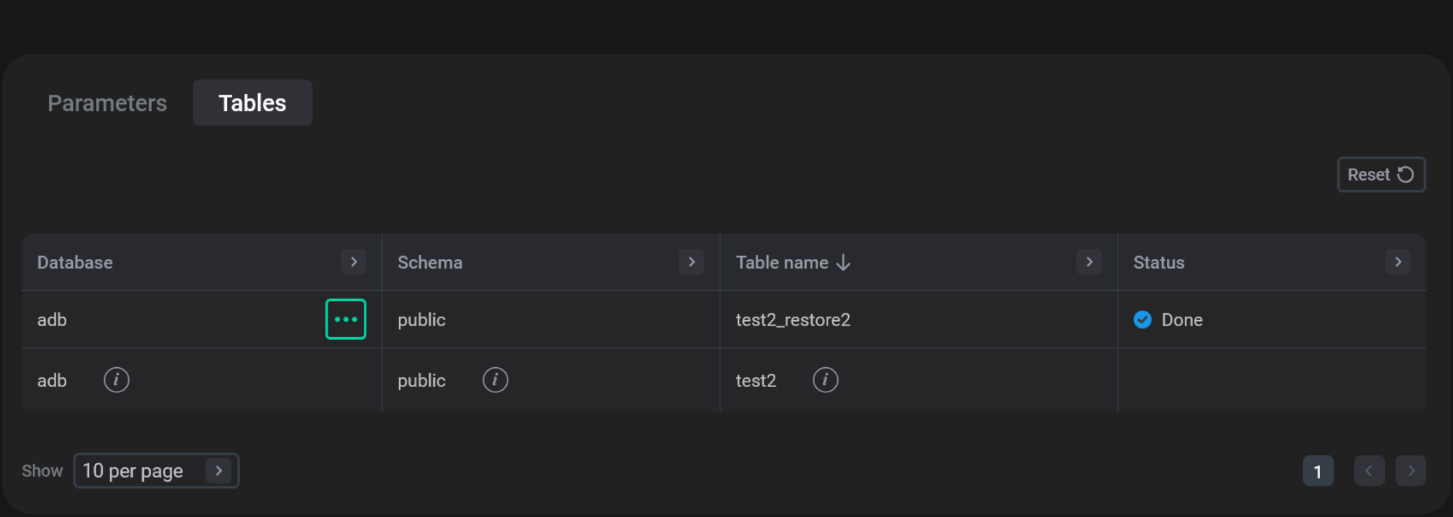
On the ***Tables*** tab, there are the following fields.

Fields of the Tables tab

Field	Description
Database	A name of the target database where the selected table is restored
Schema	A name of the target schema where the selected table is restored
Table name	A name of the restored table (after the restore operation)
Status	A table restore status. See ***Status*** above

TIP

If you click the ••• icon in the ***Database*** column, you can see the original names of the database, schema, and table from which the table data was restored — in the new row that appears.



Parameters
Tables
Reset
Show 10 per page
1

Database	Schema	Table name	Status
adb •••	public	test2_restore2	Done
adb ⓘ	public ⓘ	test2 ⓘ	

Restore details: Tables

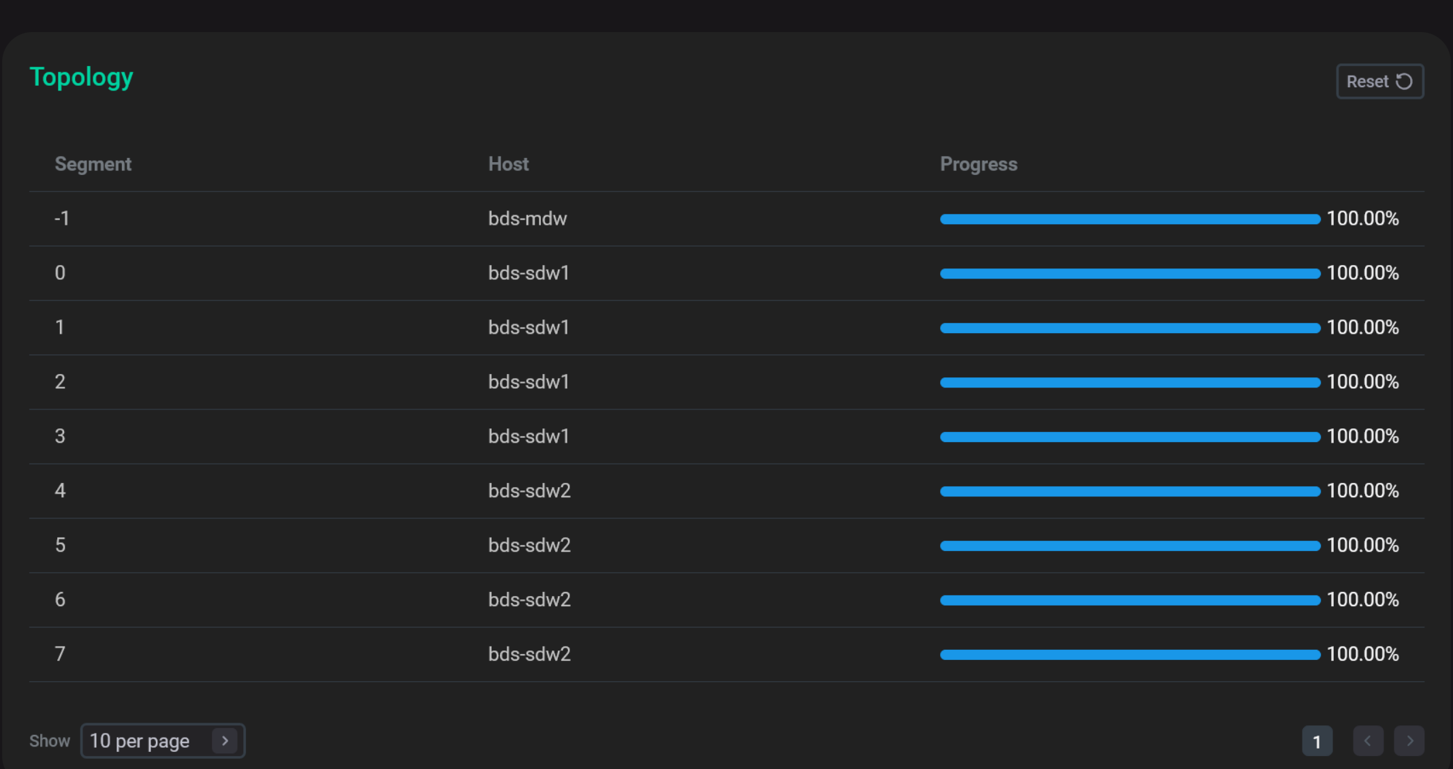
In the column headers of the table on the ***Tables*** tab there are filters that you can use to select specific data. To open a filter, click the > icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Database***), the search value should be entered.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

In the ***Topology*** section, there are the following fields.

Fields of the Topology section

Field	Description
Segment	The content identifier of a segment (or coordinator) instance. Corresponds to `gp_segment_configuration.content`
Host	A host name
Progress	Displays the restore execution progress for the current segment (as a percentage of the estimated duration). Available starting with ADBM 2.5.0 for data restores with the `Self-restore` type



Topology
Reset
Show 10 per page
1

Segment	Host	Progress
-1	bds-mdw	100.00%
0	bds-sdw1	100.00%
1	bds-sdw1	100.00%
2	bds-sdw1	100.00%
3	bds-sdw1	100.00%
4	bds-sdw2	100.00%
5	bds-sdw2	100.00%
6	bds-sdw2	100.00%
7	bds-sdw2	100.00%

Restore details: Topology

Disaster recovery

Daria Barysheva

Contents

To Table of Contents

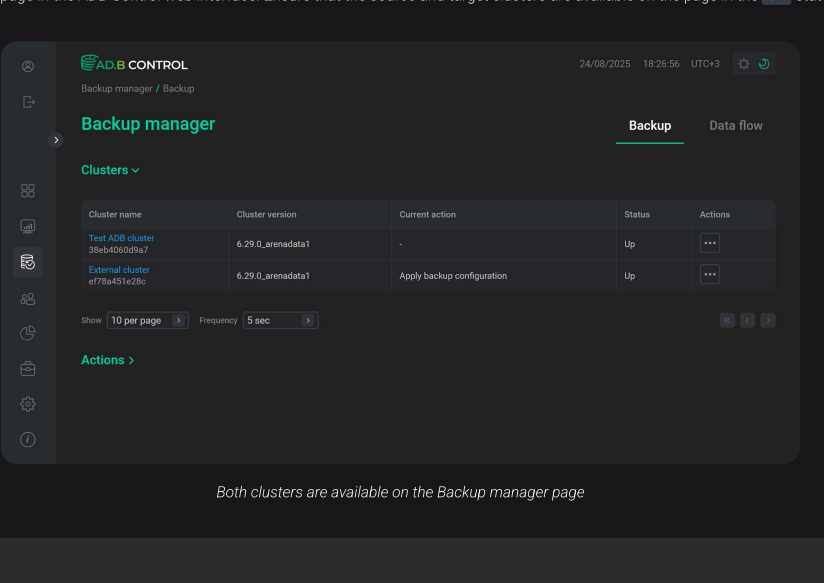
- Configure and run
- DR configuration management
  - View a list of configurations
  - Edit a configuration
  - Delete a configuration
- Data flow management
  - View active data flows
  - View finished data flows
  - View the data flow details
  - Terminate a data flow
  - Resume a data flow
  - Cancel a data flow

Configure and run

To configure disaster recovery (DR) in ADBM, follow the steps:

1. Using the ADB Control web interface, add a new cluster to which data from the source ADB cluster will be copied. Next, the added cluster will be called **target** or **DR cluster**. Note that both ADB Control and ADBM services should be added to the target cluster via ADCM.

2. Open the ***Backup manager*** page in the ADB Control web interface. Ensure that the source and target clusters are available on the page in the `UP` status.

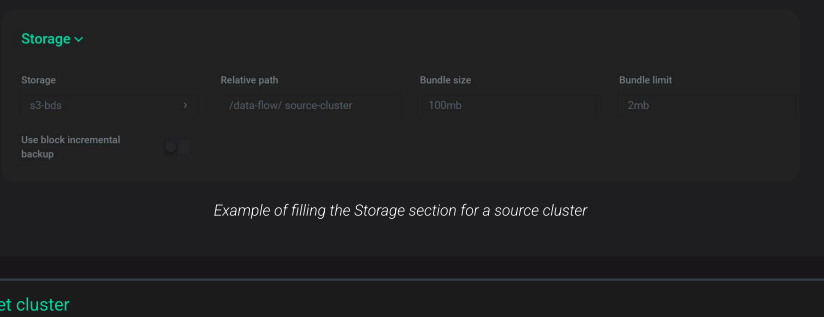


Both clusters are available on the Backup manager page

> CAUTION
>
> - To properly configure data copying, use two clusters with the same major version (the first two digits in the bundle number). For example, clusters with the `6.23.4` and `6.23.5` bundle versions can interact. But you cannot configure data copying between clusters with the `6.23.4` and `6.24.1` versions.
> - To restore data on a target ADB cluster from a backup created on another cluster, you need the same number of primary segments on both clusters. The rest of the cluster topology settings (such as mirroring, standby, number of host segments) do not affect the restore option.
> - The operating system versions and families should be compatible on the source and target cluster hosts.

3. Add backup configurations for both clusters. Note that the source ***Storage*** section (which is defined for a source cluster) should describe the storage that is available for both clusters (for example, S3).

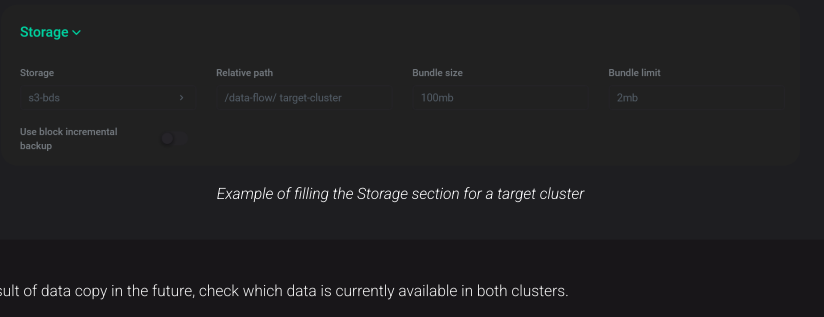
Storage section for a source cluster



Storage
Storage: s3-bds
Relative path: /data-flow/ source-cluster
Bundle size: 100mb
Bundle limit: 2mb
Use block incremental backup

Example of filling the Storage section for a source cluster

Storage section for a target cluster



Storage
Storage: s3-bds
Relative path: /data-flow/ target-cluster
Bundle size: 100mb
Bundle limit: 2mb
Use block incremental backup

Example of filling the Storage section for a target cluster

4. To verify the successful result of data copy in the future, check which data is currently available in both clusters.

Check a source cluster

- List tables of the `adb` database via the `\dt` `psql` command:

```
\dt
```

Result:

```
                  List of relations
 Schema |      Name       | Type  |  Owner  |       Storage
--------+-----------------+-------+---------+----------------------
 public | best_books      | table | gpadmin | heap
 public | book_type       | table | gpadmin | heap
 public | spatial_ref_sys | table | gpadmin | heap
 public | test            | table | gpadmin | append only columnar
 public | test2           | table | gpadmin | append only columnar
(5 rows)
```

- View the `best_books` table data:

```
SELECT * FROM best_books;
```

Result:

```
 id |   author    |         title         | public_year
----+-------------+-----------------------+-------------
  1 | Harper Lee  | To Kill a Mockingbird |        1960
(1 row)
```

Check a target cluster

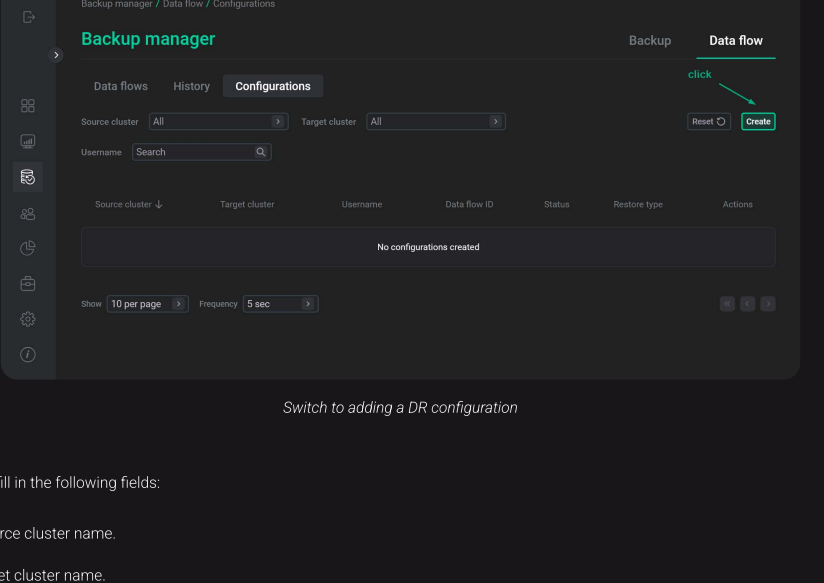
View tables of the `adb` database via the `\dt` `psql` command:

```
\dt
```

Result:

```
                 List of relations
 Schema |      Name       | Type  |  Owner  | Storage
--------+-----------------+-------+---------+---------
 public | spatial_ref_sys | table | gpadmin | heap
(1 row)
```

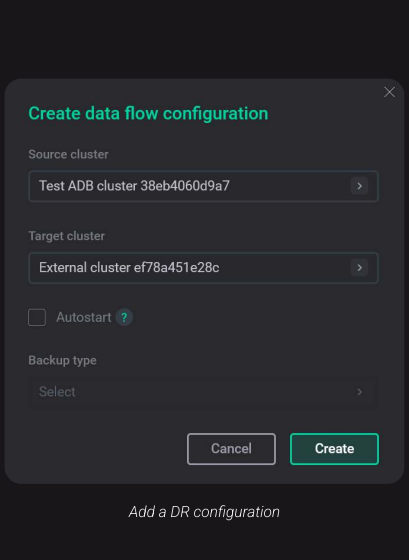
5. On the ***Data flow → Configurations*** tab of the ***Backup manager*** page, click ***Create*** to add a new DR configuration.



Switch to adding a DR configuration

6. In the window that opens, fill in the following fields:

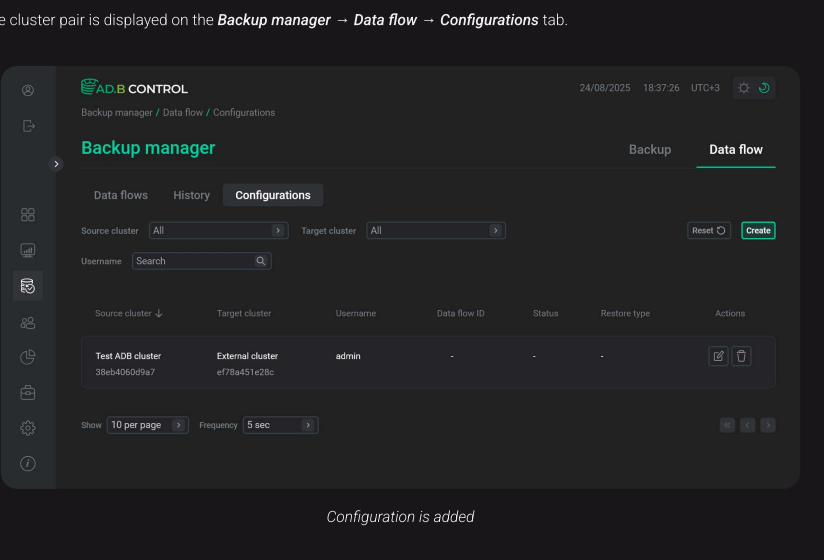
   - ***Source cluster*** — a source cluster name.
   - ***Target cluster*** — a target cluster name.
   - ***Autostart*** — a flag that indicates whether to create a data flow with the `Copy` restore type for a selected pair of clusters automatically each time a backup with the ***Backup type*** type is created in the source cluster.
   - ***Backup type*** — types of backups in the source cluster that will trigger copying data (data flow) to the target cluster. The field becomes available if the ***Autostart*** flag is set. Possible values:
     - `Full` — full backups;
     - `Diff` — differential backups;
     - `Incr` — incremental backups.



Create data flow configuration
Source cluster: Test ADB cluster 38eb4060d9a7
Target cluster: External cluster ef78a451e28c
Autostart
Backup type: Select

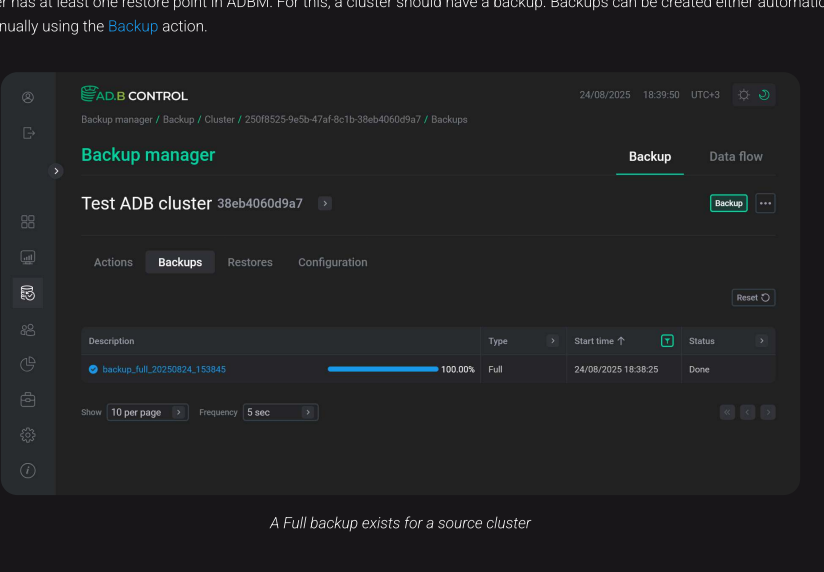
Add a DR configuration

7. Click ***Create***. As a result, the cluster pair is displayed on the ***Backup manager → Data flow → Configurations*** tab.



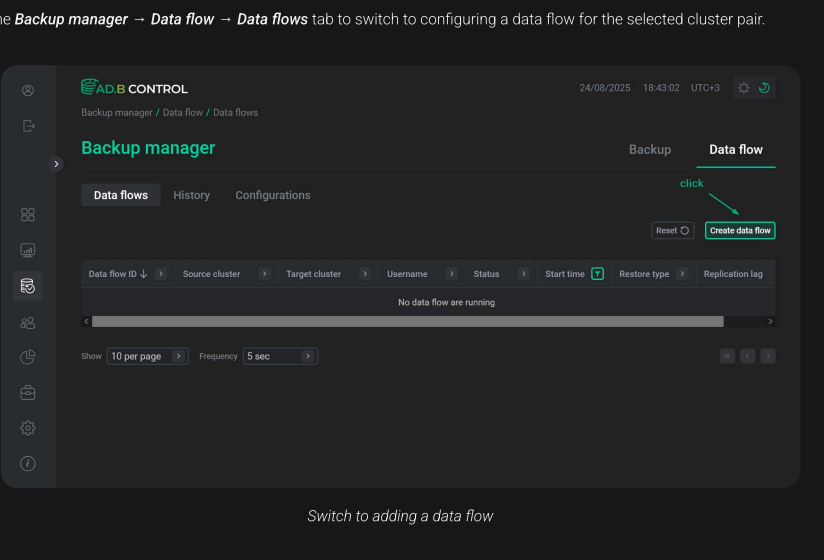
Configuration is added

8. Ensure that a source cluster has at least one restore point in ADBM. For this, a cluster should have a backup. Backups can be created either automatically according to the configured schedule or manually using the Backup action.



A Full backup exists for a source cluster

9. Click ***Create data flow*** on the ***Backup manager → Data flow → Data flows*** tab to switch to configuring a data flow for the selected cluster pair.



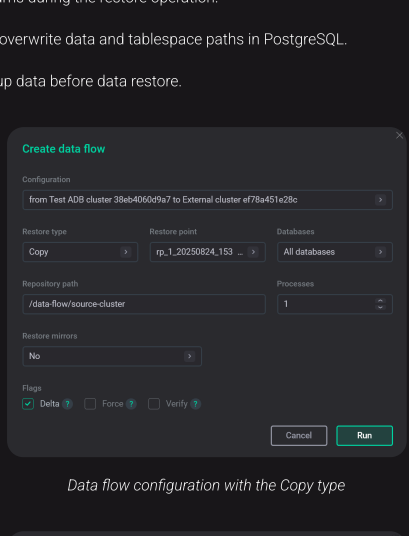
Switch to adding a data flow

10. In the window that opens, fill in the following fields:

    - ***Configuration*** — a configuration that you created earlier (which defines a pair of clusters).
    - ***Restore type*** — a restore type. Possible values:
      - `Copy` — cold standby. The target cluster data is recovered only once based on the selected restore point.
      - `Streaming` — warm standby. The target cluster data is constantly recovered based on the restore points created in the source cluster. After the replication process is terminated, the target cluster data will be actual at the moment of the latest restore point applied from the source cluster.

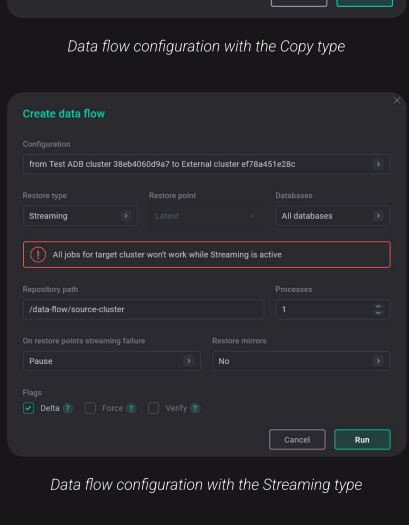
> NOTE
>
> Starting with ADBM 2.5.0, you can run a data flow with the `Copy` type, even if a target cluster is stopped (has the `Down` status). This allows you to restore data in a cluster if it has become inoperative and there are no backups for Common restore.

- ***Restore point*** — the restore point at the moment of which you want to restore the source cluster databases in the target cluster. If the `Streaming` restore type is selected, the latest restore point is always used, and the ***Restore point*** field is disabled.
- ***Databases*** — a list of the source cluster databases to restore in the target cluster.
- ***Repository path*** — a path to the repository that stores the source cluster backups. The default value is extracted from the current backup configuration.
- ***Processes*** — a maximum number of processes to restore one segment.
- ***On restore points streaming failure*** — when the `Streaming` restore type is selected, this field determines the behavior of the system in case of errors during the application of restore points to the target cluster (for example, damage of WAL archives, unplanned data flow interruption, storage unavailability, etc.). The field is available starting with ADBM 2.4.0. Possible values:
  - `Pause` — transfer the data flow to the `Paused` status for its temporary suspension. In this status, two actions become available for the data flow: Resume (to resume the data flow) and Cancel (to completely cancel the data flow). During the ***Resume*** action execution, the application of restore points to the target cluster resumes from the state when processing failed last time; and the data flow status returns to `Streaming`. The ***Cancel*** action, in turn, reverts the target cluster state to the previous restore point (preceding the one whose processing failed) and finishes the data flow (the data flow acquires the `Cancelled` status).
    This mode requires you to make manual corrections before resuming the data flow. It is suitable when you need to keep the target cluster state as close to the current state of the source cluster as possible and do not require unlocking the target cluster (for running DB requests) while making corrections.
  - `Bring online` — revert the target cluster state to the previous restore point (preceding the one whose processing failed), restore data, and finish the data flow (with unlocking the target cluster for running database queries and changing the data flow status to `Restored`).
- ***Restore mirrors*** — the drop-down list, which value indicates whether to restore data for mirrors and standby if they exist in the cluster:
  - `No` — do not restore mirrors.
  - `Parallel` — restore mirrors concurrently with primary segments.
  - `Sequential` — restore mirrors after all primary segments are restored.
- ***Flags***:
  - ***Delta*** — a flag that indicates the need to use checksums during the restore operation.
  - ***Force*** — a flag that indicates the need to completely overwrite data and tablespace paths in PostgreSQL.
  - ***Verify*** — a flag that indicates the need to verify backup data before data restore.



Create data flow
Configuration: from Test ADB cluster 38eb4060d9a7 to External cluster ef78a451e28c
Restore type: Copy
Restore point: rp_1_20250824_153 ...
Databases: All databases
Repository path: /data-flow/source-cluster
Processes: 1
Restore mirrors: No
Flags: ☑ Delta ☐ Force ☐ Verify

Data flow configuration with the Copy type



Create data flow
Configuration: from Test ADB cluster 38eb4060d9a7 to External cluster ef78a451e28c
Restore type: Streaming
Restore point: Latest
Databases: All databases
All jobs for target cluster won't work while Streaming is active
Repository path: /data-flow/source-cluster
Processes: 1
On restore points streaming failure: Pause
Restore mirrors: No
Flags: ☑ Delta ☐ Force ☐ Verify

Data flow configuration with the Streaming type

11. Click ***Run***. As a result, the ***Stream copy*** (for the `Copy` restore type) or ***Streaming cluster*** (for the `Streaming` restore type) action starts for the target cluster. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
The ***Streaming cluster*** action remains in the `Started` status until the data flow is terminated (see the note below). During that action, all restore points that are created in the source cluster will be applied to the target cluster.

12. As a successful result of all actions, the source cluster data is copied to the target cluster. To ensure this, you can run queries from step 4.

> IMPORTANT
>
> For data flows with the `Streaming` type:
>
> - Data restore completes only after terminating a data flow — manually or automatically. Auto termination can be initiated by one of the following events:
>   - Common restore of the source cluster.
>   - Running a new data flow where the source cluster is selected as a target.
>   - Creating a new backup configuration for the source cluster that changes the repository settings.
>
>   Only after successful termination, you can run queries in the target cluster.
> - The target cluster database will contain the source cluster data at the time of the last restore point that was applied during the streaming process.
> - Restore points are not created when other actions are performed on a cluster since operations on clusters are exclusive. This may lead to the increased streaming lag, for example, when a full backup is performed on a large cluster.

Check a target cluster

- List tables of the `adb` database via the `\dt` `psql` command:

```
\dt
```

Result:

```
                  List of relations
 Schema |      Name       | Type  |  Owner  |       Storage
--------+-----------------+-------+---------+----------------------
 public | best_books      | table | gpadmin | heap
 public | book_type       | table | gpadmin | heap
 public | spatial_ref_sys | table | gpadmin | heap
 public | test            | table | gpadmin | append only columnar
 public | test2           | table | gpadmin | append only columnar
(5 rows)
```

- View the `best_books` table data:

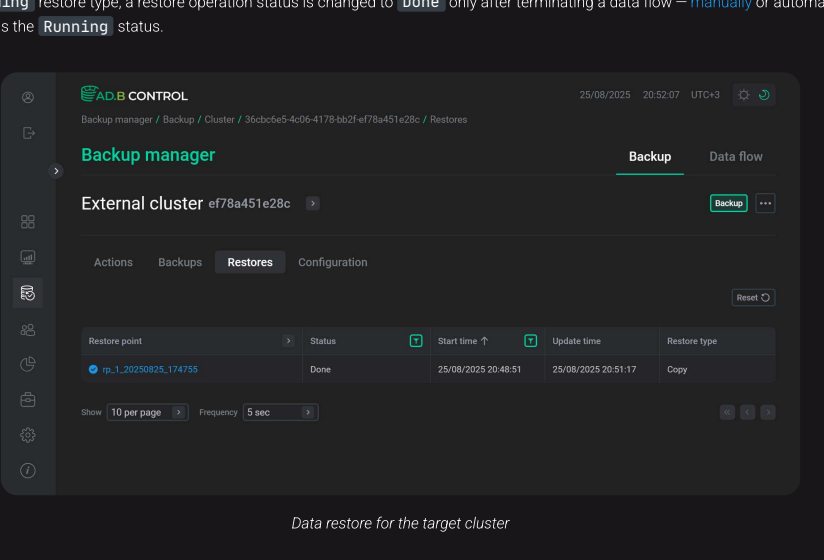
```
SELECT * FROM best_books;
```

Result:

```
 id |   author    |         title         | public_year
----+-------------+-----------------------+-------------
  1 | Harper Lee  | To Kill a Mockingbird |        1960
(1 row)
```

Note that during each data flow launch, the restore operation is created for the target cluster in ADBM. You can view it on the ***Backup manager → Backup → <target cluster name> → Restores*** tab as well as a standard data restore that is launched via the Common restore action. For restore operations made as a part of copying data between clusters, the ***Restore type*** column contains `Copy` or `Streaming` (depending on the selected restore type). For more information, see View restores in ADBM.

For data flows with the `Streaming` restore type, a restore operation status is changed to `Done` only after terminating a data flow — manually or automatically (see the note above). Until then, a restore operation has the `Running` status.



Data restore for the target cluster

Unchanging files

By default, the following files remain intact during a target cluster restore regardless of the data flow type (`Copy` or `Streaming`). Files can be set in the ADBM agent configuration. It is not recommended to remove files from the default list (as this may negatively affect the cluster availability after data restore), but you can add your own files to the list if required.

Coordinator

- *arenadata.pg_hba_custom*
- *gpperfmon/conf/gpperfmon.conf*
- *crt_from_adcm.crt*
- *key_from_adcm.key*
- *ca_cert_from_adcm.crt*
- *crl_from_adcm.crl*
- *pg_hba.conf*
- *pg_ident.conf*
- *postgresql.conf*
- *postgresql.auto.conf*
- *internal.auto.conf*

Primary segments

- *pg_hba.conf*
- *pg_ident.conf*
- *postgresql.conf*
- *postgresql.auto.conf*
- *internal.auto.conf*

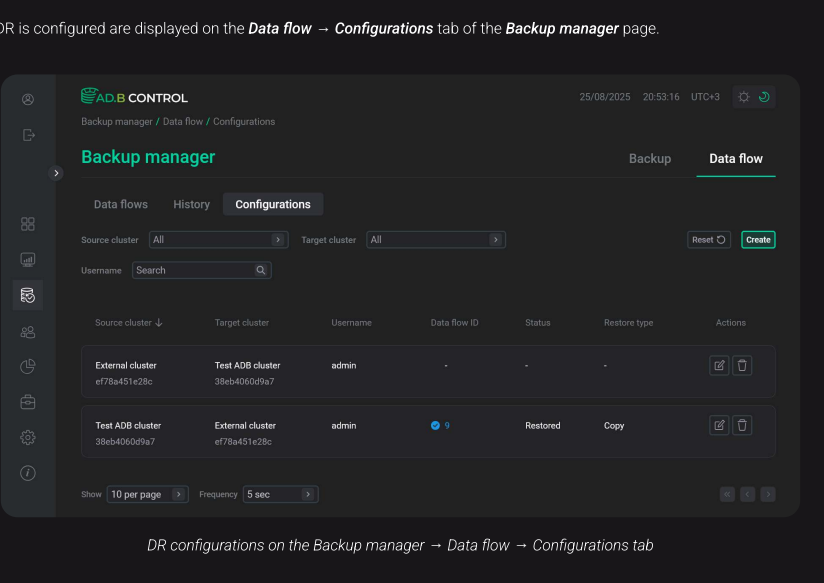
Mirror segments

- *pg_hba.conf*
- *pg_ident.conf*
- *postgresql.conf*
- *postgresql.auto.conf*
- *internal.auto.conf*
- *recovery.conf*

DR configuration management

View a list of configurations

All cluster pairs between which DR is configured are displayed on the ***Data flow → Configurations*** tab of the ***Backup manager*** page.



DR configurations on the Backup manager → Data flow → Configurations tab

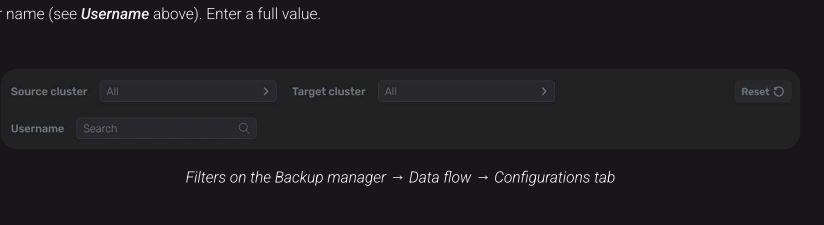
The tab displays the following information.

The "Backup manager → Data flow → Configurations" tab fields

Field	Description
Source cluster	The source cluster from which the data is to be copied
Target cluster	The target cluster to which the data is to be copied
Username	A login of the user who created the current configuration via the ADB Control web interface
Data flow ID	A unique identifier of the latest data flow that was launched for a pair of clusters defined in the current configuration
Status	A status of the ***Data flow ID*** data flow. See ***Status*** below
Restore type	The restore type that was selected during the data flow configuration

Above the table with DR configurations, there are filters that you can use to select specific data. Available filters are listed below:

- ***Source cluster*** — filter by the source cluster name (see ***Source cluster*** above). Select a value from the drop-down list.
- ***Target cluster*** — filter by the target cluster name (see ***Target cluster*** above). Select a value from the drop-down list.
- ***Username*** — filter by the user name (see ***Username*** above). Enter a full value.

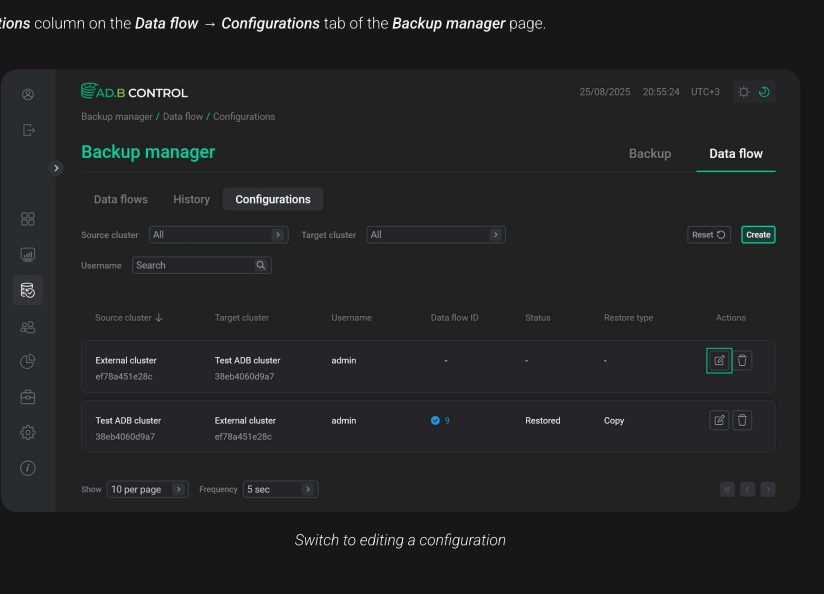


Filters on the Backup manager → Data flow → Configurations tab

Edit a configuration

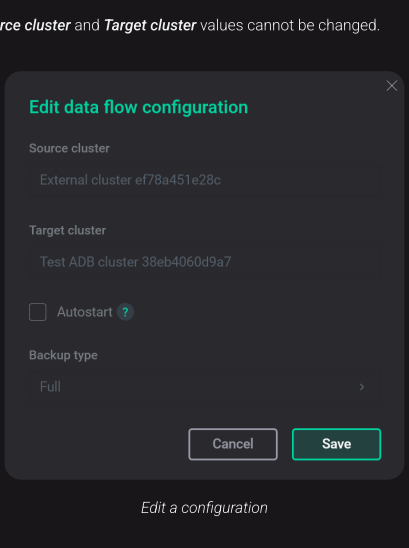
To edit a DR configuration, follow the steps:

1. Click the icon in the ***Actions*** column on the ***Data flow → Configurations*** tab of the ***Backup manager*** page.



Switch to editing a configuration

2. In the window that opens, edit necessary fields. The ***Source cluster*** and ***Target cluster*** values cannot be changed.



Edit data flow configuration
Source cluster: External cluster ef78a451e28c
Target cluster: Test ADB cluster 38eb4060d9a7
Autostart
Backup type: Full

Edit a configuration

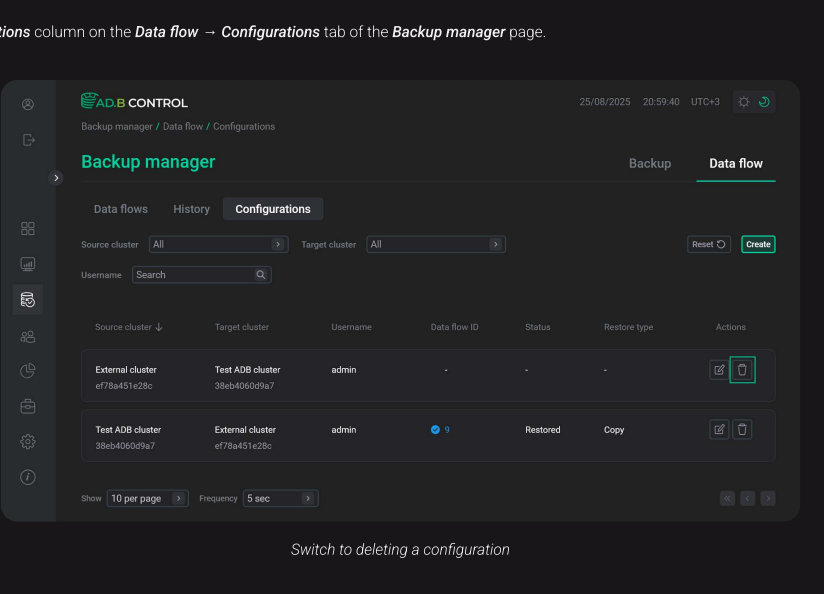
3. Click ***Save***.

Delete a configuration

> CAUTION
>
> Do not delete a configuration of the currently running data flow.

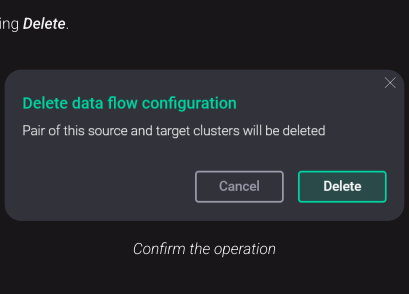
To delete an existing configuration, follow the steps:

1. Click the icon in the ***Actions*** column on the ***Data flow → Configurations*** tab of the ***Backup manager*** page.



Switch to deleting a configuration

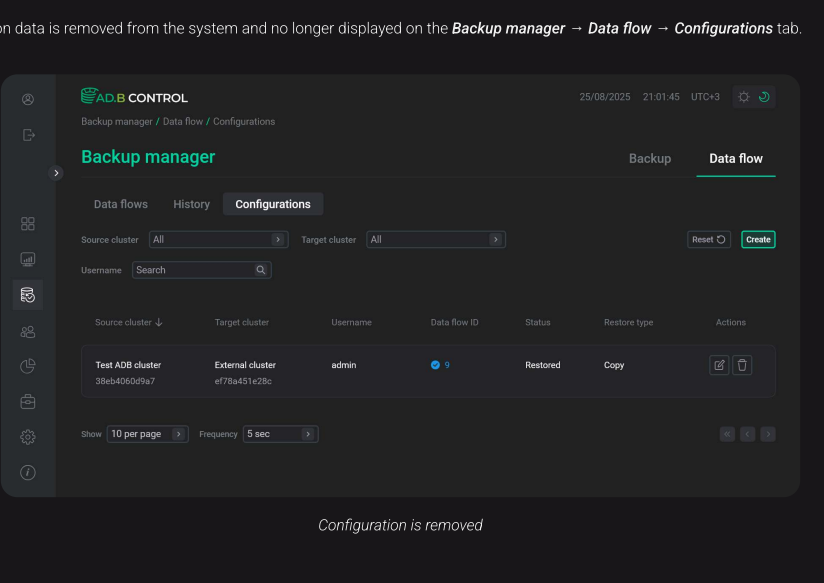
2. In the window that opens, confirm the operation by clicking ***Delete***.



Delete data flow configuration
Pair of this source and target clusters will be deleted

Confirm the operation

As a result, the configuration data is removed from the system and no longer displayed on the ***Backup manager → Data flow → Configurations*** tab.

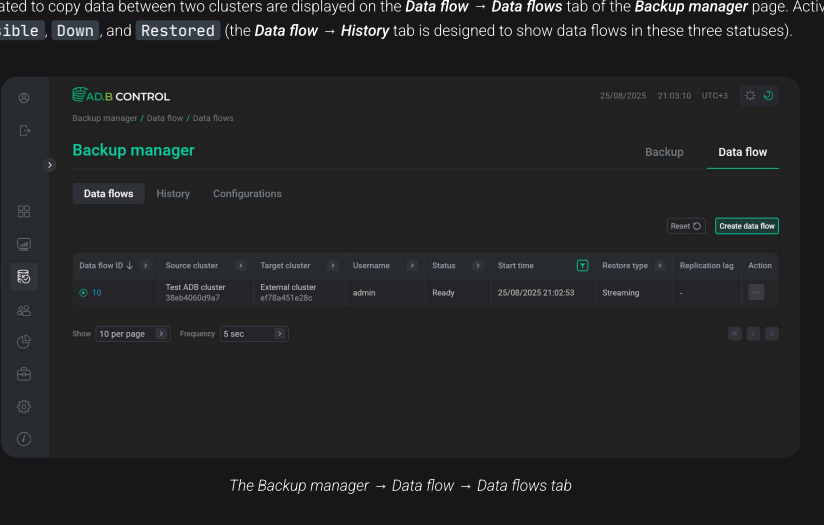


Configuration is removed

Data flow management

View active data flows

All active data flows that are created to copy data between two clusters are displayed on the ***Data flow → Data flows*** tab of the ***Backup manager*** page. Active data flows have the statuses different from `Impossible`, `Down`, and `Restored` (the ***Data flow → History*** tab is designed to show data flows in these three statuses).

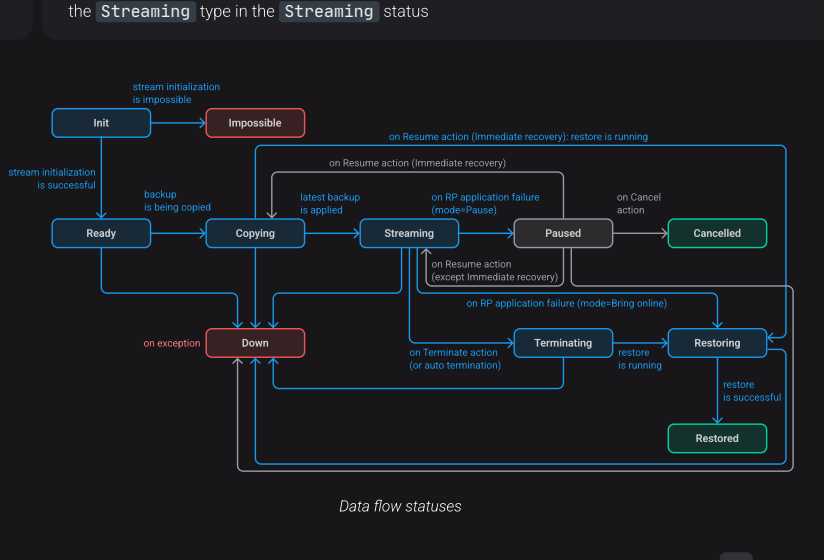


The Backup manager → Data flow → Data flows tab

The tab displays the following information.

The "Backup manager → Data flow → Data flows" tab fields

Field	Description
Data flow ID	A unique identifier of the data flow
Source cluster	The source cluster from which the data is to be copied within the current data flow
Target cluster	The target cluster to which the data is to be copied within the current data flow
Username	A login of the user who created the current data flow via the ADB Control web interface
Status	A data flow status. Possible values: `Init` — the data flow is being initialized; `Ready` — the data flow is ready for data restore; `Paused` — the data flow with the `Streaming` type is temporarily stopped (see ***On restore points streaming failure*** above); `Copying` — the backup is being copied; `Restoring` — the restore operation is running; `Streaming` — the data flow with the `Streaming` type is running; `Terminating` — the data flow launch is being terminated manually or automatically (see the note above). For more information on possible transitions between statuses, see the diagram below
Start time	A data flow creation timestamp in the `DD/MM/YYYY HH:mm:ss` format
Restore type	The restore type that was selected during the data flow configuration
Replication lag	How much the target cluster lags from the source cluster. Calculated as the difference between the current time and the time when the latest restore point was applied to the target cluster during the streaming process. Available only for data flows with the `Streaming` type in the `Streaming` status



Init
stream initialization is impossible
Impossible
stream initialization is successful
on Resume action (Immediate recovery): restore is running
on Resume action (Immediate recovery)
backup is being copied
latest backup is applied
on RP application failure (mode=Pause)
on Cancel action
Ready
Copying
Streaming
Paused
Cancelled
on Resume action (except Immediate recovery)
on RP application failure (mode=Bring online)
on exception
Down
on Terminate action (or auto termination)
Terminating
restore is running
Restoring
restore is successful
Restored

Data flow statuses

In the column headers of the table with a list of data flows, there are filters that you can use to select specific data. To open a filter, click the icon. For those columns where the set of possible values is limited (e.g. ***Status***), you can select a value from the drop-down list. For some columns (e.g. ***Username***), the search value should be entered. For columns that show date and time (e.g. ***Start time***), the time range can be selected from the calendar.

The icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

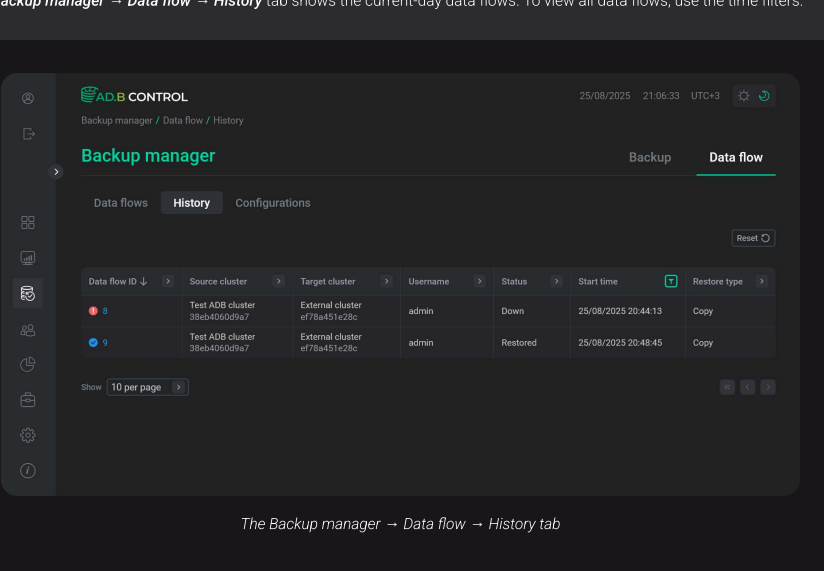
View finished data flows

The ***Backup manager → Data flow → History*** tab is used to display completed data flows, which have the following statuses:

- `Impossible` — the data flow launch is impossible.
- `Down` — the data flow launch is failed.
- `Restored` — the data flow launch is successful.
- `Cancelled` — the previously stopped data flow with the `Streaming` type is cancelled (see ***On restore points streaming failure*** above).

The content of the ***History*** tab is completely identical to the ***Data flows*** tab described above, except that the former does not contain the ***Replication lag*** column. Also, there are no available actions for data flows on the ***History*** tab.

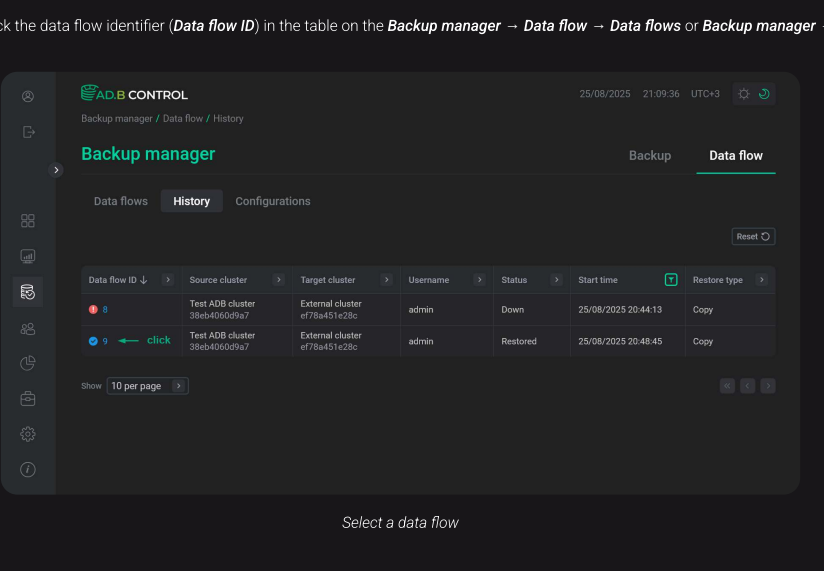
> IMPORTANT
>
> By default, the table on the ***Backup manager → Data flow → History*** tab shows the current-day data flows. To view all data flows, use the time filters.



The Backup manager → Data flow → History tab

View the data flow details

To view the data flow details, click the data flow identifier (***Data flow ID***) in the table on the ***Backup manager → Data flow → Data flows*** or ***Backup manager → Data flow → History*** tab.

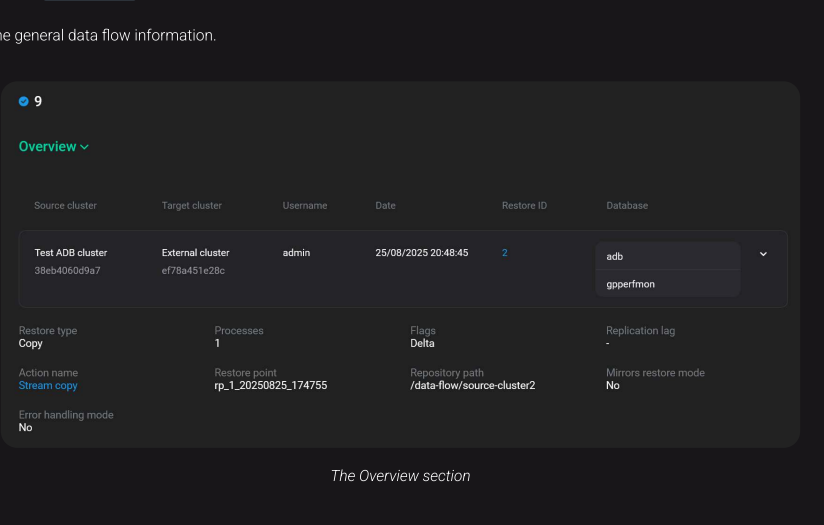


Select a data flow

The next page contains the following sections:

- ***Overview***
- ***Statuses***
- ***Restore points*** (shown only for the `Streaming` restore type)

The ***Overview*** section displays the general data flow information.



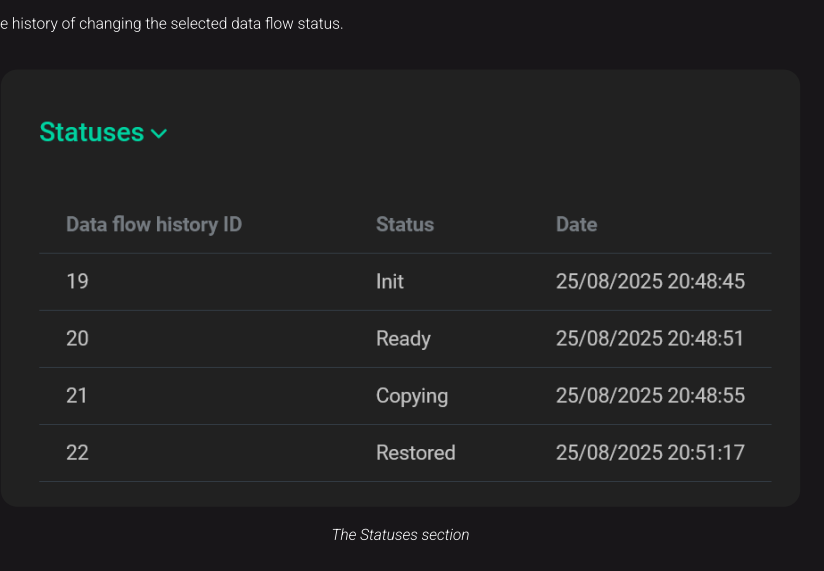
9
Overview
Source cluster: Test ADB cluster 38eb4060d9a7
Target cluster: External cluster ef78a451e28c
Username: admin
Date: 25/08/2025 20:48:45
Restore ID: 1
Database: adb, gpperfmon
Restore type: Copy
Processes: 1
Flags: Delta
Replication lag: -
Action name: Stream copy
Restore point: rp_1_20250825_174755
Repository path: /data-flow/source-cluster2
Mirrors restore mode: No
Error handling mode: No

The Overview section

The Overview section fields

Field	Description
Source cluster	A source cluster name
Target cluster	A target cluster name
Username	A login of the user who created the current data flow via the ADB Control web interface
Date	A data flow creation timestamp in the `DD/MM/YYYY HH:mm:ss` format
Restore ID	A unique identifier of the restore operation in the target cluster. To view the restore details, click the identifier
Database	A list of databases that were selected during the data flow configuration
Restore type	The restore type that was selected during the data flow configuration
Processes	A number of processes that was selected during the data flow configuration
Flags	The flags that were selected during the data flow configuration (see the ***Delta***, ***Force***, ***Verify*** flags above)
Replication lag	How much the target cluster lags from the source cluster. Calculated as the difference between the current time and the time when the latest restore point was applied to the target cluster during the streaming process. Available only for data flows with the `Streaming` type in the `Streaming` status
Action name	A name of the action within which the data flow was launched. To view the action details, click the name
Restore point	An identifier of the restore point that was selected during the data flow configuration
Repository path	A repository path that was selected during the data flow configuration
Mirrors restore mode	The restore mode selected for mirrors during the data flow configuration (see ***Restore mirrors*** above)
Error handling mode	The error handling mode specified during the data flow configuration (see ***On restore points streaming failure*** above)

The ***Statuses*** section displays the history of changing the selected data flow status.



Statuses

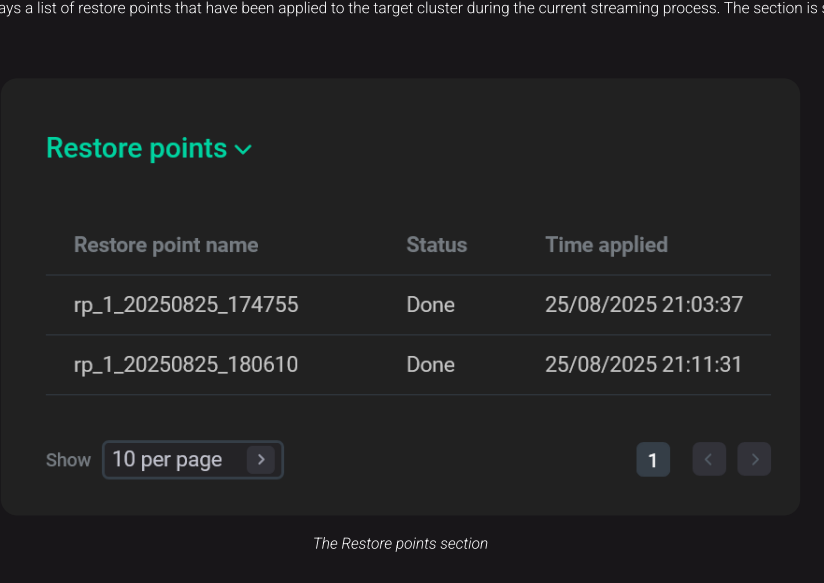
Data flow history ID	Status	Date
19	Init	25/08/2025 20:48:45
20	Ready	25/08/2025 20:48:51
21	Copying	25/08/2025 20:48:55
22	Restored	25/08/2025 20:51:17

The Statuses section

The Statuses section fields

Field	Description
Data flow history ID	A unique identifier of the data flow status change event
Status	A data flow status at the time specified in the ***Date*** field
Date	A timestamp of the data flow status change event in the `DD/MM/YYYY HH:mm:ss` format

The ***Restore points*** section displays a list of restore points that have been applied to the target cluster during the current streaming process. The section is shown only for data flows with the `Streaming` type.



Restore points

Restore point name	Status	Time applied
rp_1_20250825_174755	Done	25/08/2025 21:03:37
rp_1_20250825_180610	Done	25/08/2025 21:11:31

The Restore points section

The Restore points section fields

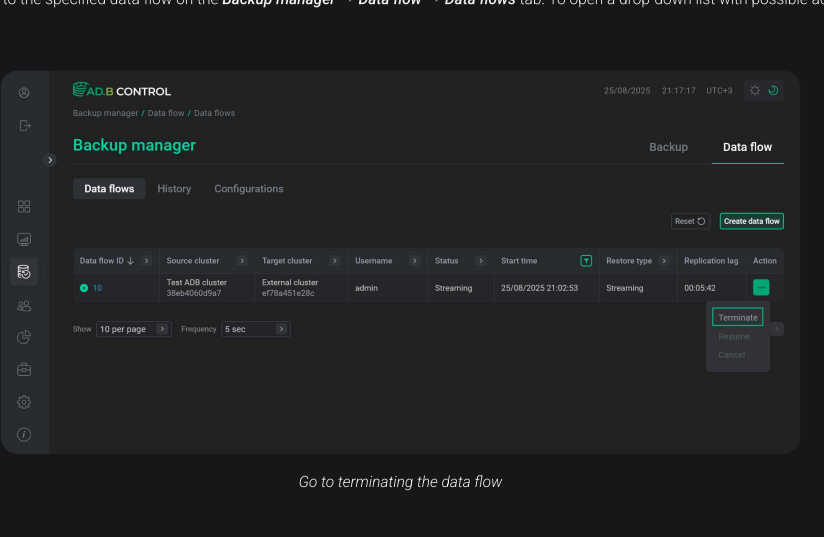
Field	Description
Restore point name	A name that was assigned to the restore point in the source cluster
Status	A status of applying the restore point to the target cluster: `Init` — initialized; `Done` — finished successfully; `Failed` — finished with an error.
Time applied	A timestamp when the restore point was applied to the target cluster (in the `DD/MM/YYYY HH:mm:ss` format)

Terminate a data flow

> IMPORTANT
>
> The ***Terminate*** action is available only for data flows with the `Streaming` type in the `Streaming` status.

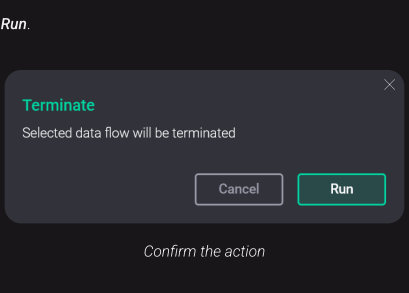
To terminate a running data flow with the `Streaming` type, follow the steps:

1. Apply the ***Terminate*** action to the specified data flow on the ***Backup manager → Data flow → Data flows*** tab. To open a drop-down list with possible actions, click ⋯ in the ***Action*** column.



Go to terminating the data flow

2. In the window that opens, confirm the action by clicking ***Run***.



Terminate
Selected data flow will be terminated

Confirm the action

3. As a result, the ***Terminate stream*** action starts for the target cluster. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
During the termination process:

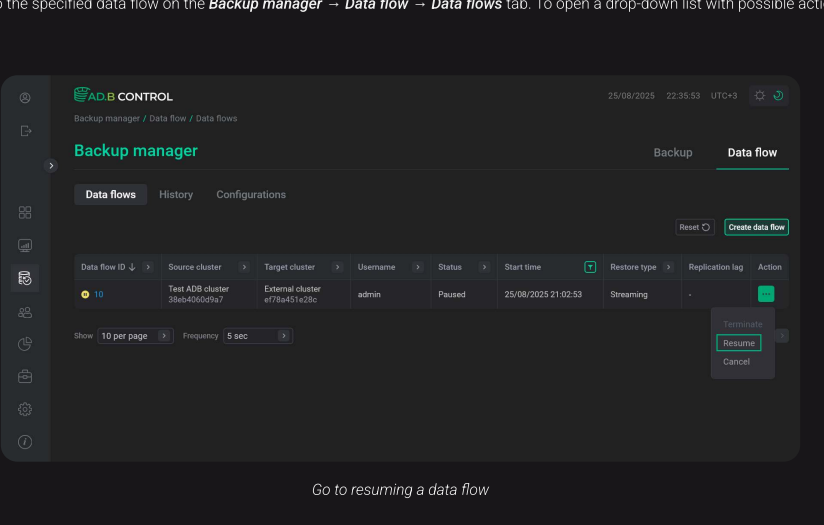
   - A status of the previously created action ***Streaming cluster*** is changed to `Done`.
   - A status of the restore operation for the target cluster is changed to `Done`.
   - A status of the selected data flow is changed as follows: `Streaming` → `Terminating` → `Restoring` → `Restored`.
   - The selected data flow is moved from the ***Backup manager → Data flow → Data flows*** tab to the ***Backup manager → Data flow → History*** tab.

Resume a data flow

> IMPORTANT
>
> The ***Resume*** action is available only for data flows with the `Streaming` type in the `Paused` status starting with ADBM 2.4.0.

To resume a stopped data flow with the `Streaming` type (see ***On restore points streaming failure*** above), follow the steps:

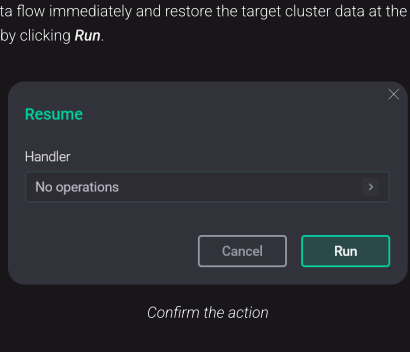
1. Apply the ***Resume*** action to the specified data flow on the ***Backup manager → Data flow → Data flows*** tab. To open a drop-down list with possible actions, click ⋯ in the ***Action*** column.



Go to resuming a data flow

2. In the window that opens, you can change the data flow error handling mode by selecting one of the values in the ***Handler*** field:

   - `No operations` — leave the `Pause` data flow handler without any changes (see ***On restore points streaming failure*** above).
   - `Set stream handler to Bring Online` — change the data flow error handling mode from `Pause` to `Bring online` (see ***On restore points streaming failure*** above).
   - `Immediate recovery` — terminate the current data flow immediately and restore the target cluster data at the moment of the last valid restore point.
   After you select the ***Handler*** value, confirm the action by clicking ***Run***.



Resume
Handler: No operations

Confirm the action

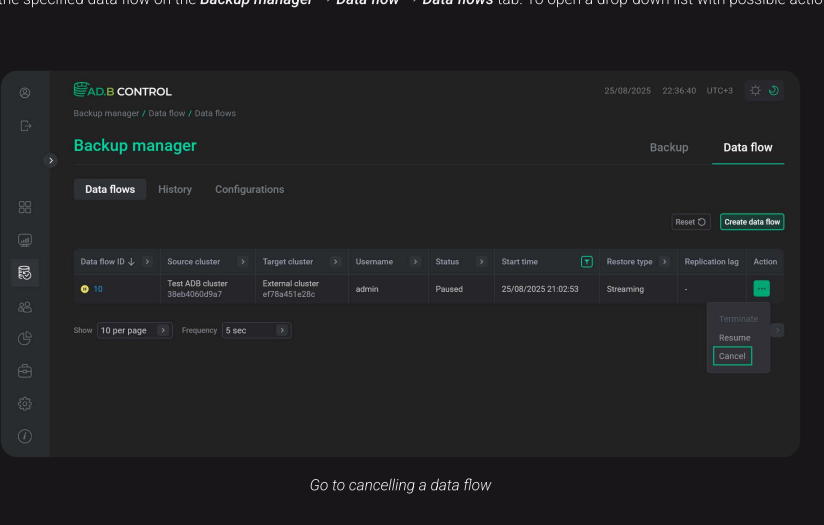
3. As a result, the ***Resume Stream*** action starts for the target cluster. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
If the data flow resumes successfully, it acquires the `Streaming` status. If you selected the target cluster restore in the action settings and it does not fail, the data flow status changes to `Restored`.

Cancel a data flow

> IMPORTANT
>
> The ***Cancel*** action is available only for data flows with the `Streaming` type in the `Paused` status starting with ADBM 2.4.0.

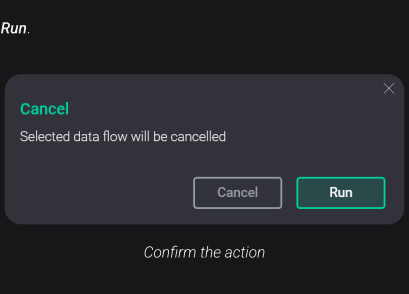
To cancel a stopped data flow with the `Streaming` type (see ***On restore points streaming failure*** above), follow the steps:

1. Apply the ***Cancel*** action to the specified data flow on the ***Backup manager → Data flow → Data flows*** tab. To open a drop-down list with possible actions, click ⋯ in the ***Action*** column.



Go to cancelling a data flow

2. In the window that opens, confirm the action by clicking ***Run***.



Cancel
Selected data flow will be cancelled

Confirm the action

3. As a result, the ***Cancel Stream*** action starts for the target cluster. This action, in turn, generates several subactions. You can see all of them on the ***Actions*** tab (for more details, see View actions in ADBM).
After all actions are successfully finished, the data flow status changes to `Cancelled`.

View actions in ADBM

Daria Barysheva

Contents

To Table of Contents

List of actions

Action details

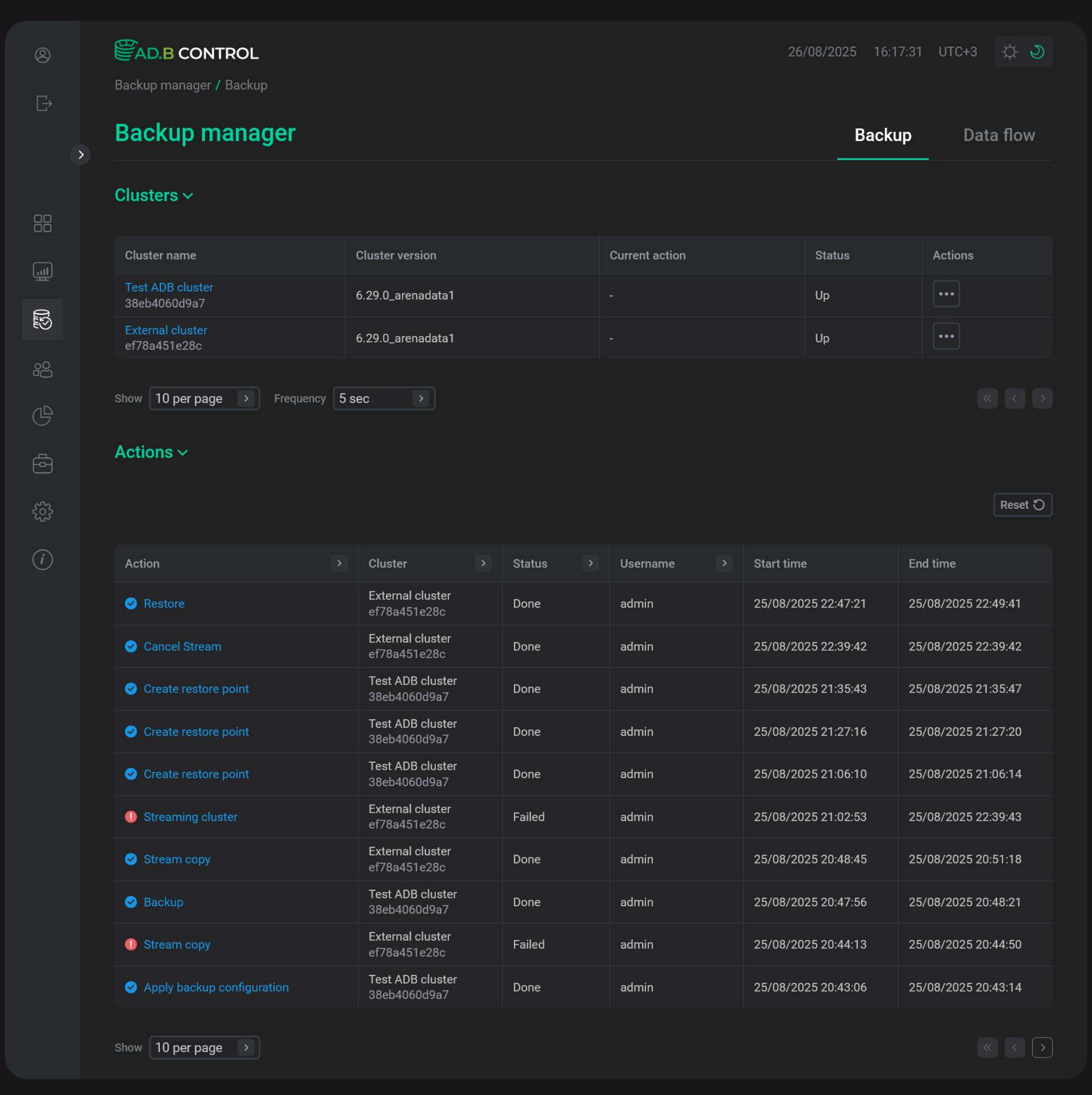
In ADBM, an **action** means any API call, SQL request, or internal procedure that starts when backup configurations or operations related to backups and DR are being applied.

Each action can trigger several other related actions — **subactions**.

List of actions

To view a list of actions, follow the steps:

1. Open the Backup manager page via the ADB Control web interface. The ***Actions*** section of the next page contains a table with general information on actions for all available ADB clusters.



AD.B CONTROL 26/08/2025 16:17:31 UTC+3

Backup manager / Backup

Backup manager Backup Data flow

Clusters

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show 10 per page Frequency 5 sec

Actions

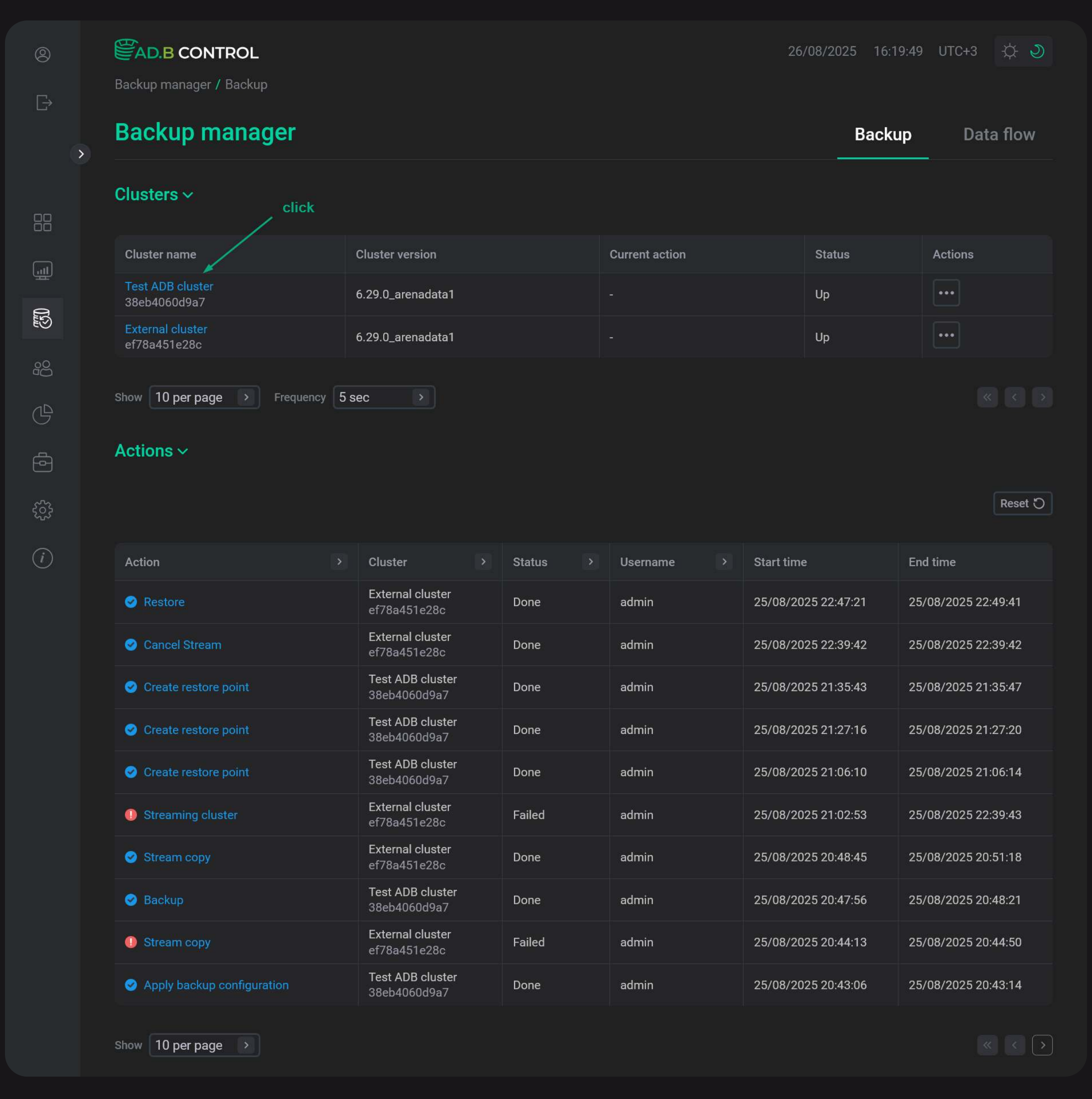
Reset

Action	Cluster	Status	Username	Start time	End time
Restore	External cluster ef78a451e28c	Done	admin	25/08/2025 22:47:21	25/08/2025 22:49:41
Cancel Stream	External cluster ef78a451e28c	Done	admin	25/08/2025 22:39:42	25/08/2025 22:39:42
Create restore point	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 21:35:43	25/08/2025 21:35:47
Create restore point	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 21:27:16	25/08/2025 21:27:20
Create restore point	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 21:06:10	25/08/2025 21:06:14
Streaming cluster	External cluster ef78a451e28c	Failed	admin	25/08/2025 21:02:53	25/08/2025 22:39:43
Stream copy	External cluster ef78a451e28c	Done	admin	25/08/2025 20:48:45	25/08/2025 20:51:18
Backup	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 20:47:56	25/08/2025 20:48:21
Stream copy	External cluster ef78a451e28c	Failed	admin	25/08/2025 20:44:13	25/08/2025 20:44:50
Apply backup configuration	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 20:43:06	25/08/2025 20:43:14

Show 10 per page

The Actions section

2. If you want to display actions for a specified cluster, click the cluster name in the table that is located in the ***Clusters*** section of the opened page.



AD.B CONTROL 26/08/2025 16:19:49 UTC+3

Backup manager / Backup

Backup manager Backup Data flow

Clusters

click

Cluster name	Cluster version	Current action	Status	Actions
Test ADB cluster 38eb4060d9a7	6.29.0_arenadata1	-	Up	•••
External cluster ef78a451e28c	6.29.0_arenadata1	-	Up	•••

Show 10 per page Frequency 5 sec

Actions

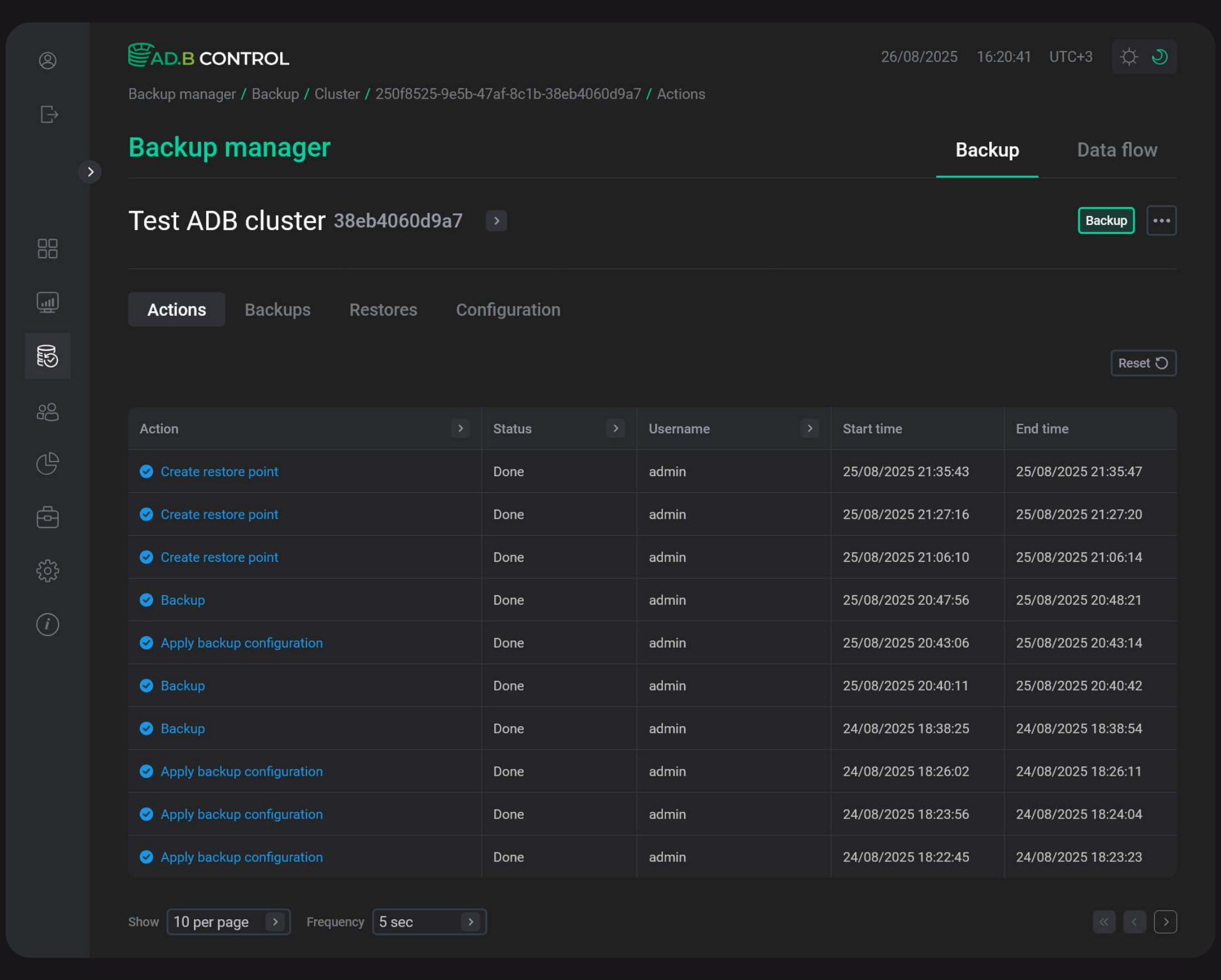
Reset

Action	Cluster	Status	Username	Start time	End time
Restore	External cluster ef78a451e28c	Done	admin	25/08/2025 22:47:21	25/08/2025 22:49:41
Cancel Stream	External cluster ef78a451e28c	Done	admin	25/08/2025 22:39:42	25/08/2025 22:39:42
Create restore point	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 21:35:43	25/08/2025 21:35:47
Create restore point	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 21:27:16	25/08/2025 21:27:20
Create restore point	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 21:06:10	25/08/2025 21:06:14
Streaming cluster	External cluster ef78a451e28c	Failed	admin	25/08/2025 21:02:53	25/08/2025 22:39:43
Stream copy	External cluster ef78a451e28c	Done	admin	25/08/2025 20:48:45	25/08/2025 20:51:18
Backup	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 20:47:56	25/08/2025 20:48:21
Stream copy	External cluster ef78a451e28c	Failed	admin	25/08/2025 20:44:13	25/08/2025 20:44:50
Apply backup configuration	Test ADB cluster 38eb4060d9a7	Done	admin	25/08/2025 20:43:06	25/08/2025 20:43:14

Show 10 per page

Select the ADB cluster

The ***Actions*** tab of the opened page contains a table with general information on actions for the selected cluster.



AD.B CONTROL 26/08/2025 16:20:41 UTC+3

Backup manager / Backup / Cluster / 250f8525-9e5b-47af-8c1b-38eb4060d9a7 / Actions

Backup manager Backup Data flow

Test ADB cluster 38eb4060d9a7 Backup •••

Actions Backups Restores Configuration

Reset

Action	Status	Username	Start time	End time
Create restore point	Done	admin	25/08/2025 21:35:43	25/08/2025 21:35:47
Create restore point	Done	admin	25/08/2025 21:27:16	25/08/2025 21:27:20
Create restore point	Done	admin	25/08/2025 21:06:10	25/08/2025 21:06:14
Backup	Done	admin	25/08/2025 20:47:56	25/08/2025 20:48:21
Apply backup configuration	Done	admin	25/08/2025 20:43:06	25/08/2025 20:43:14
Backup	Done	admin	25/08/2025 20:40:11	25/08/2025 20:40:42
Backup	Done	admin	24/08/2025 18:38:25	24/08/2025 18:38:54
Apply backup configuration	Done	admin	24/08/2025 18:26:02	24/08/2025 18:26:11
Apply backup configuration	Done	admin	24/08/2025 18:23:56	24/08/2025 18:24:04
Apply backup configuration	Done	admin	24/08/2025 18:22:45	24/08/2025 18:23:23

Show 10 per page Frequency 5 sec

The Actions tab

Both tables mentioned above contain the following information on actions.

Action list fields

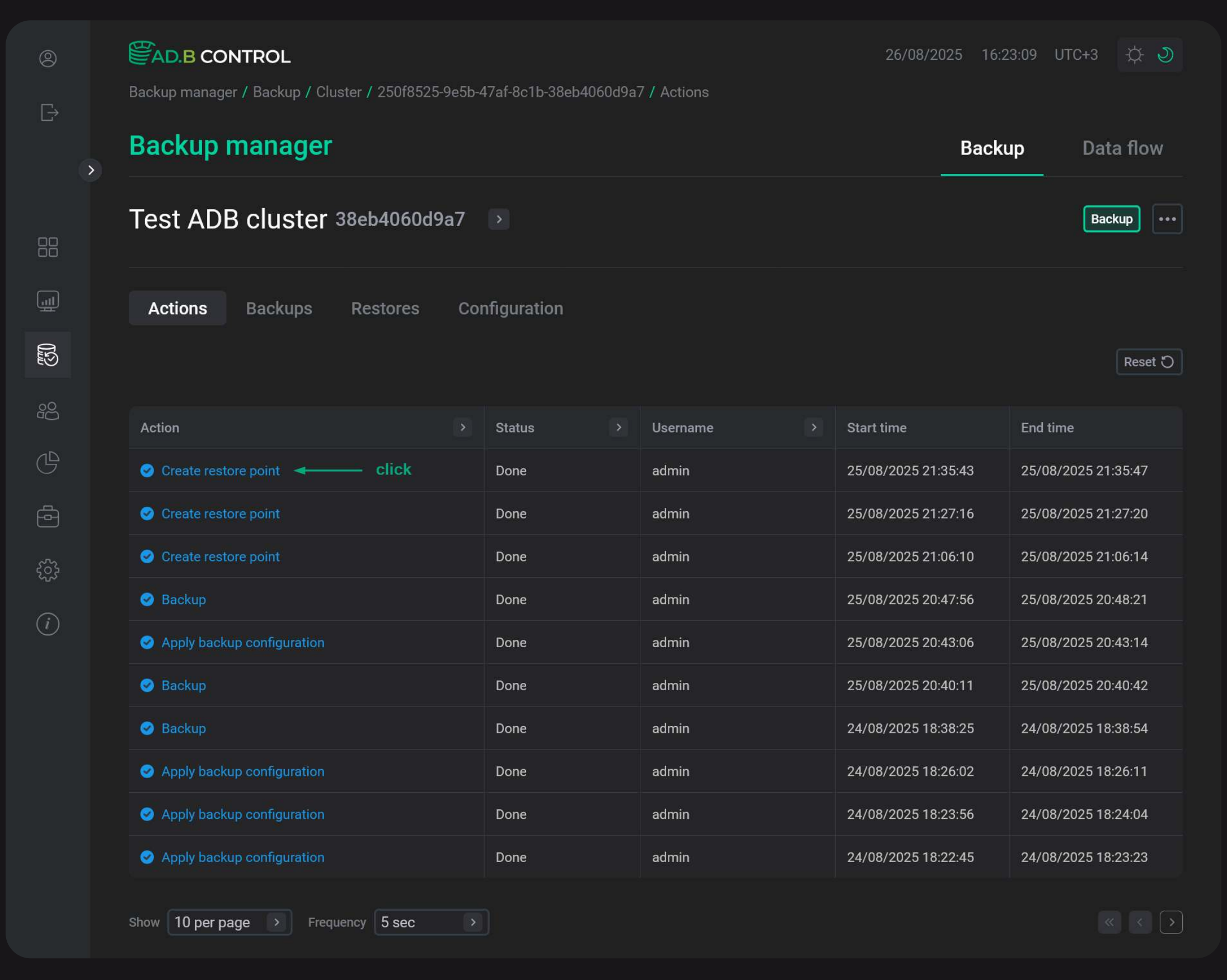
Field	Description
Action	An action full name
Cluster	A name of the cluster on which the selected action was executed. Displayed only if no specific cluster is selected on the ***Backup manager*** page
Status	An action status. Possible values: ▪ `Started` — the action is being started. ▪ `Stopped` — the action is stopped. ▪ `Done` — the action is completed successfully. ▪ `Failed` — the action is failed. ▪ `Terminating` — the action is being terminated. ▪ `Awaiting` — the action is waiting for something. ▪ `Unknown` — the action status is unknown. ▪ `Warning` — something went wrong during the action, but it is not critical for the parent action flow.
Username	The user who performed the selected action
Start time	An action start timestamp in the `DD/MM/YYYY HH:mm:ss` format
End time	An action end timestamp in the `DD/MM/YYYY HH:mm:ss` format

In the column headers of the tables that are located in the Actions section and on the Actions tab, there are filters that you can use to select specific data. To open a filter, click the › icon. For those columns where the set of possible values is limited (e.g. ***Action***), you can select a value from the drop-down list. For some columns (e.g. ***Username***), the search value should be entered.

The ▼ icon means that a filter is defined for the column. To reset all filters, click ***Reset***.

Action details

To view the action details, click the action name (***Action***) in the table located in the Actions section or on the Actions tab.



AD.B CONTROL 26/08/2025 16:23:09 UTC+3

Backup manager / Backup / Cluster / 250f8525-9e5b-47af-8c1b-38eb4060d9a7 / Actions

Backup manager Backup Data flow

Test ADB cluster 38eb4060d9a7 Backup •••

Actions Backups Restores Configuration

Reset

Action	Status	Username	Start time	End time
Create restore point ← click	Done	admin	25/08/2025 21:35:43	25/08/2025 21:35:47
Create restore point	Done	admin	25/08/2025 21:27:16	25/08/2025 21:27:20
Create restore point	Done	admin	25/08/2025 21:06:10	25/08/2025 21:06:14
Backup	Done	admin	25/08/2025 20:47:56	25/08/2025 20:48:21
Apply backup configuration	Done	admin	25/08/2025 20:43:06	25/08/2025 20:43:14
Backup	Done	admin	25/08/2025 20:40:11	25/08/2025 20:40:42
Backup	Done	admin	24/08/2025 18:38:25	24/08/2025 18:38:54
Apply backup configuration	Done	admin	24/08/2025 18:26:02	24/08/2025 18:26:11
Apply backup configuration	Done	admin	24/08/2025 18:23:56	24/08/2025 18:24:04
Apply backup configuration	Done	admin	24/08/2025 18:22:45	24/08/2025 18:23:23

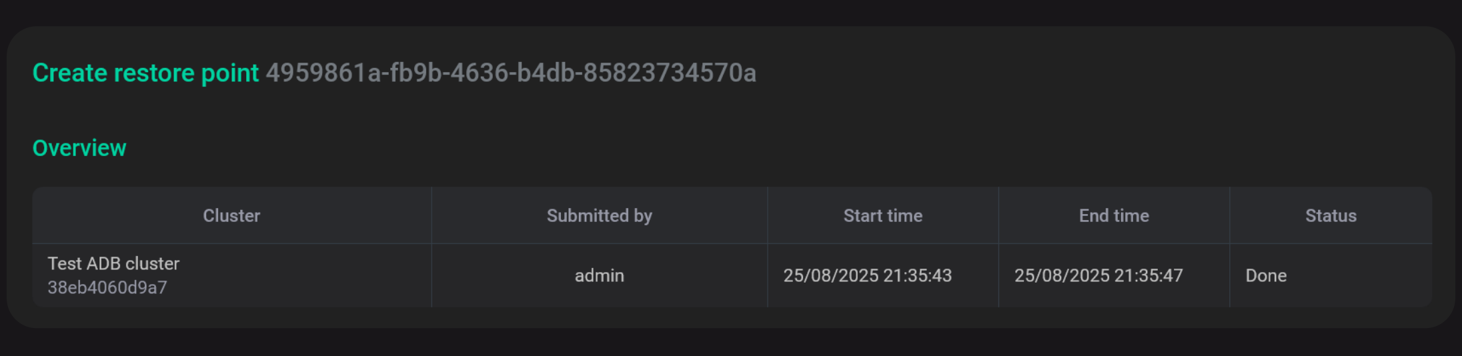
Show 10 per page Frequency 5 sec

Select an action

The next page contains the following sections:

- ***Overview***
- ***Subactions***

The ***Overview*** section displays the action name, identifier, and general information.



Create restore point 4959861a-fb9b-4636-b4db-85823734570a

Overview

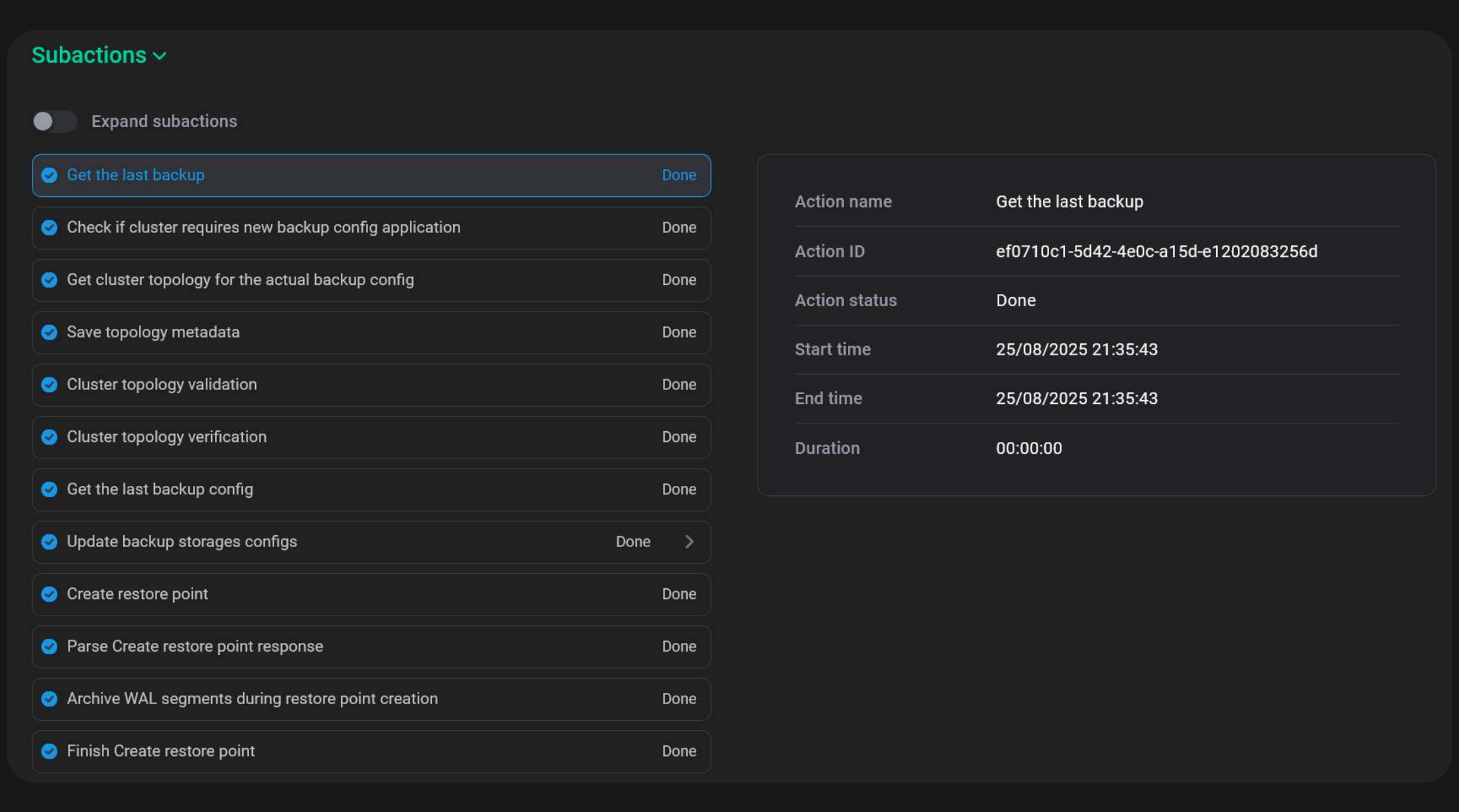
Cluster	Submitted by	Start time	End time	Status
Test ADB cluster 38eb4060d9a7	admin	25/08/2025 21:35:43	25/08/2025 21:35:47	Done

The Overview section

Fields of the Overview section

Field	Description
Cluster	A cluster name (see ***Cluster*** above)
Submitted by	The user who submitted the selected action
Start time	An action start timestamp in the `DD/MM/YYYY HH:mm:ss` format
End time	An action end timestamp in the `DD/MM/YYYY HH:mm:ss` format
Status	An action status (see ***Status*** above)

The ***Subactions*** section displays a list of subactions that are triggered by the top-level action. All subactions are ordered by the start timestamp — from the first action to the last one. For each action in the list, the action name and status are displayed.



Subactions

Expand subactions

Get the last backup	Done
Check if cluster requires new backup config application	Done
Get cluster topology for the actual backup config	Done
Save topology metadata	Done
Cluster topology validation	Done
Cluster topology verification	Done
Get the last backup config	Done
Update backup storages configs	Done ›
Create restore point	Done
Parse Create restore point response	Done
Archive WAL segments during restore point creation	Done
Finish Create restore point	Done

Action name	Get the last backup
Action ID	ef0710c1-5d42-4e0c-a15d-e1202083256d
Action status	Done
Start time	25/08/2025 21:35:43
End time	25/08/2025 21:35:43
Duration	00:00:00

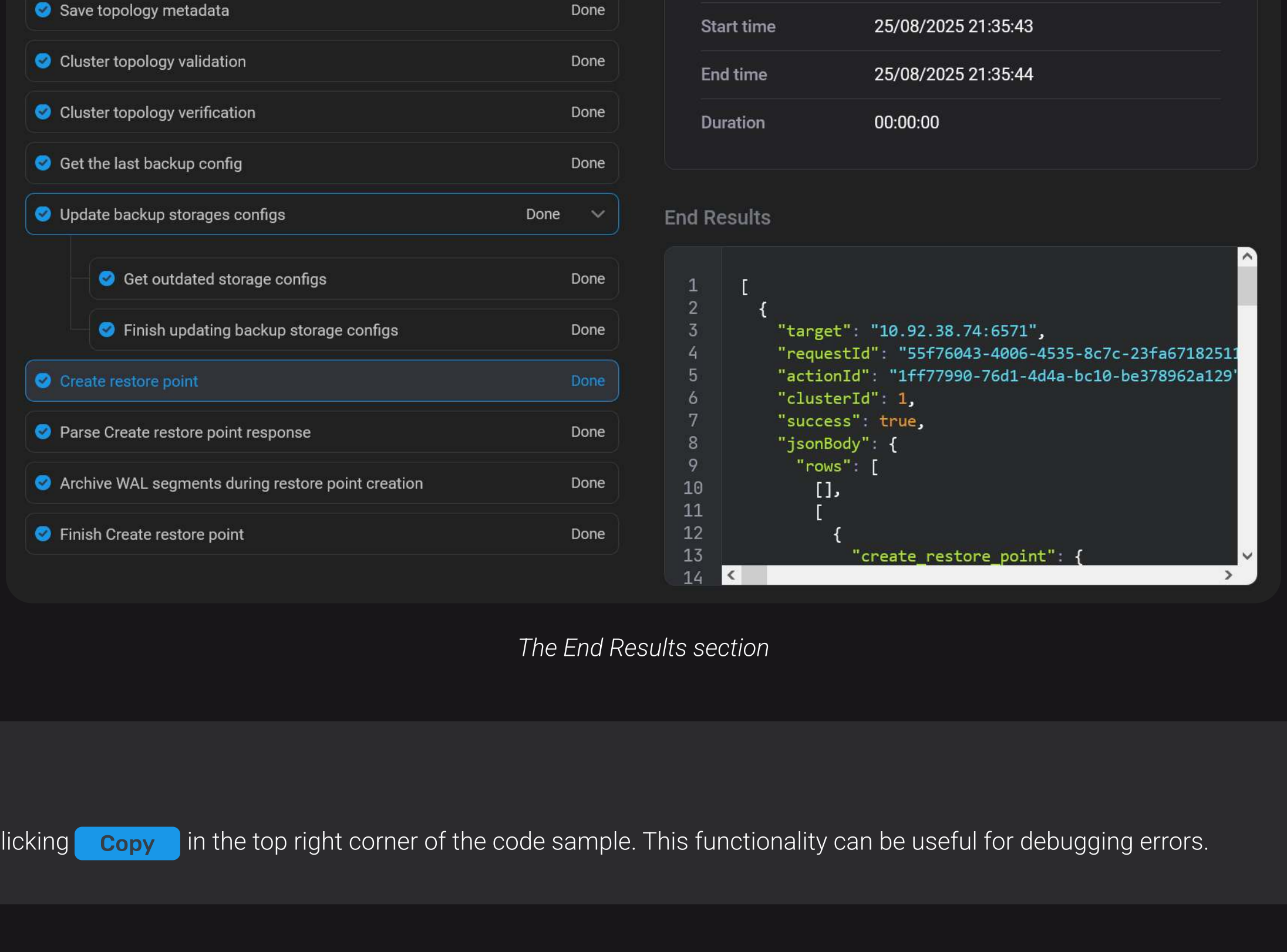
The Subactions section

When you click any subaction in the list, the detailed information on it is displayed in the right part of the screen.

Subaction details

Field	Description
Action name	A subaction name
Action ID	A subaction identifier
Action status	A subaction status (see ***Status*** above)
Start time	A subaction start timestamp in the `DD/MM/YYYY HH:mm:ss` format
End time	A subaction end timestamp in the `DD/MM/YYYY HH:mm:ss` format
Duration	A subaction duration in hours, minutes, and seconds

For some low-level actions, the ***End Results*** section with the subaction result can be displayed below the detailed information.



Subactions

Expand subactions

Get the last backup	Done
Check if cluster requires new backup config application	Done
Get cluster topology for the actual backup config	Done
Save topology metadata	Done
Cluster topology validation	Done
Cluster topology verification	Done
Get the last backup config	Done
Update backup storages configs	Done
Get outdated storage configs	Done
Finish updating backup storage configs	Done
Create restore point	Done
Parse Create restore point response	Done
Archive WAL segments during restore point creation	Done
Finish Create restore point	Done

Action name	Create restore point
Action ID	1ff77990-76d1-4d4a-bc10-be378962a129
Action status	Done
Start time	25/08/2025 21:35:43
End time	25/08/2025 21:35:44
Duration	00:00:00

End Results

```
[
  {
    "target": "10.92.38.74:6571",
    "requestId": "55f76043-4006-4535-8c7c-23fa67182511
    "actionId": "1ff77990-76d1-4d4a-bc10-be378962a129"
    "clusterId": 1,
    "success": true,
    "jsonBody": {
      "rows": [
        [],
        [
          {
            "create_restore_point": {
```

The End Results section

> **TIP**
>
> You can copy the result by clicking **Copy** in the top right corner of the code sample. This functionality can be useful for debugging errors.

Work with ADBM CLI

Daria Barysheva

Contents

To Table of Contents

Overview

backup repo-config create

backup repo-config ls

backup repo-config delete

backup config create

backup config ls

backup config delete

backup create

backup restore

backup ls

backup report

backup delete

backup status

Overview

Starting with 2.0.4, Arenadata DB Backup Manager supports a command-line tool for backup management — ADBM CLI. ADBM CLI is installed automatically in the Enterprise Edition of ADB along with the ADBM service. To use ADBM CLI, connect to the ADB coordinator host under the `gpadmin` user and run `adbm`:

```
$ sudo su - gpadmin
$ adbm <command_name> <command_options>
```

where:

- `<command_name>` — a name of the command. All supported commands are described in detail below.
- `<command_options>` — options of the selected command.

To check the current version of ADBM CLI, you can run `adbm --version | -v`:

```
$ adbm -v
```

Result:

```
adbm-cli-2.8.0
```

IMPORTANT

- Currently, ADBM CLI allows you to work only with logical backups (see `--backup-format logical | l` in the commands below). Binary backups are not supported. If you specify `--backup-format binary | b` for any command, you will receive an error.
- All CLI options are specified in lowercase. Options without values (`-h`, `-a`, etc.) are specified separately. Merging options (for example, `-ha`) is prohibited.

backup repo-config create

The `adbm backup repo-config create` command creates a new repository configuration based on the passed options. You need repository configurations to create backup configurations via the backup config create command.

Syntax:

```
$ adbm backup repo-config create \
--backup-format | -f <backupFormat> \
--repo-type <repoType> \
[--repo-path <repoPath>] \
[--file <file>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: ▪ `binary | b` — binary backups; ▪ `logical | l` — logical backups.	Yes	—
--repo-type	A type of the data repository. Possible values: ▪ `cifs` ▪ `s3` ▪ `posix` ▪ `dds` — Data Domain Storage. For more information, see Use ADB DDBoost plugin.	Yes	—
--repo-path	The directory where all necessary backup files (metadata files and data files) are stored . Cannot be specified if `--repo-type s3|dds` is used	Yes if `--repo-type` = `posix|cifs`	—
--file	A path to the JSON configuration file with the repository configuration parameters. When this option is used, the configuration parameters are taken from the specified file. See Parameters of the input JSON file below	Yes if `--repo-type` = `s3|dds`	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Parameters of the input JSON file

Parameters of the input JSON file (`--file`) depend on the repository type (`--repo-type`).

s3

Template:

```
{
  "bucket": "string",
  "endpoint": "string",
  "region": "string",
  "keyType": "string(enum)",
  "key": "string",
  "keySecret": "string",
  "caFile": "string",
  "uriType": "string(enum)",
  "path": "string"
```

```
}
```

JSON fields

Field	Description	Mandatory
bucket	A bucket name in the S3 storage	Yes
endpoint	An endpoint of the S3 storage	Yes
region	A region of the S3 storage	Yes
keyType	A key type of the S3 storage: ▪ `shared` — using shared keys; ▪ `auto` — retrieving temporary credentials automatically; ▪ `web-id` — retrieving web identity credentials automatically.	Yes
key	A key to access the S3 storage	Yes
keySecret	A secret key to access the S3 storage	Yes
caFile	A Certificate Authority (CA) PEM file to access the S3 storage	Yes
uriType	An endpoint type (see `endpoint` above). Possible values: ▪ `host` ▪ `path`	Yes
path	A repository where backups and archived WAL segments are stored	Yes

dds

Template:

```
{
  "hostname": "string",
  "username": "string",
  "password": "string",
  "storageUnit": "string",
  "directory": "string",
  "writeBufferSize": integer,
  "readBufferSize": integer,
  "executablePath": "string"
}
```

JSON fields

Field	Description	Mandatory
hostname	An IP address or name of the host that provides operations with DDBoost	Yes
username	A name of the user who is granted permissions to work with DDBoost. The user is configured on the DDBoost side. This is neither operating system nor ADB user	Yes
password	A password of the user who is granted permissions to work with DDBoost	Yes
storageUnit	A name of the storage unit that is configured on the DDBoost side	Yes

directory	A name of the directory in the DDBoost file system. That directory is used to store all backup files created when the `gpbackup` utility is run with the DDBoost plugin. In the */<storageUnit>/<directory>* folder, the plugin automatically creates all subdirectories corresponding to the creation date and time of each backup: */<storageUnit>/<directory>/YYYYMMDD/YYYYMMDDHHmmSS/*	Yes
writeBufferSize	A buffer size for writing data to DDBoost (in bytes) . Values from the following range are allowed: 64 <= writeBufferSize <= 1048576	Yes
readBufferSize	A buffer size for reading data from DDBoost (in bytes) . Values from the following range are allowed: 64 <= readBufferSize <= 1048576	Yes
executablePath	An absolute path to the plugin in the file system of ADB hosts (along with the plugin executable file name). The plugin should be installed on all cluster hosts in the same directory . The default path to the plugin executable file is *$GPHOME/bin/adb_ddp_plugin*	No

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains an identifier of the created repository configuration:

```
Backup repository configuration <repoConfigId> created
```

Example

Without --file

```
$ adbm backup repo-config create --backup-format logical --repo-path /dasha --repo-type posix
```

Result:

```
Create backup repository configuration in progress. Action id = afd1f77f-6060-4f28-b3bb-94ee3515a48f. Performing "Start
'Create logical backup repository configuration' action".done
Backup repository configuration 1 created
```

With --file

1. Create a JSON file with configuration parameters:

   ```
   $ vi s3.json
   ```

   ```
   {
     "bucket": "test",
     "endpoint": "storage.yandexcloud.net",
     "region": "ru-central1",
     "keyType": "shared",
     "key": "XXXXXXXXXXXXXXXXXXXX",
     "keySecret": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
     "caFile": "test.pem",
     "uriType": "host",
     "path": "/test"
   ```

```
}
```

2. Run the following command to create a repository configuration:

```
$ adbm backup repo-config create --backup-format logical --repo-type s3 --file s3.json
```

Result:

```
Create backup repository configuration in progress. Action id = 8dc913ca-4595-452f-8c37-2a483c7032a6. Performing "Start
'Create logical backup repository configuration' action".done
Backup repository configuration 9 created
```

backup repo-config ls

The `adbm backup repo-config ls` command lists existing repository configurations (previously created via the backup repo-config create command).

Syntax:

```
$ adbm backup repo-config ls \
--backup-format | -f <backupFormat> \
[--repo-config-id | -rc <repoConfigId>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: ▪ `binary | b` — binary backups; ▪ `logical | l` — logical backups.	Yes	—
--repo-config-id | -rc	An identifier of the repository configuration. If set, the command returns information only for the specified configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

The command returns a JSON file with an array of existing repository configurations, whose fields depend on the repository type.

Template:

```
[
  {
    "id" : integer,
    "repoType" : "cifs",
    "cifs" : {
      "id" : integer,
      "path" : "string"}
```

```
  },
  {
    "id" : integer,
    "repoType" : "s3",
    "s3" : {
      "id" : integer,
      "path" : "string",
      "bucket": "string",
      "endpoint": "string",
      "region": "string",
      "keyType": "string(enum)",
      "key": "string",
      "keySecret": "string",
      "caFile": "string",
      "uriType": "string(enum)",
      "path": "string"}
  },
  {
    "id" : integer,
    "repoType" : "posix",
    "posix" : {
      "id" : integer,
      "path" : "string"}
    },
  {
    "id" : integer,
    "repoType" : "dds",
    "dds": {
      "id" : integer,
      "path" : "string",
      "hostname": "string",
      "username": "string",
      "password": "string",
      "storageUnit": "string",
      "directory": "string",
      "writeBufferSize": integer,
      "readBufferSize": integer,
      "executablePath": "string"}
  }
]
```

JSON fields

cifs

Field	Description
id	An identifier of the repository configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command
repoType	A type of the data repository — `cifs`
cifs.id	An identifier of the cifs repository configuration
cifs.path	The directory where all necessary backup files (metadata files and data files) are stored

s3

Field	Description
id	An identifier of the repository configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command
repoType	A type of the data repository — `s3`
s3.id	An identifier of the s3 repository configuration

s3.path	A repository where backups and archived WAL segments are stored
s3.bucket	A bucket name in the S3 storage
s3.endpoint	An endpoint of the S3 storage
s3.region	A region of the S3 storage
s3.keyType	A key type of the S3 storage: - `shared` — using shared keys; - `auto` — retrieving temporary credentials automatically; - `web-id` — retrieving web identity credentials automatically.
s3.key	A key to access the S3 storage
s3.keySecret	A secret key to access the S3 storage
s3.caFile	A Certificate Authority (CA) PEM file to access the S3 storage
s3.uriType	An endpoint type (see `endpoint` above). Possible values: - `host` - `path`

posix

Field	Description
id	An identifier of the repository configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command
repoType	A type of the data repository — `posix`
posix.id	An identifier of the posix repository configuration
posix.path	The directory where all necessary backup files (metadata files and data files) are stored

dds

Field	Description
id	An identifier of the repository configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command
repoType	A type of the data repository — `dds`
dds.id	An identifier of the Data Domain Storage repository configuration
dds.path	The directory where all necessary backup files (metadata files and data files) are stored
dds.hostname	An IP address or name of the host that provides operations with DDBoost
dds.username	A name of the user who is granted permissions to work with DDBoost. The user is configured on the DDBoost side. This is neither operating system nor ADB user
dds.password	A password of the user who is granted permissions to work with DDBoost
dds.storageUnit	A name of the storage unit that is configured on the DDBoost side

dds.directory	A name of the directory in the DDBoost file system. That directory is used to store all backup files created when the `gpbackup` utility is run with the DDBoost plugin. In the */<storageUnit>/<directory>* folder, the plugin automatically creates all subdirectories corresponding to the creation date and time of each backup: */<storageUnit>/<directory>/YYYYMMDD/YYYYMMDDHHmmSS/*
dds.writeBufferSize	A buffer size for writing data to DDBoost (in bytes)
dds.readBufferSize	A buffer size for reading data from DDBoost (in bytes)
dds.executablePath	An absolute path to the plugin in the file system of ADB hosts (along with the plugin executable file name). The plugin should be installed on all cluster hosts in the same directory

Example

```
$ adbm backup repo-config ls --backup-format logical
```

Result:

```
[ {
  "id" : 1,
  "repoType" : "posix",
  "posix" : {
    "id" : 1,
    "path" : "/dasha"
  }
}, {
  "id" : 9,
  "repoType" : "s3",
  "s3" : {
    "id" : 1,
    "path" : "/test",
    "bucket" : "test",
    "endpoint" : "storage.yandexcloud.net",
    "region" : "ru-central1",
    "keyType" : "shared",
    "key" : "XXXXXXXXXXXXXXXXXXXX",
    "keySecret" : "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "caFile" : "test.pem",
    "uriType" : "host"
  }
} ]
```

backup repo-config delete

The `adbm backup repo-config delete` command allows you to delete a repository configuration with the specified identifier (previously created via the backup repo-config create command).

Syntax:

```
$ adbm backup repo-config delete \
--backup-format | -f <backupFormat> \
--repo-config-id | -rc <repoConfigId> \
[-a]
[--log-level <logLevel>]
[--log-path <logPath>]
[--help | -h]
```

Command options

Option	Description	Mandatory	Default

--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: - `binary | b` — binary backups; - `logical | l` — logical backups.	Yes	—
--repo-config-id | -rc	An identifier of the repository configuration that should be deleted. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command	Yes	—
-a	Defines whether to launch the command automatically (without user confirmation)	No	false
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains an identifier of the removed repository configuration:

```
Deletion of <repoConfigId> completed
```

Example

```
$ adbm backup repo-config delete --backup-format logical --repo-config-id 3
```

Result:

```
Are you sure you want to delete logical backup repository configuration? (y/n):y
Delete backup repository configuration in progress. Action id = 36e2b9d2-e8a3-4e5f-a0e3-ab1fad401f52. Performing "Start 'Delete
logical backup repository configuration' action".done
Deletion of 3 completed
```

backup config create

The `adbm backup config create` command creates a new backup configuration based on the passed options. You can use backup configurations to create backups via the backup create command.

Syntax:

```
$ adbm backup config create \
--backup-format | -f <backupFormat> \
--repo-config-id | -rc <repoConfigId> \
[--file <file>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

IMPORTANT

In the absence of the `--file` option, parameters of the created backup configuration are set to default values. See Input JSON file parameters below.

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: - `binary | b` — binary backups; - `logical | l` — logical backups.	Yes	—
--repo-config-id | -rc	An identifier of the repository configuration that should be used to create a backup configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command	Yes	—
--file	A path to the JSON configuration file with the backup configuration parameters. When this option is used, the configuration parameters are taken from the specified file. See Input JSON file parameters below	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Input JSON file parameters

Template:

```
{
  "compressionLevel": integer,
  "compressionType": "string(enum)",
  "copyQueueSize": integer,
  "jobs": integer,
  "pluginConfig": "string",
  "excludeSchema": [
    "string"
  ],
  "excludeTable": [
    "string"
  ],
  "excludeSchemaFile": "string",
  "excludeTableFile": "string",
  "includeSchema": [
    "string"
  ],
  "includeTable": [
    "string"
  ],
  "includeSchemaFile": "string",
  "includeTableFile": "string",
  "isDataOnly": boolean,
  "isDebug": boolean,
  "isIncremental": boolean,
  "isWithoutGlobals": boolean,
  "isLeafPartitionData": boolean,
  "isMetadataOnly": boolean,
  "isNoCompression": boolean,
```

```
  "isNoInherits": boolean,
  "isNoHistory": boolean,
  "isSingleDataFile": boolean,
  "isWithStats": boolean,
  "isMultiFormatBackups": boolean
}
```

JSON fields

Field	Description	Default
compressionLevel	Specifies the compression level used to compress data files. Allowed values: `1` — `9`. If the option value is not set, `gpbackup` uses the `1` level. Note that `gpbackup` uses compression by default	1
compressionType	Specifies the compression type used to compress data files. Possible values: ▪ `gzip` ▪ `zstd` If the option value is not set, `gpbackup` uses the `gzip` method by default	gzip
copyQueueSize	Specifies the number of `COPY` commands that `gpbackup` should enqueue when backing up. This option optimizes backup performance by reducing the amount of time spent for initializing `COPY` commands. If you do not set this option to `2` or greater, `gpbackup` enqueues one `COPY` command at a time. Requires `isSingleDataFile=true` and cannot be used with the `jobs` option	—
jobs	Specifies the number of jobs to run in parallel when backing up tables. By default, one connection is used. Increasing this number can improve the speed of backing up data. You cannot use this option in combination with the `isMetadataOnly`, `isSingleDataFile`, or `pluginConfig` options	1
pluginConfig	The location of the `gpbackup` plugin configuration file (YAML-formatted text file). If you specify the `pluginConfig` option when you back up a database, you should specify this option when you restore the database from that backup as well. Cannot be combined with `backupDir`	—
excludeSchema	An array of schemas to exclude from the backup. You cannot combine this option with other schema or table filtering options	{}
excludeTable	An array of tables to exclude from the backup. Each table should be in the format `<schema_name>.<table_name>`. If you specify a leaf partition name, `gpbackup` ignores it. The leaf partition is not excluded. You cannot combine this option with other schema or table filtering options	{}
excludeSchemaFile	Specifies a text file that contains a list of schemas to exclude from the backup. Each line in the text file should define a single schema. The file should not include trailing lines. If a schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot combine this option with other schema or table filtering options	—
excludeTableFile	Specifies a text file that contains a list of tables to exclude from the backup. Each line in the text file should define a single table using the format `<schema_name>.<table_name>`. The file should not include trailing lines. If a table or schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. If you specify a leaf partition name, `gpbackup` ignores it. The leaf partition is not excluded. You cannot combine this option with other schema or table filtering options	—

includeSchema	An array of schemas to include in the backup. Other schemas are omitted from the backup. You cannot combine this option with other schema or table filtering options	{}
includeTable	An array of tables to include in the backup. Other tables are omitted from the backup. Each table should be in the format `<schema_name>.<table_name>`. You can also specify the qualified name of a sequence, a view, or a materialized view. In that case, you should also explicitly specify the dependent objects that are required. Optionally, if you use the `isLeafPartitionData=true` option, you can specify a table leaf partition name in place of the table name to include only specific leaf partitions in a backup. You cannot combine this option with other schema or table filtering options	{}
includeSchemaFile	Specifies a text file that contains a list of schemas to include in the backup. Other schemas are omitted from the backup. Each line in the text file should define a single schema. The file should not include trailing lines. If a schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot combine this option with other schema or table filtering options	—
includeTableFile	Specifies a text file that contains a list of tables to include in the backup. Other tables are omitted from the backup. Each line in the text file should define a single table using the format `<schema_name>.<table_name>`. The file should not include trailing lines. If a table or schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You can also specify the qualified name of a sequence, a view, or a materialized view. In that case, you should also explicitly specify the dependent objects that are required. Optionally, if you use the `isLeafPartitionData=true` option, you can specify a table leaf partition name in place of the table name to include only specific leaf partitions in a backup. You cannot combine this option with other schema or table filtering options	—
isDataOnly	Defines whether to back up only the table data into CSV files — without metadata files needed to recreate the tables and other database objects	false
isDebug	Defines whether to display verbose debug messages during the backup	false
isIncremental	Defines whether to make an incremental backup	false
isWithoutGlobals	Defines whether to omit the global system objects during the backup	false
isLeafPartitionData	For partitioned tables, defines whether to create one data file per leaf partition instead of one data file for the entire table	false
isMetadataOnly	Defines whether to create only the metadata files (DDL) needed to recreate the database objects without backing up the actual table data	false
isNoCompression	Defines whether not to compress the CSV files with table data	false
isNoInherits	When invoked, only the metadata of the table itself is backed up, ignoring any inheritance relationships with other tables that would normally cause those tables to also be included in the backup set. Only works when invoked with either the `includeTable` option or the `includeTableFile` option	false
isNoHistory	When invoked, `gpbackup` does not write backup run metadata to the history database	false
isSingleDataFile	Defines whether to create a single data file on each segment host for all tables backed up on that segment. By default, `gpbackup` creates one CSV file for each table that is backed up on the segment	false

isWithStats	Defines whether to include query plan statistics in the backup set	false
isMultiFormatBackups	Defines whether to launch binary and logical backups simultaneously	false

NOTE

All fields in the JSON file are optional.

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains an identifier of the created backup configuration:

```
Backup configuration <backupConfigId> created
```

Example

Without --file

1. Run the following command to create a backup configuration:

   ```
   $ adbm backup config create --repo-config-id 1 --backup-format logical
   ```

   Result:

   ```
   Create backup configuration in progress. Action id = 745a2a51-1593-4511-a98f-69260fc22227. Performing "Start 'Create
   logical backup configuration' action".done
   Backup configuration 2 created
   ```

2. Check that the default options were applied to the backup configuration, since you did not pass the `--file` option. To do this, run the backup config ls command:

   ```
   $ adbm backup config ls --backup-format logical --config-id 2
   ```

   Result:

   ```
   {
     "id" : 2,
     "repoConfigId" : 1,
     "configParams" : {
       "excludeSchema" : [ ],
       "excludeTable" : [ ],
       "includeSchema" : [ ],
       "includeTable" : [ ],
       "isDataOnly" : false,
       "isDebug" : false,
       "isIncremental" : false,
       "isWithoutGlobals" : false,
       "isLeafPartitionData" : false,
       "isMetadataOnly" : false,
       "isNoCompression" : false,
       "isNoInherits" : false,
       "isNoHistory" : false,
       "isSingleDataFile" : false,
       "isWithStats" : false,
       "isMultiFormatBackups" : false
     }
   }
   ```

With –file

1. Create a JSON file with configuration parameters:

```
$ vi backup-conf.json
```

```
{
  "compressionLevel": 1,
  "compressionType": "gzip",
  "includeSchema": [
    "public"
  ]
}
```

2. Run the following command to create a backup configuration:

```
$ adbm backup config create --repo-config-id 1 --backup-format logical --file backup-conf.json
```

Result:

```
Create backup configuration in progress. Action id = 073fdb77-4a11-43fb-b643-905b56152643. Performing "Start 'Create
logical backup configuration' action".done
Backup configuration 3 created
```

3. Check that the options you passed were applied to the backup configuration — via the backup config ls command:

```
$ adbm backup config ls --backup-format logical --config-id 3
```

Result:

```
{
  "id" : 3,
  "repoConfigId" : 1,
  "configParams" : {
    "compressionLevel" : 1,
    "compressionType" : "gzip",
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ "public" ],
    "includeTable" : [ ],
    "isDataOnly" : false,
    "isDebug" : false,
    "isIncremental" : false,
    "isWithoutGlobals" : false,
    "isLeafPartitionData" : false,
    "isMetadataOnly" : false,
    "isNoCompression" : false,
    "isNoInherits" : false,
    "isNoHistory" : false,
    "isSingleDataFile" : false,
    "isWithStats" : false,
    "isMultiFormatBackups" : false
  }
}
```

backup config ls

The `adbm backup config ls` command lists existing backup configurations (previously created via the backup config create command).

Syntax:

```
$ adbm backup config ls \
--backup-format | -f <backupFormat> \
[--config-id | -c <backupConfigId>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: ■ `binary | b` — binary backups; ■ `logical | l` — logical backups.	Yes	—
--config-id | -c	An identifier of the backup configuration. If set, the command returns information only for the specified configuration. It is returned by the backup config create command and can be obtained with the backup config ls command	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

The command returns a JSON file with an array of existing backup configurations.

Template:

```
[ {
  "id" : integer,
  "repoConfigId" : integer,
  "configParams" : {
    "compressionLevel": integer,
    "compressionType": "string(enum)",
    "copyQueueSize": integer,
    "jobs": integer,
    "pluginConfig": "string",
    "excludeSchema": [
      "string"
    ],
    "excludeTable": [
      "string"
    ],
    "excludeSchemaFile": "string",
    "excludeTableFile": "string",
    "includeSchema": [
      "string"
    ],
    "includeTable": [
      "string"
    ],
    "includeSchemaFile": "string",
    "includeTableFile": "string",
    "isDataOnly": boolean,
    "isDebug": boolean,
    "isIncremental": boolean,
    "isWithoutGlobals": boolean,
```

```
    "isLeafPartitionData": boolean,
    "isMetadataOnly": boolean,
    "isNoCompression": boolean,
    "isNoInherits": boolean,
    "isNoHistory": boolean,
    "isSingleDataFile": boolean,
    "isWithStats": boolean,
    "isMultiFormatBackups": boolean
  }
} ]
```

JSON fields

Field	Description
id	An identifier of the backup configuration. It is returned by the backup config create command and can be obtained with the backup config ls command
repoConfigId	An identifier of the repository configuration. It is returned by the backup repo-config create command and can be obtained with the backup repo-config ls command
configParams	Options that were used to create the backup configuration. For information on each available option, see Input JSON file parameters for the `backup config create` command. All parameters are displayed when the command is used for the specific backup configuration (`--config-id` is set). Otherwise, boolean parameters are omitted. String parameters with empty values are omitted in any case

Example

Without --config-id

```
$ adbm backup config ls --backup-format logical
```

Result:

```
[ {
  "id" : 2,
  "repoConfigId" : 1,
  "configParams" : {
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ ],
    "includeTable" : [ ]
  }
}, {
  "id" : 3,
  "repoConfigId" : 1,
  "configParams" : {
    "compressionLevel" : 1,
    "compressionType" : "gzip",
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ "public" ],
    "includeTable" : [ ]
  }
} ]
```

With --config-id

```
$ adbm backup config ls --backup-format logical --config-id 3
```

Result:

```
{
  "id" : 3,
  "repoConfigId" : 1,
```

```
  "configParams" : {
    "compressionLevel" : 1,
    "compressionType" : "gzip",
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ "public" ],
    "includeTable" : [ ],
    "isDataOnly" : false,
    "isDebug" : false,
    "isIncremental" : false,
    "isWithoutGlobals" : false,
    "isLeafPartitionData" : false,
    "isMetadataOnly" : false,
    "isNoCompression" : false,
    "isNoInherits" : false,
    "isNoHistory" : false,
    "isSingleDataFile" : false,
    "isWithStats" : false,
    "isMultiFormatBackups" : false
  }
}
```

backup config delete

The `adbm backup config delete` command allows you to delete a backup configuration with the specified identifier (previously created via the backup config create command).

Syntax:

```
$ adbm backup config delete \
--backup-format | -f <backupFormat> \
--config-id | -c <backupConfigId> \
[-a]
[--log-level <logLevel>]
[--log-path <logPath>]
[--help | -h]
```

IMPORTANT

You cannot delete the most current backup configuration. When attempting to do this, you will get the following error:

```
The last logical backup config cannot be deleted
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: - `binary | b` — binary backups; - `logical | l` — logical backups.	Yes	—
--config-id | -c	An identifier of the backup configuration that should be deleted. It is returned by the backup config create command and can be obtained with the backup config ls command	Yes	—
-a	Defines whether to launch the command automatically (without user confirmation)	No	false
--log-level	A logging level for the command	No	info

--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains an identifier of the removed backup configuration:

```
Deletion of <backupConfigId> completed
```

Example

```
$ adbm backup config delete --backup-format logical --config-id 1
```

Result:

```
Are you sure you want to delete logical backup configuration? (y/n):y
Delete backup configuration in progress. Action id = 89034934-3500-4d1c-93b8-1d91b4ac1c17. Performing "Start 'Delete logical backup configuration' action".done
Deletion of 1 completed
```

backup create

The `adbm backup create` command creates a database backup based on the passed options. For logical backups, the gpbackup utility is used.

Syntax:

```
$ adbm backup create \
--backup-format | -f <backupFormat> \
--dbname | --database | -d <dbName> \
[--config-id |  -c <backupConfigId>]
[--file <file>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

IMPORTANT

- If both `--file` and `--config-id` options are absent, configuration parameters are taken from the latest (most current) backup configuration.
- If both `--file` and `--config-id` options are filled in, parameters from the `--file` option overwrite values of similar parameters from the backup configuration with the specified identifier `--config-id`. Parameters that are not explicitly defined in the `--file` option are taken from the specified backup configuration.

Command options

Option	Description	Mandatory	Default

--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: ▪ `binary | b` — binary backups; ▪ `logical | l` — logical backups.	Yes	—
--dbname | --database | -d	A name of the ADB database that should be backed up	Yes	—
--config-id | -c	An identifier of the backup configuration that should be used for the current backup. It is returned by the backup config create command and can be obtained with the backup config ls command	No	—
--file	A path to the JSON configuration file with the backup configuration parameters. See Input JSON file parameters below	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Input JSON file parameters

Template:

```
{
  "compressionLevel": integer,
  "compressionType": "string(enum)",
  "copyQueueSize": integer,
  "jobs": integer,
  "pluginConfig": "string",
  "excludeSchema": [
    "string"
  ],
  "excludeTable": [
    "string"
  ],
  "excludeSchemaFile": "string",
  "excludeTableFile": "string",
  "includeSchema": [
    "string"
  ],
  "includeTable": [
    "string"
  ],
  "includeSchemaFile": "string",
  "includeTableFile": "string",
  "isDataOnly": boolean,
  "isDebug": boolean,
  "isIncremental": boolean,
  "isWithoutGlobals": boolean,
  "isLeafPartitionData": boolean,
  "isMetadataOnly": boolean,
  "isNoCompression": boolean,
  "isNoInherits": boolean,
  "isNoHistory": boolean,
  "isSingleDataFile": boolean,
  "isWithStats": boolean,
  "isMultiFormatBackups": boolean
}
```

JSON fields

Field	Description	Default
compressionLevel	Specifies the compression level used to compress data files. Allowed values: `1` — `9`. If the option value is not set, `gpbackup` uses the `1` level. Note that `gpbackup` uses compression by default	1
compressionType	Specifies the compression type used to compress data files. Possible values: ▪ `gzip` ▪ `zstd` If the option value is not set, `gpbackup` uses the `gzip` method by default	gzip
copyQueueSize	Specifies the number of `COPY` commands that `gpbackup` should enqueue when backing up. This option optimizes backup performance by reducing the amount of time spent for initializing `COPY` commands. If you do not set this option to `2` or greater, `gpbackup` enqueues one `COPY` command at a time. Requires `isSingleDataFile=true` and cannot be used with the `jobs` option	—
jobs	Specifies the number of jobs to run in parallel when backing up tables. By default, one connection is used. Increasing this number can improve the speed of backing up data. You cannot use this option in combination with the `isMetadataOnly`, `isSingleDataFile`, or `pluginConfig` options	1
pluginConfig	The location of the `gpbackup` plugin configuration file (YAML-formatted text file). If you specify the `pluginConfig` option when you back up a database, you should specify this option when you restore the database from that backup as well. Cannot be combined with `backupDir`	—
excludeSchema	An array of schemas to exclude from the backup. You cannot combine this option with other schema or table filtering options	{}
excludeTable	An array of tables to exclude from the backup. Each table should be in the format `<schema_name>.<table_name>`. If you specify a leaf partition name, `gpbackup` ignores it. The leaf partition is not excluded. You cannot combine this option with other schema or table filtering options	{}
excludeSchemaFile	Specifies a text file that contains a list of schemas to exclude from the backup. Each line in the text file should define a single schema. The file should not include trailing lines. If a schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot combine this option with other schema or table filtering options	—
excludeTableFile	Specifies a text file that contains a list of tables to exclude from the backup. Each line in the text file should define a single table using the format `<schema_name>.<table_name>`. The file should not include trailing lines. If a table or schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. If you specify a leaf partition name, `gpbackup` ignores it. The leaf partition is not excluded. You cannot combine this option with other schema or table filtering options	—
includeSchema	An array of schemas to include in the backup. Other schemas are omitted from the backup. You cannot combine this option with other schema or table filtering options	{}
includeTable	An array of tables to include in the backup. Other tables are omitted from the backup. Each table should be in the format `<schema_name>.<table_name>`. You can also specify the qualified name of a sequence, a view, or a materialized view. In that case, you should also explicitly specify the dependent objects that are required.	{}

	Optionally, if you use the `isLeafPartitionData=true` option, you can specify a table leaf partition name in place of the table name to include only specific leaf partitions in a backup. You cannot combine this option with other schema or table filtering options	
includeSchemaFile	Specifies a text file that contains a list of schemas to include in the backup. Other schemas are omitted from the backup. Each line in the text file should define a single schema. The file should not include trailing lines. If a schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot combine this option with other schema or table filtering options	—
includeTableFile	Specifies a text file that contains a list of tables to include in the backup. Other tables are omitted from the backup. Each line in the text file should define a single table using the format `<schema_name>.<table_name>`. The file should not include trailing lines. If a table or schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You can also specify the qualified name of a sequence, a view, or a materialized view. In that case, you should also explicitly specify the dependent objects that are required. Optionally, if you use the `isLeafPartitionData=true` option, you can specify a table leaf partition name in place of the table name to include only specific leaf partitions in a backup. You cannot combine this option with other schema or table filtering options	—
isDataOnly	Defines whether to back up only the table data into CSV files — without metadata files needed to recreate the tables and other database objects	false
isDebug	Defines whether to display verbose debug messages during the backup	false
isIncremental	Defines whether to make an incremental backup	false
isWithoutGlobals	Defines whether to omit the global system objects during the backup	false
isLeafPartitionData	For partitioned tables, defines whether to create one data file per leaf partition instead of one data file for the entire table	false
isMetadataOnly	Defines whether to create only the metadata files (DDL) needed to recreate the database objects without backing up the actual table data	false
isNoCompression	Defines whether not to compress the CSV files with table data	false
isNoInherits	When invoked, only the metadata of the table itself is backed up, ignoring any inheritance relationships with other tables that would normally cause those tables to also be included in the backup set. Only works when invoked with either the `includeTable` option or the `includeTableFile` option	false
isNoHistory	When invoked, `gpbackup` does not write backup run metadata to the history database	false
isSingleDataFile	Defines whether to create a single data file on each segment host for all tables backed up on that segment. By default, `gpbackup` creates one CSV file for each table that is backed up on the segment	false
isWithStats	Defines whether to include query plan statistics in the backup set	false
isMultiFormatBackups	Defines whether to launch binary and logical backups simultaneously	false

NOTE

All fields in the JSON file are optional.

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains a name of the created backup:

```
Logical backup <backupName> created
```

Example

Without --file

1. Run the following command to create a backup:

   ```
   $ adbm backup create --dbname test --backup-format logical --config-id 2
   ```

   Result:

   ```
   Create backup in progress. Action id = 5b9ffc21-1578-4e9d-a152-066cf2c02102. Performing "Start 'Create logical backup' action".done
   Logical backup 20240726105135 created
   ```

2. Check that options from the specified backup configuration (`--config-id=2`) were applied to the backup, since the `--file` option was not used. To do this, run the backup ls command. In the example below, all options have default values, since `--file` was not used when creating the backup configuration as well:

   ```
   $ adbm backup ls --backup-format logical --backup-name 20240726105135
   ```

   Result:

   ```
   {
     "id" : 9,
     "name" : "20240726105135",
     "configId" : 2,
     "clusterId" : 1,
     "startTime" : 1721991095000,
     "updateTime" : 1721991110827,
     "status" : "done",
     "type" : "full",
     "version" : "1.30.3",
     "database" : "test",
     "configParams" : {
       "excludeSchema" : [ ],
       "excludeTable" : [ ],
       "includeSchema" : [ ],
       "includeTable" : [ ],
       "isDataOnly" : false,
       "isDebug" : false,
       "isIncremental" : false,
       "isWithoutGlobals" : false,
       "isLeafPartitionData" : false,
       "isMetadataOnly" : false,
       "isNoCompression" : false,
       "isNoInherits" : false,
       "isNoHistory" : false,
       "isSingleDataFile" : false,
       "isWithStats" : false,
       "isMultiFormatBackups" : false
     }
   }
   ```

With --file

1. Create a JSON file with configuration parameters:

   ```
   $ vi backup.json
   ```

   ```
   {
     "compressionLevel": 2,
     "compressionType": "zstd",
     "isMetadataOnly": true
   }
   ```

2. Run the following command to create a backup:

   ```
   $ adbm backup create --dbname test --backup-format logical --config-id 3 --file backup.json
   ```

   Result:

   ```
   Create backup in progress. Action id = 9e95c4c2-f9f9-46ea-a0b5-cad35d8961f8. Performing "Start 'Create logical backup'
   action".done
   Logical backup 20240726095356 created
   ```

3. Check that the options you passed were applied to the backup — via the backup ls command:

   ```
   $ adbm backup ls --backup-format logical --backup-name 20240726095356
   ```

   Result:

   ```
   {
     "id" : 3,
     "name" : "20240726095356",
     "configId" : 3,
     "clusterId" : 1,
     "startTime" : 1721987636000,
     "updateTime" : 1721987651035,
     "status" : "done",
     "type" : "full",
     "version" : "1.30.3",
     "database" : "test",
     "configParams" : {
       "compressionLevel" : 2,
       "compressionType" : "zstd",
       "excludeSchema" : [ ],
       "excludeTable" : [ ],
       "includeSchema" : [ "public" ],
       "includeTable" : [ ],
       "isDataOnly" : false,
       "isDebug" : false,
       "isIncremental" : false,
       "isWithoutGlobals" : false,
       "isLeafPartitionData" : false,
       "isMetadataOnly" : true,
       "isNoCompression" : false,
       "isNoInherits" : false,
       "isNoHistory" : false,
       "isSingleDataFile" : false,
       "isWithStats" : false,
       "isMultiFormatBackups" : false
     }
   }
   ```

backup restore

The `adbm backup restore` command restores data from the specified backup (previously created via the backup create command) based on the passed options. For logical backups, the gprestore utility is used.

Syntax:

```
$ adbm backup restore \
--backup-format | -f <backupFormat> \
--backup-name <backupName> \
[--file <file>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: - `binary | b` — binary backups; - `logical | l` — logical backups.	Yes	—
--backup-name	- **For binary backups** — a name of the restore point. - **For logical backups** — a name of the logical backup in the `YYYYMMDDHHMMSS` format. It is returned by the backup create command and can be obtained with the backup ls command.	Yes	—
--file	A path to the JSON configuration file with the restore operation parameters. When this option is used, the configuration parameters are taken from the specified file. See Input JSON file parameters below	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Input JSON file parameters

Template:

```
{
  "copyQueueSize": integer,
  "jobs": integer,
  "pluginConfig": "string",
  "redirectSchema": "string",
  "redirectDb": "string",
  "excludeSchema": [
    "string"
  ],
  "excludeTable": [
    "string"
  ],
  "excludeSchemaFile": "string",
  "excludeTableFile": "string",
  "includeSchema": [
    "string"
  ],
  "includeTable": [
    "string"
  ],
  "includeSchemaFile": "string",
  "includeTableFile": "string",
  "isDebug": boolean,
  "isDataOnly": boolean,
```

```
  "isIncremental": boolean,
  "isMetadataOnly": boolean,
  "isWithStats": boolean,
  "isCreateDb": boolean,
  "isTruncateTable": boolean,
  "isResizeCluster": boolean,
  "isOnErrorContinue": boolean,
  "isWithGlobals": boolean,
  "isRunAnalyze": boolean
}
```

JSON fields

Field	Description	Default
copyQueueSize	Specifies the number of `COPY` commands that `gprestore` should enqueue when restoring a backup set. This option optimizes restore performance by reducing the amount of time spent for initializing `COPY` commands. If you do not set this option to `2` or greater, `gprestore` enqueues one `COPY` command at a time. Can be used only if the backup was taken with the `isSingleDataFile=true` option	—
jobs	Specifies the number of parallel connections to use when restoring table data and metadata. By default, one connection is used. Increasing this number can improve the speed of restoring data. If the backup was created with the `isSingleDataFile=true` option to combine table backups into a single file per segment, you cannot set the `jobs` to a value higher than 1 to perform a parallel restore operation	1
pluginConfig	The location of the `gpbackup` plugin configuration file (YAML-formatted text file). If you specify the `pluginConfig` option when you back up a database, you should specify this option when you restore the database from that backup as well. Cannot be combined with `backupDir`	—
redirectSchema	Defines whether to restore data in the specified schema instead of the original schemas. The selected schema should exist. This option should be used with any option that includes tables or schemas: `includeTable`, `includeTableFile`, `includeSchema`, or `includeSchemaFile`. You cannot use this option with an option that excludes schemas or tables such as `excludeSchema`, `excludeSchemaFile`, `excludeTable`, or `excludeTableFile`. You can use this option with the `isMetadataOnly` or `isDataOnly` options	—
redirectDb	Defines whether to restore data in the specified database instead of the database that was backed up	—
excludeSchema	Specifies an array of database schemas to exclude from the restore operation. You cannot combine this option with other schema or table filtering options	{}
excludeTable	Specifies an array of tables to exclude from the restore operation. Each table should be in the format `<schema_name>.<table_name>`. You cannot specify a leaf partition of a partitioned table. You cannot combine this option with other schema or table filtering options	{}
excludeSchemaFile	Specifies a text file that contains a list of database schemas to exclude from the restore operation. Each line in the text file should define a single schema. The file should not include trailing lines. If a schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot combine this option with other schema or table filtering options	—
excludeTableFile	Specifies a text file that contains a list of tables to exclude from the restore operation.	—

	Each line in the text file should define a single table using the format `<schema_name>.<table_name>`. The file should not include trailing lines. If a table or schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot specify a leaf partition of a partitioned table. You cannot combine this option with other schema or table filtering options	
includeSchema	Specifies an array of database schemas to restore. Other schemas are omitted from the restore operation. You cannot combine this option with other schema or table filtering options	{}
includeTable	Specifies an array of tables to restore. Other tables are omitted from the restore operation. Each table should be in the format `<schema_name>.<table_name>`. You cannot specify a leaf partition of a partitioned table. You can also specify the qualified name of a sequence, a view, or a materialized view. In that case, you should also explicitly specify the dependent objects that are required. You cannot combine this option with other schema or table filtering options	{}
includeSchemaFile	Specifies a text file that contains a list of database schemas to restore. Other schemas are omitted from the restore operation. Each line in the text file should define a single schema. The file should not include trailing lines. If a schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot combine this option with other schema or table filtering options	—
includeTableFile	Specifies a text file that contains a list of tables to restore. Other tables are omitted from the restore operation. Each line in the text file should define a single table using the format `<schema_name>.<table_name>`. The file should not include trailing lines. If a table or schema name uses any character other than a lowercase letter, number, or an underscore character, include that name in double quotes. You cannot specify a leaf partition of a partitioned table. You can also specify the qualified name of a sequence, a view, or a materialized view. In that case, you should also explicitly specify the dependent objects that are required. You cannot combine this option with other schema or table filtering options	—
isDataOnly	Allows you to restore table data from a backup without creating the database tables and other objects. This option assumes the database objects exist in the target database	false
isDebug	Defines whether to display verbose and debug log messages during a restore operation	false
isIncremental	Allows you to restore only the table data in the incremental backup specified by the `--backup-name` option. Table data is not restored from previous incremental backups in the backup set. Requires the `isDataOnly=true` option. This is a beta feature and is not recommended for production environments	false
isMetadataOnly	Allows you to create database tables and other objects from a backup without restoring the table data. This option assumes the database objects do not exist in the target database	false
isWithStats	Defines whether to restore query plan statistics from the backup set. If the selected backup was not created with the `isWithStats=true` option, an error is returned. Cannot be combined with `isRunAnalyze=true`	false
isCreateDb	Defines whether to create the database before restoring the database object metadata. The database is created by cloning the empty standard system database `template0`	false

isTruncateTable	Defines whether to truncate data from a set of tables before restoring the table data from a backup. This option lets you replace table data with data from a backup. Otherwise, table data might be duplicated. You should specify the set of tables with either the option `includeTable` or `includeTableFile`. You should also specify `isDataOnly` to restore table data without creating the tables. Can be combined with the `redirectDb` option. Cannot be combined with `redirectSchema`	false
isResizeCluster	Enables restoring data to a cluster that has a different number of segments than the cluster from which the data was backed up	false
isOnErrorContinue	Defines whether to continue the restore operation if an SQL error occurs when creating database metadata (such as tables, roles, or functions) or restoring data. If another type of error occurs, the utility exits. The default is to exit on the first error. If the option is specified, the `jobs` and `copyQueueSize` options are disabled for metadata restore, as metadata cannot be safely restored in parallel in the presence of errors. Additionally, when the option is set, the `gprestore` utility writes text files to the backup directory that contain a list of tables that generated SQL errors	false
isWithGlobals	Defines whether to restore ADB system objects in the backup set, in addition to database objects	false
isRunAnalyze	Defines whether to run `ANALYZE` on the tables that are restored. For a partitioned table, `ANALYZE` is run on the root partitioned table. If the backup being restored used `isLeafPartitionData=true`, `gprestore` runs `ANALYZE` only on the individual leaf partitions that are restored, not the root partitioned table. If `isWithStats=true` was specified for the backup which data is restored, those statistics are ignored. Cannot be combined with `isWithStats=true`. Depending on the tables being restored, running `ANALYZE` on restored tables may increase the duration of the restore operation. As an alternative, you can run the `ANALYZE` command on the tables after the tables are restored	false

NOTE

All fields in the JSON file are optional.

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains a name of the backup, which data is restored:

```
Restore of <backupName> completed
```

Example

Without --file

```
$ adbm backup restore --backup-name 20240715151947 --backup-format logical
```

Result:

```
Create backup restore in progress. Action id = 9d9230ec-758e-4f3f-821e-74aa54ce9f80. Performing "Start 'Create logical
```

```
restore' action".done
Restore of 20240715151947 completed
```

With --file

1. Create a JSON file with configuration parameters:

   ```
   $ vi restore.json
   ```

   ```
   {
     "redirectDb": "test2",
     "isCreateDb": true,
     "isRunAnalyze": true
   }
   ```

2. Run the following command to restore data from the specified backup:

   ```
   $ adbm backup restore --backup-name 20240726121436 --backup-format logical --file restore.json
   ```

   Result:

   ```
   Create backup restore in progress. Action id = 618badbe-1b9d-4d90-8812-7112c99517b8. Performing "Start 'Create logical
   restore' action".done
   Restore of 20240726121436 completed
   ```

backup ls

The `adbm backup ls` command lists existing backups (previously created via the backup create command).

Syntax:

```
$ adbm backup ls \
--backup-format | -f <backupFormat> \
[--database | -d <database>] \
[--backup-type <backupType>] \
[--backup-status <backupStatus>] \
[--backup-name <backupName>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: ▪ `binary | b` — binary backups; ▪ `logical | l` — logical backups.	Yes	—
--database | -d	A database name	No	—
--backup-type	A backup type. Possible values for logical backups: ▪ `full` — full backup; ▪ `incremental` — incremental backup.	No	—
--backup-status	A backup status. Possible values for logical backups: ▪ `init` ▪ `running`	No	—

	- `failed` - `done`		
--backup-name	A backup name. If set, the command displays information only for the specified backup. It is returned by the backup create command and can be obtained with the backup ls command	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

The command returns a JSON file with an array of existing backups.

Template:

```
[ {
  "id" : integer,
  "name" : "string",
  "configId" : integer,
  "clusterId" : integer,
  "startTime" : timestamp,
  "updateTime" : timestamp,
  "status" : "string(enum)",
  "type" : "string(enum)",
  "version" : "string",
  "database" : "string",
  "configParams" : {
    "compressionLevel": integer,
    "compressionType": "string(enum)",
    "copyQueueSize": integer,
    "jobs": integer,
    "pluginConfig": "string",
    "excludeSchema": [
      "string"
    ],
    "excludeTable": [
      "string"
    ],
    "excludeSchemaFile": "string",
    "excludeTableFile": "string",
    "includeSchema": [
      "string"
    ],
    "includeTable": [
      "string"
    ],
    "includeSchemaFile": "string",
    "includeTableFile": "string",
    "isDataOnly": boolean,
    "isDebug": boolean,
    "isIncremental": boolean,
    "isWithoutGlobals": boolean,
    "isLeafPartitionData": boolean,
    "isMetadataOnly": boolean,
    "isNoCompression": boolean,
    "isNoInherits": boolean,
    "isNoHistory": boolean,
    "isSingleDataFile": boolean,
    "isWithStats": boolean,
    "isMultiFormatBackups": boolean
  }
} ]
```

JSON fields

Field	Description
id	A backup identifier
name	A backup name. It is returned by the backup create command and can be obtained with the backup ls command
configId	A backup configuration identifier. It is returned by the backup config create command and can be obtained with the backup config ls command
clusterId	A cluster identifier
startTime	Record creation time (timestamp of the backup start)
updateTime	Record modification time (timestamp of the backup status update)
status	A backup status. Possible values for logical backups: - `init` - `running` - `failed` - `done`
type	A backup type. Possible values for logical backups: - `full` — full backup; - `incremental` — incremental backup.
version	The version of the `gpbackup` utility that was used to create a backup
database	A database name
configParams	Configuration options that were used for the backup. For information on each available option, see Input JSON file parameters for the `backup create` command. All parameters are displayed when the command is used for the specific backup (`--backup-name` is set). Otherwise, boolean parameters are omitted. String parameters with empty values are omitted in any case

Example

Without --backup-name

```
$ adbm backup ls --backup-format logical
```

Result:

```
[ {
  "id" : 1,
  "name" : "20240715151947",
  "configId" : 2,
  "clusterId" : 1,
  "startTime" : 1721056787000,
  "updateTime" : 1721056792351,
  "status" : "done",
  "type" : "full",
  "version" : "1.30.3",
  "database" : "test",
  "configParams" : {
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
```

```
    "includeSchema" : [ ],
    "includeTable" : [ ]
  }
}, {
  "id" : 2,
  "name" : "20240726095046",
  "configId" : 3,
  "clusterId" : 1,
  "startTime" : 1721987446000,
  "updateTime" : 1721987462071,
  "status" : "done",
  "type" : "full",
  "version" : "1.30.3",
  "database" : "test",
  "configParams" : {
    "compressionLevel" : 1,
    "compressionType" : "gzip",
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ "public" ],
    "includeTable" : [ ]
  }
}, {
  "id" : 3,
  "name" : "20240726095356",
  "configId" : 3,
  "clusterId" : 1,
  "startTime" : 1721987636000,
  "updateTime" : 1721987651035,
  "status" : "done",
  "type" : "full",
  "version" : "1.30.3",
  "database" : "test",
  "configParams" : {
    "compressionLevel" : 2,
    "compressionType" : "zstd",
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ "public" ],
    "includeTable" : [ ]
  }
} ]
```

With --backup-name

```
$ adbm backup ls --backup-format logical --backup-name 20240726095356
```

Result:

```
{
  "id" : 3,
  "name" : "20240726095356",
  "configId" : 3,
  "clusterId" : 1,
  "startTime" : 1721987636000,
  "updateTime" : 1721987651035,
  "status" : "done",
  "type" : "full",
  "version" : "1.30.3",
  "database" : "test",
  "configParams" : {
    "compressionLevel" : 2,
    "compressionType" : "zstd",
    "excludeSchema" : [ ],
    "excludeTable" : [ ],
    "includeSchema" : [ "public" ],
    "includeTable" : [ ],
    "isDataOnly" : false,
    "isDebug" : false,
    "isIncremental" : false,
    "isWithoutGlobals" : false,
    "isLeafPartitionData" : false,
    "isMetadataOnly" : true,
    "isNoCompression" : false,
```

```
    "isNoInherits" : false,
    "isNoHistory" : false,
    "isSingleDataFile" : false,
    "isWithStats" : false,
    "isMultiFormatBackups" : false
  }
}
```

backup report

The `adbm backup report` command displays a report for the specified backup (previously created via the backup create command).

Syntax:

```
$ adbm backup report \
--backup-format | -f <backupFormat> \
--backup-name <backupName> \
[--output <outputFormat>] \
[--log-level <logLevel>] \
[--log-path <logPath>] \
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: - `binary | b` — binary backups; - `logical | l` — logical backups.	Yes	—
--backup-name	- **For binary backups** — a name of the restore point. - **For logical backups** — a name of the logical backup in the `YYYYMMDDHHMMSS` format. It is returned by the backup create command and can be obtained with the backup ls command.	Yes	—
--output	The command output format. Possible values: - `text` - `json`	No	text
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

The command returns a detailed report for the specified backup (in the text or JSON form depending on the `--output` option value). This report contains the ADB and `gpbackup` versions; command options used to create the selected backup; the backup start and end timestamps, and other useful information.

Example

text

```
$ adbm backup report --backup-name 20241206091238 --backup-format logical --output text
```

Result:

```
Greenplum Database Backup Report

timestamp key:         20241206091238
gpdb version:          6.27.1_arenadata59 build 3956+gitc4500d63cbe
gpbackup version:      1.30.5

database name:         test
command line:          gpbackup --dbname test --backup-dir /dasha --compression-level 2 --compression-type zstd --verbose
compression:           zstd
plugin executable:     None
backup section:        All Sections
object filtering:      None
includes statistics:   No
data file format:      Multiple Data Files Per Segment
incremental:           False

start time:            Fri Dec 06 2024 09:12:38
end time:              Fri Dec 06 2024 09:12:40
duration:              0:00:02

backup status:         Success

database size:         167 MB
segment count:         8

count of database objects in backup:
aggregates                   0
casts                        0
collations                   0
constraints                  4
conversions                  0
default privileges           0
database gucs                0
event triggers               0
extensions                   1
foreign data wrappers        0
foreign servers              0
functions                    0
indexes                      2
operator classes             0
operator families            0
operators                    0
procedural languages         0
protocols                    0
resource groups              2
resource queues              1
roles                        3
rules                        0
schemas                      2
sequences                    2
tables                       6
tablespaces                  0
text search configurations   0
text search dictionaries     0
text search parsers          0
text search templates        0
triggers                     0
types                        0
user mappings                0
views                        0
```

JSON

```
$ adbm backup report --backup-name 20241206091238 --backup-format logical --output json
```

Result:

```
{
  "report" : "Greenplum Database Backup Report\n\ntimestamp key:         20241206091238\ngpdb version:          6.27.
1_arenadata59 build 3956+gitc4500d63cbe\ngpbackup version:      1.30.5\n\ndatabase name:         test\ncommand line:
gpbackup --dbname test --backup-dir /dasha --compression-level 2 --compression-type zstd --verbose\ncompression:
zstd\nplugin executable:     None\nbackup section:        All Sections\nobject filtering:      None\nincludes statistics:
No\ndata file format:      Multiple Data Files Per Segment\nincremental:           False\n\nstart time:            Fri Dec
06 2024 09:12:38\nend time:              Fri Dec 06 2024 09:12:40\nduration:              0:00:02\n\nbackup status:
Success\n\ndatabase size:         167 MB\nsegment count:         8\n\ncount of database objects in backup:\naggregates
0\ncasts                    0\ncollations               0\nconstraints              4\nconversions
0\ndefault privileges       0\ndatabase gucs            0\nevent triggers           0\nextensions
1\nforeign data wrappers    0\nforeign servers          0\nfunctions                0\nindexes
2\noperator classes         0\noperator families        0\noperators                0\nprocedural languages
0\nprotocols                0\nresource groups          2\nresource queues          1\nroles
3\nrules                    0\nschemas                  2\nsequences                2\ntables
6\ntablespaces              0\ntext search configurations 0\ntext search dictionaries  0\ntext search parsers
0\ntext search templates    0\ntriggers                 0\ntypes                    0\nuser mappings
0\nviews                    0\n"
}
```

backup delete

The `adbm backup delete` command allows you to delete a backup with the specified name (previously created via the backup create command).

Syntax:

```
$ adbm backup delete \
--backup-format | -f <backupFormat> \
--backup-name <backupName> \
[--before]
[-a]
[--log-level <logLevel>]
[--log-path <logPath>]
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: - `binary | b` — binary backups; - `logical | l` — logical backups.	Yes	—
--backup-name	- **For binary backups** — a name of the restore point. - **For logical backups** — a name of the logical backup in the `YYYYMMDDHHMMSS` format. It is returned by the backup create command and can be obtained with the backup ls command.	Yes	—
before	Indicates whether to delete all backups older than the given one	No	—
-a	Defines whether to launch the command automatically (without user confirmation)	No	false
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

In the command output, you can see an identifier of the ADBM action in the following format:

```
Action id = <actionId>
```

The successful result also contains a name of the removed backup:

```
Deletion of <backupName> completed
```

Example

```
$ adbm backup delete --backup-format logical --backup-name 20240715151947
```

Result:

```
Are you sure you want to delete logical backup? (y/n):y
Delete backup in progress. Action id = 682cc090-e64a-4332-b084-8b1728dcf71a. Performing "Start 'Delete logical backup' action".
done
Deletion of 20240715151947 completed
```

backup status

The `adbm backup status` command returns information about the specified action that was launched by one of the following commands:

- backup repo-config create
- backup repo-config delete
- backup config create
- backup config delete
- backup create
- backup restore
- backup delete

Syntax:

```
$ adbm backup status \
--backup-format | -f <backupFormat> \
--action-id <actionId> \
[--all]
[--log-level <logLevel>]
[--log-path <logPath>]
[--help | -h]
```

Command options

Option	Description	Mandatory	Default
--backup-format | -f	Determines whether the command operates with binary or logical backups. Possible values: ■ `binary | b` — binary backups; ■ `logical | l` — logical backups.	Yes	—

--action-id	A unique identifier of the action. It is returned by the following commands: - backup repo-config create - backup repo-config delete - backup config create - backup config delete - backup create - backup restore - backup delete	Yes	—
--all	Indicates whether to include information about child actions into the command output (in the `childActions` field)	No	—
--log-level	A logging level for the command	No	info
--log-path	A path where the command log files are stored	No	/home/gpadmin/gpAdminLogs
--help | -h	Displays a helper with a list of options supported by the command	No	—

Output parameters

The command returns a JSON file with information on the specified action.

Template:

```
{
  "clusterName" : "string",
  "startTime" : timestamp,
  "endTime" : timestamp,
  "result": string,
  "actionData" : {
    "id" : "string",
    "name" : "string",
    "status" : "string(enum)",
    "type" : "string(enum)",
    "username" : "string"
  },
  "childActions" : [
    {
      "clusterName" : "string",
      "startTime" : timestamp,
      "endTime" : timestamp,
      "result": string,
      "actionData" : {
        "id" : "string",
        "name" : "string",
        "status" : "string(enum)",
        "type" : "string(enum)",
        "username" : "string"
      },
      "childActions" : [],
      "level" : 1
    },
    ...
   ],
  "level" : 0
}
```

JSON fields

Field	Description

clusterName	A cluster name
startTime	An action start timestamp
endTime	An action end timestamp
result	An action result (if any)
actionData.id	An action identifier
actionData.name	An action name
actionData.status	An action status. Possible values: ▪ `Started` — the action is being started. ▪ `Stopped` — the action is stopped. ▪ `Done` — the action is completed successfully. ▪ `Failed` — the action is failed. ▪ `Terminating` — the action is being terminated. ▪ `Awaiting` — the action is waiting for something. ▪ `Unknown` — the action status is unknown. ▪ `Warning` — something went wrong during the action, but it is not critical for the parent action flow.
actionData.type	An action type. Possible values: ▪ `sql` — SQL request; ▪ `cmd` — shell command; ▪ `api` — API call.
actionData.username	The user who performed the selected action
childActions	An array of the child actions (if any). Filled in if the `--all` command option is set. Each child action has the same fields as the top-level action
level	An action level. `0` for the parent action, `1` — for its children, and so on.

Example

Without --all

```
$ adbm backup status --action-id 682cc090-e64a-4332-b084-8b1728dcf71a --backup-format logical
```

Result:

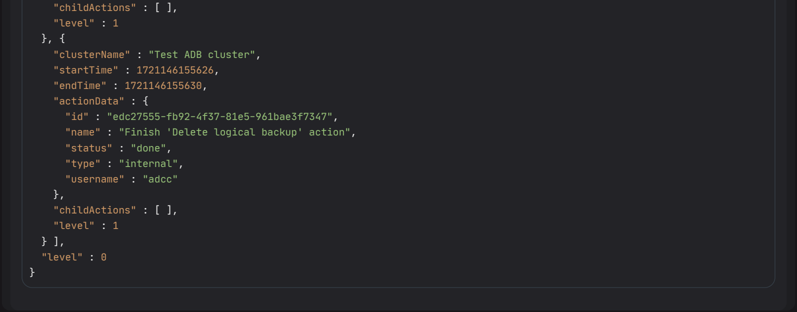
```
{
  "clusterName" : "Test ADB cluster",
  "startTime" : 1721146147611,
  "endTime" : 1721146155632,
  "actionData" : {
    "id" : "682cc090-e64a-4332-b084-8b1728dcf71a",
    "name" : "Start 'Delete logical backup' action",
    "status" : "done",
    "type" : "api",
    "username" : "adcc"
  },
  "childActions" : [ ],
  "level" : 0
}
```

With --all

```
$ adbm backup status --action-id 682cc090-e64a-4332-b084-8b1728dcf71a --backup-format logical --all
```

Result:

```
{
  "clusterName" : "Test ADB cluster",
  "startTime" : 1721146147611,
  "endTime" : 1721146155632,
  "actionData" : {
    "id" : "682cc090-e64a-4332-b084-8b1728dcf71a",
    "name" : "Start 'Delete logical backup' action",
    "status" : "done",
    "type" : "api",
    "username" : "adcc"
  },
  "childActions" : [ {
    "clusterName" : "Test ADB cluster",
    "startTime" : 1721146147709,
    "endTime" : 1721146147745,
    "actionData" : {
      "id" : "e59a4f9b-fbc2-45a2-ab5a-9014c3ee3dca",
      "name" : "Get cluster topology",
      "status" : "done",
      "type" : "internal",
      "username" : "adcc"
    },
    "childActions" : [ ],
    "level" : 1
  }, {
    "clusterName" : "Test ADB cluster",
    "startTime" : 1721146147751,
    "endTime" : 1721146147778,
    "actionData" : {
      "id" : "0c43d0e1-0122-49e1-8409-540608c9e571",
      "name" : "Save topology metadata",
      "status" : "done",
      "type" : "internal",
      "username" : "adcc"
    },
    "childActions" : [ ],
    "level" : 1
  }, {
    "clusterName" : "Test ADB cluster",
    "startTime" : 1721146147789,
    "endTime" : 1721146147808,
    "actionData" : {
      "id" : "6623b6f4-61df-4428-8443-0b0b86813d10",
      "name" : "Get logical backup for deletion",
      "status" : "done",
      "type" : "internal",
      "username" : "adcc"
    },
    "childActions" : [ ],
    "level" : 1
  }, {
    "clusterName" : "Test ADB cluster",
    "startTime" : 1721146147818,
    "endTime" : 1721146155591,
    "result" : "[{\"target\":\"10.92.43.169:6571\",\"requestId\":\"d7dfc140-efc6-4cff-a6de-
b7cc68802455\",\"actionId\":\"100f4faf-e130-4ce7-b6c2-
19b5ffe052b8\",\"clusterId\":1,\"success\":true,\"jsonBody\":null,\"errors\":[],\"segmentIds\":[-1],\"commands\":
[{\"command\":\"delete logical backups from gpbackup_history.db\",\"segmentId\":-1}]},{\"target\":\"10.92.40.
79:6571\",\"requestId\":\"38395ef8-0fea-4098-82f2-e78c74c1c959\",\"actionId\":\"100f4faf-e130-4ce7-b6c2-
19b5ffe052b8\",\"clusterId\":1,\"success\":true,\"jsonBody\":null,\"errors\":[],\"segmentIds\":[4,5,6,7],\"commands\":
[{\"command\":\"delete logical backups from gpbackup_history.db\",\"segmentId\":-1}]},{\"target\":\"10.92.40.
109:6571\",\"requestId\":\"93a8b36b-4e5f-4d98-9625-eb28868e2a2a\",\"actionId\":\"100f4faf-e130-4ce7-b6c2-
19b5ffe052b8\",\"clusterId\":1,\"success\":true,\"jsonBody\":null,\"errors\":[],\"segmentIds\":[0,1,2,3],\"commands\":
[{\"command\":\"delete logical backups from gpbackup_history.db\",\"segmentId\":-1}]}]",
    "actionData" : {
      "id" : "100f4faf-e130-4ce7-b6c2-19b5ffe052b8",
      "name" : "Physical removal of logical backups",
      "status" : "done",
      "type" : "internal",
      "username" : "adcc"
    },
```



```
    "childActions" : [ ],
    "level" : 1
  }, {
    "clusterName" : "Test ADB cluster",
    "startTime" : 1721146155626,
    "endTime" : 1721146155630,
    "actionData" : {
      "id" : "edc27555-fb92-4f37-81e5-961bae3f7347",
      "name" : "Finish 'Delete logical backup' action",
      "status" : "done",
      "type" : "internal",
      "username" : "adcc"
    },
    "childActions" : [ ],
    "level" : 1
  } ],
  "level" : 0
}
```

ADB ClickHouse Connector overview

Daria Barysheva

To Table of Contents

Unlike the previous version Tkhemali Connector 1.X based on the mechanism of external tables, ADB ClickHouse Connector is implemented on the basis of a foreign data wrapper and foreign tables. Consequently, the new version of the ADQM/ClickHouse connector has the following features:

- **More convenient and secure user management**. While in the previous connector version you had to write user credentials in the `LOCATION` option at the level of each external table or in a separate file explicitly, in ADB ClickHouse Connector the user login and password are stored in a separate ADB database object user mapping. It is enough to create a user mapping object once for one ClickHouse cluster with which you plan to connect. If necessary, the data can be modified later via the `ALTER USER MAPPING` command.

- **Multi-level connection settings**. General settings that will be the same for all or most ClickHouse target tables (load distribution strategy, need to use a staging layer, restrictions for batch data, and so on) can be specified at the server object level, and private settings (e.g. specific table name) — at the foreign table level. In the previous version of the connector, all settings had to be specified at the external table level.

- **New load balancing mechanism**. In the new connector version, the list of ClickHouse hosts (through which you plan to load data from ADB) is specified in the `hosts` option. The separate option `distribution_type` is used to declare the load distribution strategy. Two types of distribution are supported now: random and round-robin. The second method is preferable in terms of uniformity of load. At the same time, when writing to distributed tables, it is possible to specify the ClickHouse cluster name in the `cluster` option at the foreign table level. In this case, it is not necessary to define explicitly all hosts: one entry point is sufficient, and the full list of hosts will be automatically obtained from the cluster metadata. In the previous version of the connector, it was also possible to get a list of hosts for distributed tables, but the load on ClickHouse cluster hosts could be uneven.

- **Flexible management of custom settings**. Custom settings of connection and execution of queries to ClickHouse can now be defined in the `clickhouse_properties` option in the following format: `<custom_option>=<value>;[…]`. To change the settings, it is enough to execute the `ALTER SERVER` or the `ALTER FOREIGN TABLE` command. In the previous version of the connector, the custom settings for ClickHouse were stored in the XML file that needed to be copied to all ADB segment hosts.

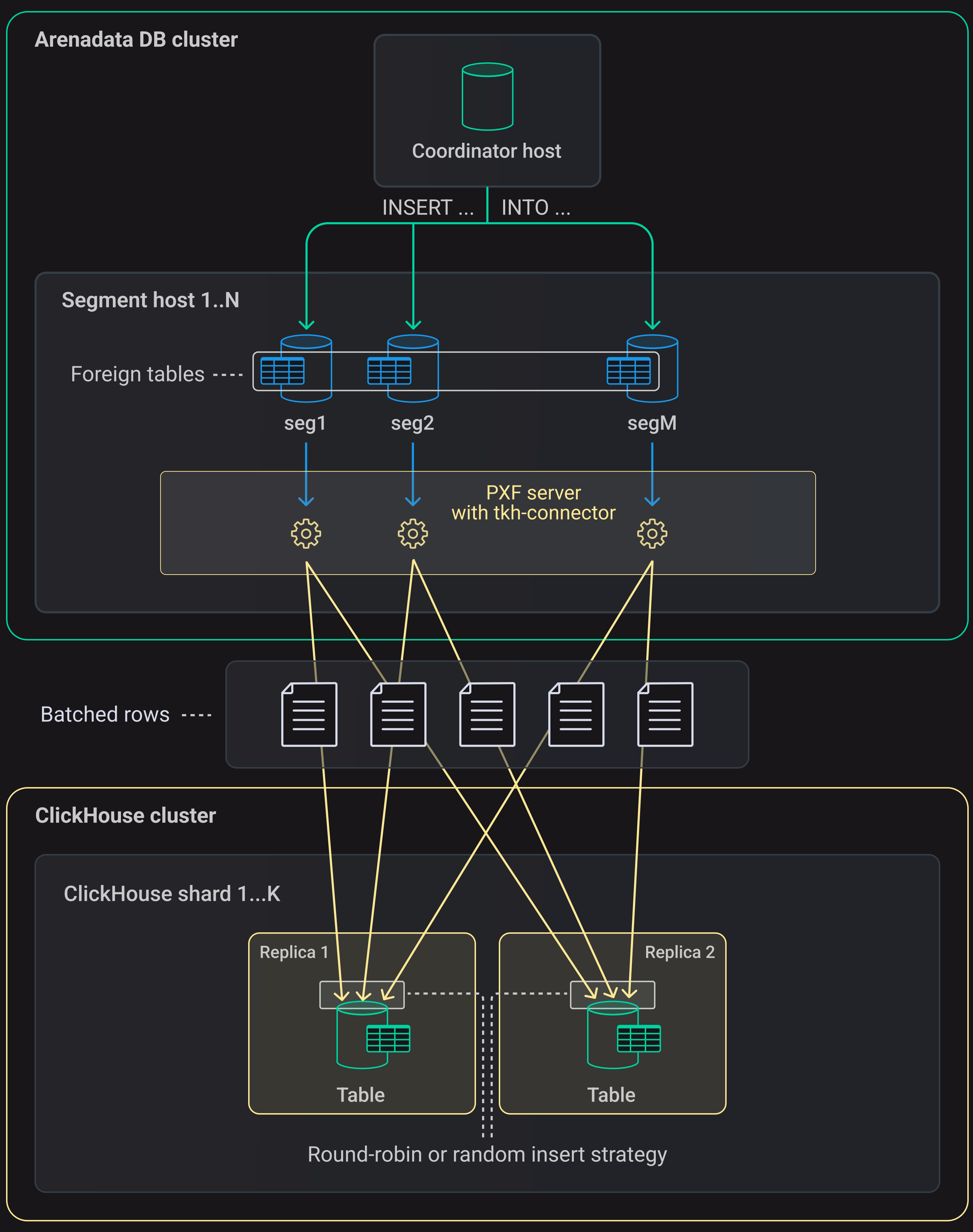
- **More control over the resource release**. A foreign data wrapper supports "out of box" transactions and allows subscribing to transaction control hooks. Due to this, the new connector version provides more control over the progress of requests and resource release, which is especially relevant for transactional data loading via staging tables. In the previous version of the connector, a staging layer might not have been removed under certain circumstances and it was necessary to remove it manually.

- **Simple query syntax**. There is no need to use the `txn` function to enable a transaction mode (staging layer usage) anymore. In the new connector version for data transmission from ADB to ClickHouse, it is enough to execute the ordinary `INSERT` query against the corresponding foreign table. Staging tables are now managed by a separate flag `use_staging`. In addition, in the previous connector version, it was necessary to specify full names of staging tables when creating external tables. In ADB ClickHouse Connector, only the target table name in ClickHouse is required to be filled in the `resource` option at the foreign table level. Names for staging tables are automatically generated based on the template that is specified in the `staging_table_name_format` option.

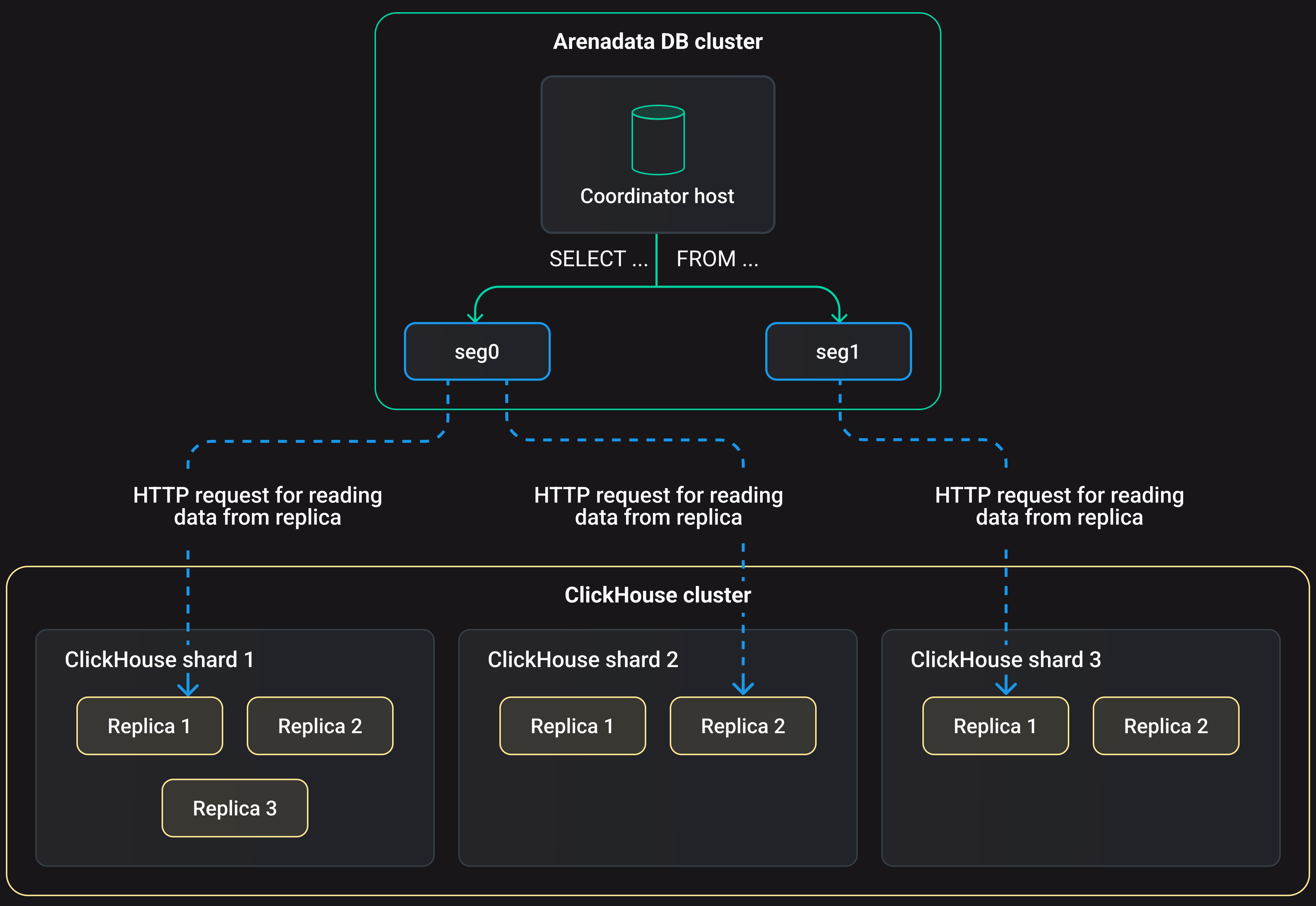
IMPORTANT

- ADB ClickHouse Connector is available in the Enterprise version of ADB:
  - In ADB 7 — starting with the 7.4.0 version.
  - In ADB 6 — starting with the 6.23.3.44 version.
- Starting with 6.24.3.47 in ADB 6 and 7.4.0 in ADB 7, ADB ClickHouse Connector supports both writing and reading data from ADQM/ClickHouse. However, to use a foreign table in ADB for reading, you should specify column names that match exactly the corresponding names in a source table in ADQM/ClickHouse.

The high-level architecture view of the new connector version is shown below.


Arenadata DB cluster
Coordinator host
INSERT ...
INTO ...
Segment host 1..N
Foreign tables
seg1
seg2
segM
PXF server
with tkh-connector
Batched rows
ClickHouse cluster
ClickHouse shard 1...K
Replica 1
Replica 2
Table
Table
Round-robin or random insert strategy

INSERT execution schema


Arenadata DB cluster
Coordinator host
SELECT ...
FROM ...
seg0
seg1
HTTP request for reading
data from replica
HTTP request for reading
data from replica
HTTP request for reading
data from replica
ClickHouse cluster
ClickHouse shard 1
Replica 1
Replica 2
Replica 3
ClickHouse shard 2
Replica 1
Replica 2
ClickHouse shard 3
Replica 1
Replica 2

SELECT execution schema

ADB ClickHouse Connector installation

Daria Barysheva

To Table of Contents

To install ADB ClickHouse Connector, follow the steps:

1. Make sure that PXF and ADB ClickHouse Connector services are installed. For information on how to add services to ADB clusters, see Add services.

2. Add the new custom profile `Tkh_clickhouse:text` to the PXF configuration file */var/lib/pxf/conf/pxf-profiles.xml* on **coordinator**. To edit the file, you can use the `vi` or `vim` command:

```
$ sudo vi /var/lib/pxf/conf/pxf-profiles.xml
```

The content of the changed file can look as follows. In the given example, the file does not contain other profiles:

```
<profiles>
    <profile>
        <name>Tkh_clickhouse:text</name>
        <description>Clickhouse</description>
        <plugins>
            <accessor>io.arenadata.tkh.pxf.TkhFdwAccessor</accessor>
            <resolver>io.arenadata.tkh.pxf.TkhResolver</resolver>
        </plugins>
    </profile>
</profiles>
```

NOTE

The directory, where PXF configuration files are located, is defined by the `PXF_BASE` environment variable. The default value is */var/lib/pxf*. The path may differ in your environment.

3. Log in under the default user `gpadmin`. All the commands listed in the subsequent steps should be performed on **coordinator**:

```
$ sudo su - gpadmin
```

4. Synchronize the PXF configuration on all hosts in the ADB cluster:

```
$ pxf cluster sync
```

Result:

```
Syncing PXF configuration files from coordinator host to standby coordinator host and 2 segment hosts...
PXF configs synced successfully on 3 out of 3 hosts
```

5. Restart PXF:

```
$ pxf cluster restart
```

Result:

```
Restarting PXF on coordinator host, standby coordinator host, and 2 segment hosts...
PXF restarted successfully on 4 out of 4 hosts
```

6. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

7. To check that the new extensions `pxf_fdw` and `tkh_fdw` are available, use the `psql` meta-command `\dx`:

```
\dx
```

The output contains the `pxf_fdw` and `tkh_fdw` extensions:

```
                                      List of installed extensions
       Name        | Version |   Schema   |                              Description
-------------------+---------+------------+-----------------------------------------------------------------------
 adb_fdw           | 1.0     | public     | foreign-data wrapper for remote ADB servers
 arenadata_toolkit | 1.0     | public     | extension is used for manipulation of objects created by adb-bundle
 dblink            | 1.2     | public     | connect to other PostgreSQL databases from within a database
 gp_exttable_fdw   | 1.0     | pg_catalog | External Table Foreign Data Wrapper for Greengage
 gp_toolkit        | 1.8     | gp_toolkit | various GPDB administrative views/functions
 kadb_fdw          | 0.16    | kadb       | Kafka-ADB foreign data wrapper
 plpgsql           | 1.0     | pg_catalog | PL/pgSQL procedural language
 pxf               | 2.1     | public     | Extension which allows to access unmanaged data
 pxf_fdw           | 1.0     | public     | PXF Foreign Data Wrapper for Greengage
 tkh_fdw           | 1.0     | public     | ClickHouse Foreign Data Wrapper for Greenplum
(10 rows)
```

8. You can also check that the new foreign data wrapper `tkh_fdw` is available using the following SQL query against the `pg_catalog.pg_foreign_data_wrapper` system table:

```
SELECT * FROM pg_catalog.pg_foreign_data_wrapper;
```

Result:

```
  oid  |     fdwname     | fdwowner | fdwhandler | fdwvalidator | fdwacl |                                        fdwoptions
-------+-----------------+----------+------------+--------------+--------+----------------------------------------------------------------
-------------------------------------
 13702 | gp_exttable_fdw |       10 |      13700 |        13701 |        |
 17124 | dblink_fdw      |       10 |          0 |        17123 |        |
 17175 | adb_fdw         |       10 |      17173 |        17174 |        |
 17187 | jdbc_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=jdbc,"mpp_execute=all segments"}
 17188 | hdfs_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=hdfs,"mpp_execute=all segments"}
 17189 | hive_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=hive,"mpp_execute=all segments"}
 17190 | hbase_pxf_fdw   |       10 |      17185 |        17186 |        | {protocol=hbase,"mpp_execute=all segments"}
 17191 | s3_pxf_fdw      |       10 |      17185 |        17186 |        | {protocol=s3,"mpp_execute=all segments"}
 17192 | gs_pxf_fdw      |       10 |      17185 |        17186 |        | {protocol=gs,"mpp_execute=all segments"}
 17193 | adl_pxf_fdw     |       10 |      17185 |        17186 |        | {protocol=adl,"mpp_execute=all segments"}
 17194 | wasbs_pxf_fdw   |       10 |      17185 |        17186 |        | {protocol=wasbs,"mpp_execute=all segments"}
 17195 | file_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=file,"mpp_execute=all segments"}
 17201 | tkh_fdw         |       10 |      17200 |        17199 |        | {protocol=tkh_clickhouse,"mpp_execute=all
segments",format=text}
 17210 | kadb_fdw        |       10 |      17209 |        17211 |        | {"mpp_execute=all
segments",k_initial_offset=0,k_automatic_offsets=true,k_latency_ms=2000}
(14 rows)
```

ADB ClickHouse Connector configuration

Daria Barysheva

Contents

To Table of Contents

Server

User mapping

Foreign table

To send data from ADB to ClickHouse via ADB ClickHouse Connector, you should first create the following objects on the ADB cluster side:

1. Server — encapsulates connection information that a foreign data wrapper uses to access an external data source.
2. User mapping — provides user authentication to an external data source.
3. Foreign table — a table in ADB that defines the remote data structure. A foreign table has no storage in ADB, but can be used in queries just like a normal table.

To see how to use the listed objects to send data from ADB to ADQM, refer to ADB ClickHouse Connector usage examples.

IMPORTANT

- For each ClickHouse cluster to which you plan to send queries from ADB, it is sufficient to create one server and one user mapping. However, there may be several foreign tables if you need to send queries against various ClickHouse tables.
- All options that are listed in the tables below (Server options, User mapping options, Foreign table options) should be defined in the text format (i.e. in single quotes `'`). The ***Type*** column in the tables shows what data type will be used when validating input values.

Server

To create a server, use the `CREATE SERVER` command. The basic command syntax is listed below:

```
CREATE SERVER <server_name> [ TYPE '<server_type>' ] [ VERSION '<server_version>' ]
    FOREIGN DATA WRAPPER <fdw_name>
    [ OPTIONS ( [ <option> '<value>' [, ... ]] ) ]
```

where:

- `<server_name>` — a server name in ADB. Should be unique within the current ADB database.
- `<server_type>` — an optional server type.
- `<server_version>` — an optional server version.
- `<fdw_name>` — a foreign data wrapper name. You should use the `tkh_fdw` foreign data wrapper, which is automatically created after the connector installation (see step 8 in ADB ClickHouse Connector installation).
- `<option>` — server options that define the connection details. The options that are available for ADB ClickHouse Connector are listed in the Server options table. Note that these options can be defined both at the server level and at the foreign table level. Options marked as required should be specified at least in one definition (server or table). Table-level options take precedence.
- `<value>` — option values.

NOTE

- To create a server, you need the `USAGE` privilege on the `tkh_fdw` foreign data wrapper. The user who creates the server becomes its owner.
- The full description of the `CREATE SERVER` command syntax is available in the Greenplum documentation.
- To edit a server definition, use the ALTER SERVER command. To delete a server, use DROP SERVER.

Server options

Name	Type	Description	Default	Required
database	TEXT	A name of the ClickHouse database where the target table is located. Should match the following regular expression: `^[a-zA-Z_][0-9a-zA-Z_]*$`	—	Yes
hosts	TEXT	A list of ClickHouse hosts where the target table is located. A ClickHouse cluster does not have a single entry point (coordinator node), so the requests can be sent to any host in the cluster. To reduce the load, data batches are sent to hosts from the connector according to the `random` or the `round-robin` distribution strategy (see `distribution_type` below). Validation rules for the `hosts` option are listed below	—	Yes
lines_batch_size	INT	The maximum number of rows in the data batch sent to ClickHouse by the connector. Positive integers are allowed	100000	No
bytes_batch_size_mb	INT	The maximum batch size of data sent to ClickHouse by the connector (in MB). If the `lines_batch_size` option is also defined, the `bytes_batch_size_mb` value will be used. Positive integers are allowed	—	No
send_compressed	BOOL	The flag that indicates whether to use compression on the connector side before sending data to ClickHouse. Possible values: - `true` — use data compression. - `false` — do not use data compression.	false	No
send_delay	INT	A delay between connector requests to ClickHouse (in milliseconds). Positive integers are allowed	300	No
insert_distributed_sync	BOOL	Enables or disables synchronous data load to distributed tables. Possible values: - `true` — data is inserted synchronously. The `INSERT` query is considered successful after all data is added to all shards (at least to one replica per chard if `internal_replication` = `true` in ClickHouse). - `false` — data is inserted asynchronously.	true	No
distribution_type	TEXT	Defines the strategy of load distribution between ClickHouse hosts (see `hosts` above) when performing data load operations from the connector side. Possible values: - `random` — hosts are selected randomly. - `round-robin` — hosts are selected based on the round-robin algorithm. This option value is preferable in terms of the uniform load distribution.	random	No
use_staging	BOOL	The flag that indicates whether to create staging tables in ClickHouse before inserting data into the target table. Used to emulate transactions that are not present in ClickHouse. Staging tables are supported only for the following ClickHouse engine families: MergeTree (including Replicated* engines) and Distributed. For other engines, an error will be returned, since not all engine types support partitioning in ClickHouse. Possible values: - `true` — use staging tables. - `false` — do not use staging tables. In this mode, there are certain risks, as the consistency of data insertions can be disturbed. However, the mode is suitable for single insertions into empty target tables.	—	Yes
staging_table_name_format	TEXT	Defines the staging table name format (see `use_staging` above). Example: `'$_tmp_$'`. The first placeholder (`$`) is replaced with the name of the target table in ClickHouse (see `resource` in Foreign table options). The second placeholder is replaced with the integer identifier that is automatically generated on the coordinator node of ADB	—	Yes if `use_staging` = `true`
clickhouse_properties	TEXT	A list of ClickHouse custom options in the `<custom_option>=<value>;[…]` format, where `<custom_option>` is an option name and `<value>` is an option value	—	No
fdw_startup_cost	DOUBLE	The cost estimate for obtaining the first tuple from the data source. Positive double values are allowed. Can be used only in `SELECT` queries to ADQM/ClickHouse	1000.0	No
fdw_tuple_cost	DOUBLE	The cost estimate for obtaining one tuple from the data source. Positive double values are allowed. Can be used only in `SELECT` queries to ADQM/ClickHouse	0.01	No
fdw_tuples_count	DOUBLE	The estimate for the number of all rows in the data source. Does not take into account the query selectivity, `LIMIT` clauses, etc. Positive double values >= `1` are allowed. Can be used only in `SELECT` queries to ADQM/ClickHouse	100000.0	No

Validation rules for the hosts option

- The list of ClickHouse hosts should be specified in the following format: `<hostname>:<port> [, …]`, where `<hostname>` is a name of each host and `<port>` is an HTTP port number to connect to the corresponding host.
- The maximum length of the `<hostname>` value is 255 characters.
- You can use in the `<hostname>` value digits `0-9`, ASCII characters `a-zA-Z`, hyphens `-`, underscores `_`, and dots `.`.
- The presence of the `<port>` substring for each `<hostname>` is required. Only digits `0-9` are allowed in the `<port>` value. The maximum length of the `<port>` value is 5 characters.
- The maximum length of the `hosts` value: (255 (the `<hostname>` limit) + 1 (`:`) + 5 (the `<port>` limit) + 1 (`,`)) x 127 (number of hosts) = 33274 characters.

User mapping

To create a user mapping, use the `CREATE USER MAPPING` command. The basic command syntax is listed below:

```
CREATE USER MAPPING FOR { <username> | USER | CURRENT_USER | PUBLIC }
    SERVER <server_name>
    [ OPTIONS ( <option> '<value>' [, ... ] ) ]
```

where:

- `<username>` — a user name in ADB. The following constants are allowed:
  - `USER` or `CURRENT_USER` — the current user who established the ADB connection.
  - `PUBLIC` — all present and future ADB users.
- `<server_name>` — a server name.
- `<option>` — user mapping options, which typically include a user name and a password to connect to an external data source. The options that are available for ADB ClickHouse Connector are listed in the User mapping options table.
- `<value>` — option values.

NOTE

- To create a user mapping, you should be the owner of the corresponding server.
- The full description of the `CREATE USER MAPPING` command syntax is available in the Greenplum documentation.
- To edit a user mapping definition, use the ALTER USER MAPPING command. To delete a user mapping, use DROP USER MAPPING.

User mapping options

Name	Type	Description	Default	Required
clickhouse_username	TEXT	A user name in ClickHouse	—	Yes
clickhouse_password	TEXT	A user password in ClickHouse. If there is no password (for example, in the test environment), an empty string is specified	—	Yes

Foreign table

To create a foreign table, use the `CREATE FOREIGN TABLE` command. The basic command syntax is listed below:

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] <table_name> ( [
    <column_name> <data_type> [ COLLATE <collation> ] [ <column_constraint> [ ... ] ]
      [, ... ]
] )
    SERVER <server_name>
  [ OPTIONS ( <option> '<value>' [, ... ] ) ]
```

where:

- `<table_name>` — a foreign table name in ADB.
- `<column_name>` — a column name.
- `<data_type>` — a column data type.
- `<collation>` — a column collation.
- `<column_constraint>` — a constraint defined at the column level. The name `<constraint_name>` is given optionally. The syntax:

```
[ CONSTRAINT <constraint_name> ]
{ NOT NULL |
  NULL |
  DEFAULT <default_expr> }
```

  Possible constraints:

  - `NOT NULL` — specifies that the column is not allowed to contain null values.
  - `NULL` — specifies that the column is allowed to contain null values. It is the default behavior (unless `NOT NULL` is specified).
  - `DEFAULT` — sets the default column value equal to `<default_expr>`.
- `<server_name>` — a server name.
- `<option>` — foreign table options. For ADB ClickHouse Connector, all options defined at the server level can be overwritten at the foreign table level (partially or fully). When you specify an option for a server and a foreign table at the same time, the table level takes precedence. Additionally, at the foreign table level, two more options are available (see Foreign table options below).
- `<value>` — option values.

NOTE

- To create a foreign table, you need the `USAGE` privilege on the corresponding server and on all column data types that are used in the table definition. The user who creates the foreign table becomes its owner.
- The full description of the `CREATE FOREIGN TABLE` command syntax is available in the Greenplum documentation.
- To edit a foreign table definition, use the ALTER FOREIGN TABLE command. To drop a foreign table, use DROP FOREIGN TABLE.

Foreign table options

Name	Type	Description	Default	Required
resource	TEXT	A table name in ClickHouse. If you need your own strategy for distribution data across hosts (sharding), you should insert data into the distributed ClickHouse table with the sharding key configured. To do this, specify the distributed table name in the `resource` option. When writing data to a distributed table, remember that physical (local) tables should be present on all hosts in the ClickHouse cluster, otherwise the connector will return an error. The option value should match the following regular expression: `^[a-zA-Z_][0-9a-zA-Z_]*$`	—	Yes
cluster	TEXT	A cluster name in ClickHouse. For more information, see Typical cluster in the ADQM documentation. If you specify the `cluster` option, the list of ClickHouse hosts should be defined in the `hosts` option at the foreign table level and consist of one element `<hostname>:<port>`, even if it has been already declared at the server level. Otherwise, there may be an ambiguous situation when there is a list of hosts at the server level, but the work is actually done with hosts received from the ClickHouse cluster metadata. The `cluster` option provides the convenience of users. With that option, it is enough to specify a cluster name and one entry point, the full list of hosts will be received from ClickHouse. The load between hosts will be still distributed according to the selected `distribution_type` value (see Server options). The maximum `cluster` length is 128 characters	—	No

ADB ClickHouse Connector usage examples

Daria Barysheva

Contents

To Table of Contents

Insert data
Select data
- Correct example
- Error example

This article describes how to transfer data between ADB and a distributed table in ADQM 25.3.2.39 via ADB ClickHouse Connector. The following prerequisites are met:

- The ADQM cluster is installed according to the Online installation guide on the following hosts: `dev-adqm-01.ru-central1.internal`, `dev-adqm-02.ru-central1.internal`, `dev-adqm-03.ru-central1.internal`, `dev-adqm-04.ru-central1.internal`.
- The `default` database exists in ADQM.
- The `default` user exists in ADQM.
- The `default_cluster` cluster is defined in the `remote_servers` section of the *config.xml* file in ADQM. For more information, see Typical cluster in the ADQM documentation.
- All ADB nodes have access to the `8123` HTTP port on all ADQM nodes through which you plan to load data.

Insert data

To check that you can add data to ADQM from ADB, follow the steps:

1. Connect to the `default` database of the ADQM cluster under the `default` user (for example, via clickhouse-client). Create the following tables in the given order.

   - The `test_local` local table:

```
CREATE TABLE test_local ON CLUSTER default_cluster
(
    id Int32,
    value_string String
)
ENGINE = ReplicatedMergeTree('/clickhouse/tables/{shard}/test_local', '{replica}')
ORDER BY id;
```

   - The `test_distr` distributed table:

```
CREATE TABLE test_distr ON CLUSTER default_cluster AS default.test_local
ENGINE = Distributed(default_cluster, default, test_local, rand());
```

2. Connect to the `adb` database of the ADB cluster under the `gpadmin` user (for example, via `psql`). Create the following objects in the given order.

   - The `clickhouse_server` server:

```
CREATE SERVER clickhouse_server
FOREIGN DATA WRAPPER tkh_fdw
OPTIONS (
        database 'default',
        hosts 'dev-adqm-01.ru-central1.internal:8123, dev-adqm-02.ru-central1.internal:8123, dev-adqm-03.ru-central1.
internal:8123, dev-adqm-04.ru-central1.internal:8123',
        distribution_type 'random',
        lines_batch_size '100000',
        send_compressed 'true',
        send_delay '300',
        insert_distributed_sync 'true',
        use_staging 'true',
        staging_table_name_format '$_tmp_$');
```

   - The user mapping for the `gpadmin` user:

```
CREATE USER MAPPING FOR gpadmin
SERVER clickhouse_server
OPTIONS (clickhouse_username 'default', clickhouse_password '');
```

   - The `tkhemali_test` foreign table:

```
CREATE FOREIGN TABLE tkhemali_test (id INT, value_string TEXT)
SERVER clickhouse_server
OPTIONS (resource 'test_distr',
        hosts 'dev-adqm-01.ru-central1.internal:8123',
        cluster 'default_cluster');
```

> **IMPORTANT**
>
> Since the `cluster` option is defined, the work will be essentially produced with the hosts obtained from the ADQM cluster metadata. Due to this, remember to overwrite the `hosts` option value defined for `clickhouse_server` at the server level and leave only one host in the option value at the table level.

3. In ADB, insert some data into the `tkhemali_test` foreign table:

```
INSERT INTO tkhemali_test VALUES
(1, 'test1'),
(2, 'test2'),
(3, 'test3'),
(4, 'test4'),
(5, 'test5'),
(6, 'test6'),
(7, 'test7'),
(8,'test8'),
(9,'test9'),
(10,'test10');
```

   Result:

```
INSERT 0 10
```

4. In ADQM, check that data from ADB has been successfully transferred:

```
SELECT * FROM test_distr;
```

   Result:

```
    ┌─id─┬─value_string─┐
 1. │  5 │ test5        │
 2. │  7 │ test7        │
 3. │  9 │ test9        │
 4. │ 10 │ test10       │
 5. │  3 │ test3        │
 6. │  6 │ test6        │
 7. │  8 │ test8        │
 8. │  1 │ test1        │
 9. │  4 │ test4        │
10. │  2 │ test2        │
    └────┴──────────────┘

10 rows in set. Elapsed: 0.008 sec.
```

Select data

> **NOTE**
>
> ADB ClickHouse Connector supports the ability to select data from ADQM/ClickHouse to ADB starting with the 6.24.3.47 version in ADB 6 and the 7.4.0 version in ADB 7. However, to use a foreign table in ADB for reading, you should specify column names that match exactly the corresponding names in a source table in ADQM/ClickHouse.

Correct example

To check that ADQM data can be read from ADB, do the following:

1. Perform steps 1—2 that are listed above (if you have not passed them yet).
2. Delete data from the `test_local` table in ADQM:

```
ALTER TABLE default.test_local ON CLUSTER default_cluster DELETE WHERE true;
```

3. In ADQM, add some data to the `test_distr` table:

```
INSERT INTO test_distr VALUES
(1, 'test1'),
(2, 'test2'),
(3, 'test3'),
(4, 'test4'),
(5, 'test5'),
(6, 'test6'),
(7, 'test7'),
(8,'test8'),
(9,'test9'),
(10,'test10');
```

   Result:

```
Query id: 9eec272c-e292-46c5-be17-33bee1b0188e

Ok.

10 rows in set. Elapsed: 0.010 sec.
```

4. Check that data added to ADQM can be successfully selected in ADB:

```
SELECT * FROM tkhemali_test ORDER BY id;
```

   Result:

```
 id | value_string
----+--------------
  1 | test1
  2 | test2
  3 | test3
  4 | test4
  5 | test5
  6 | test6
  7 | test7
  8 | test8
  9 | test9
 10 | test10
(10 rows)
```

Error example

The following example shows that reading data from ADQM/ClickHouse to ADB is impossible if column names in the foreign ADB table and the source ADQM/ClickHouse table do not match:

1. Perform all steps from the previous example with correct data reading (if you have not passed them yet).
2. Ensure that reading data based on the current foreign table structure is successful in ADB:

```
SELECT * FROM tkhemali_test ORDER BY id;
```

   Result:

```
 id | value_string
----+--------------
  1 | test1
  2 | test2
  3 | test3
  4 | test4
  5 | test5
  6 | test6
  7 | test7
  8 | test8
  9 | test9
 10 | test10
(10 rows)
```

3. Rename some columns of the `tkhemali_test` table in ADB:

```
ALTER FOREIGN TABLE tkhemali_test RENAME value_string TO wrong_string;
```

4. Check again the ability to select data from `tkhemali_test` in ADB:

```
SELECT * FROM tkhemali_test ORDER BY id;
```

   The following error is displayed:

```
ERROR:  CH easy datum. HTTP error: 404; CH message: Code: 47. DB::Exception: Unknown expression identifier `wrong_string` in
scope SELECT id, wrong_string FROM default.test_local. (UNKNOWN_IDENTIFIER) (version 25.3.2.39) (ch_easy_datum.c:141)  (seg1
slice1 10.92.38.12:10000 pid=1648668) (ch_easy_datum.c:141)
```

> **NOTE**
>
> If names are the same only for part of columns in the foreign table in ADB and the source table in ADQM/ClickHouse — the SQL query succeeds while the resulting dataset uses only the specified columns.

Integration between ADB and Kafka

Daria Barysheva

To Table of Contents

To integrate ADB and Kafka, you can use one of the following connectors:

- **ADB to Kafka Connector** — based on PXF and external tables. Implements data load from ADB to Kafka. For more information, see:
  - ADB to Kafka Connector installation
  - ADB to Kafka Connector configuration
  - ADB to Kafka Connector usage examples
- **Kafka to ADB Connector** — based on a foreign data wrapper mechanism. Uses librdkafka for communication with Kafka and implements transactional data load from Kafka to ADB. For more information, see:
  - Kafka to ADB Connector installation
  - Kafka to ADB Connector configuration
  - Kafka to ADB Connector usage examples

ADB to Kafka Connector installation

Daria Barysheva

To Table of Contents

To install ADB to Kafka Connector, follow the steps:

1. Make sure that PXF and ADB to Kafka services are installed. For information on how to add services to ADB clusters, see Add services.

2. Add the new custom profile `kafka` to the PXF configuration file */var/lib/pxf/conf/pxf-profiles.xml* on **coordinator**. To edit the file, you can use the `vi` or `vim` command:

```
$ sudo vi /var/lib/pxf/conf/pxf-profiles.xml
```

The content of the changed file can look as follows. In the given example, the file does not contain other profiles:

```
<profiles>
    <profile>
        <name>kafka</name>
        <description>A profile for export data into Apache Kafka</description>
        <plugins>
            <accessor>org.greenplum.pxf.plugins.kafka.KafkaAccessor</accessor>
            <resolver>org.greenplum.pxf.plugins.kafka.KafkaResolver</resolver>
        </plugins>
        <optionMappings>
            <!-- Kafka properties -->
            <mapping option="BOOTSTRAP_SERVERS" property="kafka.bootstrap.servers"/>
            <mapping option="SECURITY_PROTOCOL" property="kafka.property.security.protocol" />
            <mapping option="SASL_MECHANISM" property="kafka.property.sasl.mechanism" />
            <mapping option="TRUSTSTORE_LOCATION" property="kafka.property.ssl.truststore.location" />
            <mapping option="TRUSTSTORE_PASSWORD" property="kafka.property.ssl.truststore.password" />
            <mapping option="KEYSTORE_LOCATION" property="kafka.property.ssl.keystore.location" />
            <mapping option="KEYSTORE_PASSWORD" property="kafka.property.ssl.keystore.password" />
            <mapping option="KERBEROS_SERVICE_NAME" property="kafka.property.sasl.kerberos.service.name" />

            <!-- Kafka JAAS properties -->
            <mapping option="KERBEROS_PRINCIPAL" property="kafka.jaas.property.sasl.kerberos.principal" />
            <mapping option="KERBEROS_KEYTAB" property="kafka.jaas.property.sasl.kerberos.keytab" />
            <mapping option="KERBEROS_USE_KEYTAB" property="kafka.jaas.property.sasl.kerberos.useKeyTab" />
            <mapping option="KERBEROS_STORE_KEY" property="kafka.jaas.property.sasl.kerberos.storeKey" />
            <mapping option="KERBEROS_USE_TICKET_CACHE" property="kafka.jaas.property.sasl.kerberos.useTicketCache" />
            <mapping option="KERBEROS_DEBUG" property="kafka.jaas.property.sasl.kerberos.debug" />
            <mapping option="SASL_USER" property="kafka.jaas.property.sasl.user" />
            <mapping option="SASL_USER_PASSWORD" property="kafka.jaas.property.sasl.user.password" />
            <mapping option="JAAS_CONFIG_FILE" property="kafka.jaas.property.config.file" />

            <!-- General properties -->
            <mapping option="BATCH_SIZE" property="kafka.batch.size"/>
            <mapping option="KAFKA_ADMIN_CONNECTION_TIMEOUT" property="kafka.admin.close.connection.timeout"/>
            <mapping option="TOPIC_AUTO_CREATE_FLAG" property="kafka.topic.auto.create"/>
            <mapping option="AVRO_DEFAULT_DECIMAL_PRECISION" property="avro.decimal.default.precision" />
            <mapping option="AVRO_DEFAULT_DECIMAL_SCALE" property="avro.decimal.default.scale" />
        </optionMappings>
    </profile>
</profiles>
```

NOTE

The directory, where PXF configuration files are located, is defined by the `PXF_BASE` environment variable. The default value is */var/lib/pxf*. The path may differ in your environment.

3. Log in under the default user `gpadmin`. All the commands listed in the subsequent steps should be performed on **coordinator**:

```
$ sudo su - gpadmin
```

4. Synchronize the PXF configuration on all hosts in the ADB cluster:

```
$ pxf cluster sync
```

Result:

```
Syncing PXF configuration files from coordinator host to standby coordinator host and 2 segment hosts...
PXF configs synced successfully on 3 out of 3 hosts
```

5. Restart PXF:

```
$ pxf cluster restart
```

Result:

```
Restarting PXF on coordinator host, standby coordinator host, and 2 segment hosts...
PXF restarted successfully on 4 out of 4 hosts
```

ADB to Kafka Connector configuration

Daria Barysheva

Contents

To Table of Contents

Create a writable external table
Use the SERVER option
Use custom options
Custom configuration of the SASL authentication mechanism
Greenplum and AVRO data types

To send data from ADB to Kafka via ADB to Kafka Connector, you should first create a writable external table on the ADB cluster side. In the `LOCATION` clause, set the PXF protocol with the `kafka` profile and connection options.

To see how to send data from ADB to ADS using the settings listed below, refer to ADB to Kafka Connector usage examples.

Create a writable external table

To create a writable external table, use the `CREATE WRITABLE EXTERNAL TABLE` command. The basic command syntax is listed below:

```
CREATE WRITABLE EXTERNAL TABLE <table_name> (
    { <column_name> <data_type> [, ...] | LIKE <other_table> }
)
LOCATION (
    'pxf://<kafka_topic>?PROFILE=kafka[&SERVER=<server_name>][&<option>=<value>[...]]'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export')
[DISTRIBUTED BY (<column_name> [, ... ] ) | DISTRIBUTED RANDOMLY];
```

where:

- `<table_name>` — an external table name in ADB.
- `<column_name>` — a column name.
- `<data_type>` — a column data type.
- `<other_table>` — a source table from which column names, column data types, and a data distribution policy will be copied to the new external table. Note that the column constraints and default values that are specified in the source table are not copied because they are not supported in external tables.
- `<kafka_topic>` — a topic name in Kafka.
- `<server_name>` — a server name in the *$PXF_BASE/servers/* directory (by default, */var/lib/pxf/servers/*). For more information, see Use the SERVER option.
- `<option>` — options that define the connection details. The options that are available for ADB to Kafka Connector are listed in the following tables: General options, Kafka options, JAAS options for the SASL GSSAPI mechanism (Kerberos), JAAS options for the SASL PLAIN mechanism.
- `<value>` — option values.

NOTE

- For the optimal performance, the data distribution policy in the external table should be similar to the policy in the source table from which data will be selected before sending it to Kafka. This will allow you to send data directly from ADB segments rather than redistribute it before sending. For this purpose, it is recommended to use the `LIKE` clause when creating an external table, or explicitly specify a similar distribution key in the `DISTRIBUTED BY` clause.
- The full description of the `CREATE EXTERNAL TABLE` command syntax is available in the Greenplum documentation.
- To alter an external table definition, use the ALTER EXTERNAL TABLE command. To delete an external table, use DROP EXTERNAL TABLE.

General options

Name	Description	Default	Required
KAFKA_ADMIN_CONNECTION_TIMEOUT	A grace period during which current operations are allowed to complete after closing Kafka connection (in milliseconds). Once that time period is over, all operations that have not yet been completed are aborted with the following exception: `org.apache.kafka.common.errors.TimeoutException`	30000	No
BATCH_SIZE	A number of rows that producers should batch in one Avro message	1	No
TOPIC_AUTO_CREATE_FLAG	A flag that indicates whether a Kafka topic should be automatically created when producers try to write data into it. The topic will be created with one partition and replication factor = `1`	true	No
AVRO_DEFAULT_DECIMAL_PRECISION	A maximum number of digits in a number (except a decimal point). Positive integers are allowed	38	No
AVRO_DEFAULT_DECIMAL_SCALE	A maximum number of digits to the right of the decimal point in a number. Must be zero or a positive integer less than or equal to the `AVRO_DEFAULT_DECIMAL_PRECISION` value	18	No

Kafka options

Name	Description	Default	Required
BOOTSTRAP_SERVERS	A comma-separated list of Kafka brokers, each of which is specified in the following format: `<hostname>:<port>` or `<IP>:<port>`	—	Yes
SECURITY_PROTOCOL	A security protocol that is used to communicate with Kafka brokers. Possible values: - `PLAINTEXT` - `SASL_PLAINTEXT` - `SASL_SSL` - `SSL` (one-way and two-way authentication)	PLAINTEXT	No
SASL_MECHANISM	A SASL mechanism that is used for client connections. Possible values: - `PLAIN` - `GSSAPI` - `SCRAM-SHA-256` - `SCRAM-SHA-512` If you need another authentication mechanism, you can use custom settings to set up required Kafka producer parameters and JAAS configuration file	—	Yes if `SECURITY_PROTOCOL=SASL_PLAINTEXT|SASL_SSL`
TRUSTSTORE_LOCATION	A path to the truststore file	—	Yes if `SECURITY_PROTOCOL = SASL_SSL`
TRUSTSTORE_PASSWORD	A password for the truststore file	—	Yes if `SECURITY_PROTOCOL = SASL_SSL`
KEYSTORE_LOCATION	A path to the keystore file	—	Yes if `SECURITY_PROTOCOL = SASL_SSL|SSL` (two-way authentication)
KEYSTORE_PASSWORD	A password for the keystore file	—	Yes if `SECURITY_PROTOCOL = SASL_SSL|SSL` (two-way authentication)
KERBEROS_SERVICE_NAME	A name of the Kerberos principal used by Kafka. Can be defined in the Kafka configuration file or JAAS configuration file	—	Yes if `SASL_MECHANISM = GSSAPI`

JAAS options for the SASL GSSAPI mechanism (Kerberos)

The following parameters are used if `SECURITY_PROTOCOL` is set to `SASL_PLAINTEXT` or `SASL_SSL`, and `SASL_MECHANISM` is set to `GSSAPI`. Since each parameter can depend on others, all of them are marked as optional.

NOTE

For more information on JAAS configuration, see JAAS Configuration in the Kafka documentation.

Name	Description	Default	Required
KERBEROS_PRINCIPAL	A name of the Kerberos principal that should be used. The principal can be a simple username such as `"testuser"` or a service name such as `"host/testhost.eng.sun.com"`	—	No
KERBEROS_KEYTAB	A file name of the keytab to get a principal secret key	—	No
KERBEROS_USE_KEYTAB	A flag that indicates whether to get a principal secret key from a keytab	true	No
KERBEROS_STORE_KEY	A flag that indicates whether to store a keytab or a principal key in the Subject private credentials	true	No
KERBEROS_USE_TICKET_CACHE	A flag that indicates whether to obtain a ticket-granting ticket (TGT) from the ticket cache	false	No
KERBEROS_DEBUG	A flag that indicates whether to output debug messages	false	No
JAAS_CONFIG_FILE	A path to the JAAS configuration file. Use if options present in the current table are not enough for a specified JAAS configuration. See details below	—	No

JAAS options for the SASL PLAIN, SCRAM-SHA-256, or SCRAM-SHA-512 mechanism

The following parameters are used if `SECURITY_PROTOCOL` is set to `SASL_PLAINTEXT` or `SASL_SSL`, and `SASL_MECHANISM` is set to `PLAIN`, `SCRAM-SHA-256`, or `SCRAM-SHA-512`.

Name	Description	Default	Required
SASL_USER	A user name for client connections	—	Yes if `SASL_MECHANISM=PLAIN|SCRAM-SHA-256|SCRAM-SHA-512`
SASL_USER_PASSWORD	A password of the `SASL_USER` user	—	Yes if `SASL_MECHANISM=PLAIN|SCRAM-SHA-256|SCRAM-SHA-512`
JAAS_CONFIG_FILE	A path to the JAAS configuration file. Use if options present in the current table are not enough for a specified JAAS configuration. See details below	—	No

JAAS_CONFIG_FILE

ADB to Kafka Connector uses Java Authentication and Authorization Service (JAAS) when connecting to Kafka via `SASL_PLAINTEXT` and `SASL_SSL` protocols. For JAAS configuration, the `sasl.jaas.config` parameter is used. There are two options to set up that parameter in the connector:

- Set options that are listed in the following tables: JAAS options for the SASL GSSAPI mechanism (Kerberos) and JAAS options for the SASL PLAIN mechanism (except `JAAS_CONFIG_FILE`).
- Save all options in the configuration file and specify its path in `JAAS_CONFIG_FILE`. If the `JAAS_CONFIG_FILE` parameter is used, other options from the abovementioned tables are ignored; all properties are read from the file.

It is recommended to use `JAAS_CONFIG_FILE` if additional options, that are not present in two abovementioned tables, are required for a specific JAAS configuration. The file, which path is set in the `JAAS_CONFIG_FILE` option, should contain only one line. File examples are listed below.

File example for SASL_MECHANISM = PLAIN

```
org.apache.kafka.common.security.plain.PlainLoginModule required username\="kafka_user" password\="kafka_password";
```

File example for SASL_MECHANISM = GSSAPI

```
com.sun.security.auth.module.Krb5LoginModule required principal="kafka_user@COMPANY.INTERNAL"
keyTab="/var/lib/pxf/conf/ssl/kafka_user.keytab" useKeyTab="true" storeKey="true" useTicketCache="false" debug="false";
```

File example for SASL_MECHANISM = SCRAM-SHA-256|SCRAM-SHA-512

```
org.apache.kafka.common.security.scram.ScramLoginModule required username="kafka_user" password="kafka_password";
```

Before starting queries to an external table, you should copy the file to each ADB segment host.

Use the SERVER option

By using `SERVER=<server_name>` in the external table creation query, you can read options from the configuration file without explicitly specifying them in the `LOCATION` clause. To use the `SERVER` option, follow the steps:

1. Add a new directory to *$PXF_BASE/servers/* on **coordinator**. The new directory name will be used as a server name (`<server_name>`) below:

```
$ sudo mkdir /var/lib/pxf/servers/<server_name>
```

NOTE

The directory, where PXF configuration files are located, is defined by the `PXF_BASE` environment variable. The default value is */var/lib/pxf*. The path may differ in your environment.

2. Create the *kafka-site.xml* configuration file in the */var/lib/pxf/servers/<server_name>* directory:

```
$ sudo vi /var/lib/pxf/servers/<server_name>/kafka-site.xml
```

The example of *kafka-site.xml* is listed below.

kafka-site.xml example

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>kafka.bootstrap.servers</name>
        <value>bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092</value>
    </property>
    <property>
        <name>kafka.batch.size</name>
        <value>10</value>
    </property>
    <property>
        <name>kafka.topic.auto.create</name>
        <value>true</value>
    </property>
    <property>
        <name>avro.decimal.default.precision</name>
        <value>38</value>
    </property>
    <property>
        <name>avro.decimal.default.scale</name>
        <value>18</value>
    </property>
</configuration>
```

3. In the *kafka-site.xml* file, specify configuration parameters that will be used to connect to Kafka. Available parameters are listed below. For each of them, the corresponding option from the abovementioned query is specified. Mandatory requirements and default values remain the same.

kafka-site.xml parameters

kafka-site.xml parameter	Option in the LOCATION query clause
kafka.bootstrap.servers	BOOTSTRAP_SERVERS
kafka.property.security.protocol	SECURITY_PROTOCOL
kafka.property.sasl.mechanism	SASL_MECHANISM
kafka.property.ssl.truststore.location	TRUSTSTORE_LOCATION
kafka.property.ssl.truststore.password	TRUSTSTORE_PASSWORD
kafka.property.ssl.keystore.location	KEYSTORE_LOCATION
kafka.property.ssl.keystore.password	KEYSTORE_PASSWORD
kafka.property.sasl.kerberos.service.name	KERBEROS_SERVICE_NAME
kafka.jaas.property.sasl.kerberos.principal	KERBEROS_PRINCIPAL
kafka.jaas.property.sasl.kerberos.keytab	KERBEROS_KEYTAB
kafka.jaas.property.sasl.kerberos.useKeyTab	KERBEROS_USE_KEYTAB
kafka.jaas.property.sasl.kerberos.storeKey	KERBEROS_STORE_KEY
kafka.jaas.property.sasl.kerberos.useTicketCache	KERBEROS_USE_TICKET_CACHE
kafka.jaas.property.sasl.kerberos.debug	KERBEROS_DEBUG
kafka.jaas.property.sasl.user	SASL_USER
kafka.jaas.property.sasl.user.password	SASL_USER_PASSWORD
kafka.jaas.property.config.file	JAAS_CONFIG_FILE
kafka.batch.size	BATCH_SIZE
kafka.admin.close.connection.timeout	KAFKA_ADMIN_CONNECTION_TIMEOUT
kafka.topic.auto.create	TOPIC_AUTO_CREATE_FLAG
avro.decimal.default.precision	AVRO_DEFAULT_DECIMAL_PRECISION
avro.decimal.default.scale	AVRO_DEFAULT_DECIMAL_SCALE

4. Log in under the default user `gpadmin`. All the commands listed in the subsequent steps should be performed on **coordinator**:

```
$ sudo su - gpadmin
```

5. Synchronize the PXF configuration on all hosts in the ADB cluster:

```
$ pxf cluster sync
```

Result:

```
Syncing PXF configuration files from coordinator host to standby coordinator host and 2 segment hosts...
PXF configs synced successfully on 3 out of 3 hosts
```

6. Restart PXF:

```
$ pxf cluster restart
```

Result:

```
Restarting PXF on coordinator host, standby coordinator host, and 2 segment hosts...
PXF restarted successfully on 4 out of 4 hosts
```

7. When creating an external table, specify `SERVER=<server_name>` in the `LOCATION` clause. Do not fill in other options:

```
CREATE WRITABLE EXTERNAL TABLE <table_name> (
    { <column_name> <data_type> [, ...] | LIKE <other_table> }
)
LOCATION (
    'pxf://<kafka_topic>?PROFILE=kafka&SERVER=<server_name>'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

Use custom options

In addition to ADB to Kafka Connector options mentioned above, you can define any other property available for a Kafka producer. There are two ways to set custom options:

- Define Kafka properties in the `LOCATION` clause when creating an external table. For example:

```
CREATE WRITABLE EXTERNAL TABLE test1 (a INT, b TEXT)
  LOCATION ('pxf://test_topic?PROFILE=kafka&BOOTSTRAP_SERVERS=bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.
internal:9092,bds-ads3.ru-central1.internal:9092&kafka.property.delivery.timeout.ms=131000')
  FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

- Add Kafka properties to the *kafka-site.xml* file for a server that will be specified in the SERVER option. For example:

```
<property>
    <name>kafka.property.delivery.timeout.ms</name>
    <value>131000</value>
</property>
```

IMPORTANT

When setting custom options, you should add the `kafka.property.` prefix to property names from Kafka documentation. Due to this, the `delivery.timeout.ms` Kafka property is called `kafka.property.delivery.timeout.ms` in the examples above.

Custom configuration of the SASL authentication mechanism

If your Kafka broker is configured to authenticate clients with the authentication mechanism that is absent in the Kafka options list (see `SASL_MECHANISM`), you can try to configure the plugin with the custom properties and JAAS configuration file (if required). For example, you would like to use SASL/OAUTHBEARER authentication that is not in the list. Do the following:

1. Set `SECURITY_PROTOCOL` to `SASL_SSL` or `SASL_PLAINTEXT` depending on the Kafka broker configuration.
2. Set `SASL_MECHANISM` to `OAUTHBEARER`.
3. Set required Kafka clients properties that are needed for this authentication. See Use custom options for information.
4. Create a JAAS configuration file (see JAAS_CONFIG_FILE) and copy it to the same directory on each segment host. For example, you can copy it to the folder with the *kafka-site.xml* file.
5. Set `JAAS_CONFIG_FILE` to the value which is an absolute path of the file from the previous point.
6. Create a table using the following query:

```
CREATE WRITABLE EXTERNAL TABLE kafka_tbl_oauthbearer (a INT, b TEXT, c TEXT)
LOCATION ('pxf://pxf-sasl-oauthbearer?PROFILE=kafka&BOOTSTRAP_SERVERS=bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.
internal:9092,bds-ads3.ru-central1.
internal:9092&SECURITY_PROTOCOL=SASL_SSL&SASL_MECHANISM=OAUTHBEARER&JAAS_CONFIG_FILE=/var/lib/pxf/servers/kafka/oauth.config&kafka.
property.sasl.oauthbearer.token.endpoint.url=http://localhost:8080/token&kafka.property.sasl.oauthbearer.scope.claim.name=kafka-
pxf')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

Alternatively, you can use the kafka-site.xml file. From the example above, the file would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property>
        <name>kafka.bootstrap.servers</name>
        <value>bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092</value>
    </property>
    <property>
        <name>kafka.property.security.protocol</name>
        <value>SASL_SSL</value>
    </property>
    <property>
        <name>kafka.property.sasl.mechanism</name>
        <value>OAUTHBEARER</value>
    </property>
    <property>
        <name>kafka.jaas.property.config.file</name>
        <value>/var/lib/pxf/servers/kafka/oauth.config</value>
    </property>
    <property>
        <name>kafka.property.sasl.oauthbearer.token.endpoint.url</name>
        <value>http://localhost:8080/token</value>
    </property>
    <property>
        <name>kafka.property.sasl.oauthbearer.scope.claim.name</name>
        <value>kafka-pxf</value>
    </property>
</configuration>
```

In case of saving the file in the */var/lib/pxf/servers/kafka_server* directory, the table definition would look like this:

```
CREATE WRITABLE EXTERNAL TABLE kafka_tbl_oauthbearer (a INT, b TEXT, c TEXT)
LOCATION ('pxf://pxf-sasl-oauthbearer?PROFILE=kafka&SERVER=kafka_server')
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

Greenplum and AVRO data types

The following table describes how Greenplum data types are converted into AVRO types when the ADB to Kafka Connector is used.

Greenplum type	AVRO primitive type	AVRO logical type
BOOLEAN	BOOLEAN	—
TEXT	STRING	—
VARCHAR	STRING	—
TIMESTAMP	LONG	timestamp-micros
BIGINT	LONG	—
TIME	LONG	time-micros
NUMERIC	DOUBLE	—
FLOAT8	DOUBLE	—
REAL	FLOAT	—
SMALLINT	INT	—
INTEGER	INT	—
DATE	INT	date

ADB to Kafka Connector usage examples

Daria Barysheva

Contents

To Table of Contents

- PLAINTEXT security protocol (by default)
  - Without SERVER option
  - With SERVER option
- SASL_PLAINTEXT security protocol
  - PLAIN mechanism
  - GSSAPI mechanism (Kerberos)
- SASL_SSL security protocol with GSSAPI mechanism (Kerberos)

This article describes how to transfer data from ADB to Kafka (as a part of ADS) via ADB to Kafka Connector. The following prerequisites are met:

- The ADS cluster is installed according to the Online installation guide on the following hosts: `bds-ads1.ru-central1.internal`, `bds-ads2.ru-central1.internal`, `bds-ads3.ru-central1.internal`.
- The ADB cluster is installed according to the Online installation guide on the following hosts: `bds-mdw.ru-central1.internal`, `bds-sdw1.ru-central1.internal`, `bds-sdw2.ru-central1.internal`.
- In ADB, the ***PXF*** and ***ADB to Kafka*** services are added and installed.
- The `adb` database exists in ADB.
- All ADB nodes have access to the `9092` port on all ADS nodes through which you plan to load data.

> NOTE
>
> Since ADB to Kafka Connector uses the AVRO OCF format for data serialization, you cannot deserialize data using *kafka-avro-console-consumer*. In all examples mentioned above, the *kafka-console-consumer.sh* script is used, which returns the schema of data obtained from ADB and the data itself in the HEX representation. Deserialization can be done with the Kafka to ADB Connector (see Kafka to ADB Connector usage examples).

PLAINTEXT security protocol (by default)

Without SERVER option

1. Connect to any ADS host where the Kafka service is installed. Go to the root directory and create a test topic via the following command:

```
$ bin/kafka-topics.sh --create --topic topic_adb_to_kafka --bootstrap-server bds-ads1.ru-central1.internal:9092,bds-ads2.ru-
central1.internal:9092,bds-ads3.ru-central1.internal:9092
```

Result:

```
Created topic topic_adb_to_kafka.
```

> TIP
>
> - For more information on adding and reading Kafka topics, see Quick start with Kafka in the ADS documentation.
> - Topic creation is optional. If you specify a non-existent topic in step 3, it will be created automatically unless the TOPIC_AUTO_CREATE_FLAG value is changed to `false`.

2. Connect to the `adb` database on ADB coordinator under the `gpadmin` user (for example, via `psql`):

```
$ sudo su - gpadmin
```

```
$ psql adb
```

3. Create an external table with the following structure:

```
CREATE WRITABLE EXTERNAL TABLE ext_adb_to_kafka (a INT, b TEXT)
LOCATION (
    'pxf://topic_adb_to_kafka?PROFILE=kafka&BOOTSTRAP_SERVERS=bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.
internal:9092,bds-ads3.ru-central1.internal:9092'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

4. Insert some data into the external table:

```
INSERT INTO ext_adb_to_kafka VALUES
(1,'test1'),
(2,'test2'),
(3,'test3'),
(4,'test4'),
(5,'test5');
```

5. In ADS, check that data from ADB has been successfully added. To do this, on the host where Kafka is installed, go to the root directory and run *bin/kafka-console-consumer.sh*:

```
$ bin/kafka-console-consumer.sh --topic topic_adb_to_kafka --from-beginning --bootstrap-server bds-ads1.ru-central1.
internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092 -value-deserializer org.apache.kafka.common.
serialization.BytesDeserializer --property print.key=false --property print.value=true --property print.offset=true
```

The command output contains information about 5 messages (with offset number):

```
Offset:0        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xA3\xFB\xC7\xF5n\x81\x8E\x96\x16\xEA\xF3\xE4s\x06P\x8E\x02\x12\x02\x02\x02\x0Atest1\xA3\xFB\xC7
\xF5n\x81\x8E\x96\x16\xEA\xF3\xE4s\x06P\x8E
Offset:1        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\x94-;\xFE\x15C}\x93\xEC\xACwjs[J\xF4\x02\x12\x02\x04\x02\x0Atest2\x94-;\xFE\x15C}\x93\xEC\xACwj
s[J\xF4
Offset:2        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00H\xD4N\xFAU_\x05\x01\xBAXv\xC7\xE8\xED\xFCr\x02\x12\x02\x0A\x02\x0Atest5H\xD4N\xFAU_\x05\x01\xBA
Xv\xC7\xE8\xED\xFCr
Offset:3        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xDD\xF6\x82r&G\xD4l\x1A\x00\x81\xE3^\x06\xE2\x13\x02\x12\x02\x06\x02\x0Atest3\xDD\xF6\x82r&G\xD
4l\x1A\x00\x81\xE3^\x06\xE2\x13
Offset:4        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xCD\xCCY\xC7\xA9#\xED\xB9K\xE8\xB8\x97i)\x809\x02\x12\x02\x08\x02\x0Atest4\xCD\xCCY\xC7\xA9#\xE
D\xB9K\xE8\xB8\x97i)\x809
^CProcessed a total of 5 messages
```

With SERVER option

1. Perform steps 1—6 that are described in the Use the SERVER option section of the ***ADB to Kafka Connector configuration*** article. As `<server_name>`, use `kafka_test`.
2. In ADB, create an external table `ext_adb_to_kafka2`. Unlike the table from the previous example, specify the `SERVER` option:

```
CREATE WRITABLE EXTERNAL TABLE ext_adb_to_kafka2 (a INT, b TEXT)
LOCATION (
    'pxf://topic_adb_to_kafka?PROFILE=kafka&SERVER=kafka_test'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

3. Insert some data into the `ext_adb_to_kafka2` table:

```
INSERT INTO ext_adb_to_kafka2 VALUES
(1,'test1');
```

4. In ADS, check that data from ADB has been successfully added to the `topic_adb_to_kafka` topic:

```
$ bin/kafka-console-consumer.sh --topic topic_adb_to_kafka --from-beginning --bootstrap-server bds-ads1.ru-central1.
internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092 -value-deserializer org.apache.kafka.common.
serialization.BytesDeserializer --property print.key=false --property print.value=true --property print.offset=true
```

The command output contains 6 messages:

```
Offset:0        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xA3\xFB\xC7\xF5n\x81\x8E\x96\x16\xEA\xF3\xE4s\x06P\x8E\x02\x12\x02\x02\x02\x0Atest1\xA3\xFB\xC7
\xF5n\x81\x8E\x96\x16\xEA\xF3\xE4s\x06P\x8E
Offset:1        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\x94-;\xFE\x15C}\x93\xEC\xACwjs[J\xF4\x02\x12\x02\x04\x02\x0Atest2\x94-;\xFE\x15C}\x93\xEC\xACwj
s[J\xF4
Offset:2        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00H\xD4N\xFAU_\x05\x01\xBAXv\xC7\xE8\xED\xFCr\x02\x12\x02\x0A\x02\x0Atest5H\xD4N\xFAU_\x05\x01\xBA
Xv\xC7\xE8\xED\xFCr
Offset:3        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xDD\xF6\x82r&G\xD4l\x1A\x00\x81\xE3^\x06\xE2\x13\x02\x12\x02\x06\x02\x0Atest3\xDD\xF6\x82r&G\xD
4l\x1A\x00\x81\xE3^\x06\xE2\x13
Offset:4        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xCD\xCCY\xC7\xA9#\xED\xB9K\xE8\xB8\x97i)\x809\x02\x12\x02\x08\x02\x0Atest4\xCD\xCCY\xC7\xA9#\xE
D\xB9K\xE8\xB8\x97i)\x809
Offset:5        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":["null","string"],"doc":""}]}\x00%\xA7\xB4\x18
\xE3\x8C\xCDx\xB4\xEE\xE4'Cjs\x02\x12\x02\x02\x02\x0Atest1%\xA7\xB4\x18 \xE3\x8C\xCDx\xB4\xEE\xE4'Cjs
^CProcessed a total of 6 messages
```

SASL_PLAINTEXT security protocol

PLAIN mechanism

1. Connect to any ADS host where the Kafka service is installed. Configure SASL_PLAINTEXT authentication for ADS cluster according to the SASL PLAINTEXT article.
2. Ensure that Kafka authentication is configured correctly by following steps from the Check the installed SASL PLAINTEXT section.
3. Create a new user `adb-to-kafka` with password `123` in ADS by following steps from the Create a user section.
4. According to the User authentication in the Kafka environment section, update the */etc/kafka/conf/client.properties* file contents:

```
security.protocol=SASL_PLAINTEXT
sasl.mechanism=PLAIN
# Uncomment and set necessary username/password
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \
  username="adb-to-kafka" \
  password="123";
```

5. Connect to the `adb` database on ADB coordinator under the `gpadmin` user (for example, via `psql`):

```
$ sudo su - gpadmin
```

```
$ psql adb
```

6. Create an external table with the following structure in ADB:

```
CREATE WRITABLE EXTERNAL TABLE ext_adb_to_kafka_sasl (a INT, b TEXT)
LOCATION (
    'pxf://topic_adb_to_kafka_sasl?PROFILE=kafka&BOOTSTRAP_SERVERS=bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.
internal:9092,bds-ads3.ru-central1.internal:9092&SECURITY_PROTOCOL=SASL_PLAINTEXT&SASL_MECHANISM=PLAIN&SASL_USER=adb-to-
kafka&SASL_USER_PASSWORD=123'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

7. Insert some data into the `ext_adb_to_kafka_sasl` external table:

```
INSERT INTO ext_adb_to_kafka_sasl VALUES
(1,'test1');
```

8. In ADS, check that data from ADB has been successfully added. To do this, on the host where Kafka is installed, go to the root directory and run *bin/kafka-console-consumer.sh*.

> NOTE
>
> Unlike the examples mentioned above, when using the SASL_PLAINTEXT protocol, you should fill in the `--consumer.config` option. Use it to specify a path to the *client.properties* file described in step 4 above.

```
$ bin/kafka-console-consumer.sh --topic topic_adb_to_kafka_sasl --from-beginning --bootstrap-server bds-ads1.ru-central1.
internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092 -value-deserializer org.apache.kafka.common.
serialization.BytesDeserializer --property print.key=false --property print.value=true --property print.offset=true --consumer.
config /etc/kafka/conf/client.properties
```

The command output contains one message:

```
Offset:0        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00c\xA9\x9A\xC6ZoI\xA2br\xCD\xFBi=@\x81\x02\x12\x02\x02\x02\x0Atest1c\xA9\x9A\xC6ZoI\xA2br\xCD\xFB
i=@\x81
^CProcessed a total of 1 messages
```

Authentication errors

If user credentials are not specified correctly in the external table definition, the following error occurs when you try to add data to that table:

```
ERROR:  PXF server error : Cannot acquire list of topics for servers: bds-ads1.ru-central1.internal:9092,bds-ads2.ru-
central1.internal:9092,bds-ads3.ru-central1.internal:9092. Details: org.apache.kafka.common.errors.
SaslAuthenticationException: Authentication failed: Invalid username or password  (seg0 10.92.38.140:10000 pid=2356853)
HINT:  Check the PXF logs located in the '/var/lib/pxf/logs' directory on host '0.0.0.0' or 'set client_min_messages=LOG'
for additional details.
```

For more details, you can view the */var/lib/pxf/logs/pxf-service.log* file on the specified segment host under the `gpadmin` user:

```
$ sudo su - gpadmin
$ tail /var/lib/pxf/logs/pxf-service.log
```

GSSAPI mechanism (Kerberos)

> CAUTION
>
> If you configured SASL_PLAINTEXT authentication for your ADS cluster according to the SASL PLAINTEXT article (e.g. when running the previous example), you should disable it before ADS kerberization. To do this, deactivate the ***SASL_PLAINTEXT Authentication*** switcher on the ADS cluster configuration page and apply changes by clicking ***Save***.

1. Configure MIT Kerberos KDC on a separate host (`bds-mit-kerberos.ru-central1.internal` in the given example) and kerberize your ADS cluster according to the MIT Kerberos article. As a realm name, use `ADS-KAFKA.LOCAL`.
2. Ensure that Kafka authentication is configured correctly by following steps from the Check installed Kerberos SASL section.
3. On the host where MIT Kerberos KDC is deployed, create a new principal `adb-to-kafka`, which will be used by ADB to Kafka Connector. Specify `123` as a password:

```
$ sudo kadmin.local -q "add_principal -pw 123 adb-to-kafka@ADS-KAFKA.LOCAL"
```

Result:

```
Principal "adb-to-kafka@ADS-KAFKA.LOCAL" created.
```

4. According to the Create a configuration file .properties for the user section, create the */tmp/client.properties* file in ADS.
5. Follow the steps from the Create a JAAS file for a user section.
6. In ADB, install the following packages on coordinator and segment hosts:

```
$ sudo apt install krb5-user sssd-krb5
```

7. On ADB coordinator and each segment host, edit the */etc/krb5.conf* file contents in the same way as you configured it for MIT Kerberos KDC:

```
$ sudo vi /etc/krb5.conf
```

The file contents:

```
[libdefaults]
        default_realm = ADS-KAFKA.LOCAL
        kdc_timesync = 1
        ccache_type = 4
        forwardable = true
        proxiable = true
        fcc-mit-ticketflags = true

[realms]
        ADS-KAFKA.LOCAL = {
                kdc = bds-mit-kerberos.ru-central1.internal
                admin_server = bds-mit-kerberos.ru-central1.internal
        }
```

8. On ADB coordinator, create a new directory to store keytab files. Assign an owner and necessary permissions to that directory:

```
$ sudo mkdir /var/lib/pxf/conf/kerberos
$ sudo chmod 744 /var/lib/pxf/conf/kerberos
$ sudo chown gpadmin:gpadmin /var/lib/pxf/conf/kerberos
```

9. Connect to ADB coordinator under the `gpadmin` user:

```
$ sudo su - gpadmin
```

10. On ADB coordinator, run the interactive utility `ktutil`:

```
$ ktutil
```

11. In the console that opens, run the following commands one by one (after the `ktutil:` prompt). After running each command, you should specify a password for the `adb-to-kafka` principal (`123` in the given example):

```
add_entry -password -p adb-to-kafka@ADS-KAFKA.LOCAL -k 1 -e aes128-cts-hmac-sha1-96
add_entry -password -p adb-to-kafka@ADS-KAFKA.LOCAL -k 1 -e aes256-cts-hmac-sha1-96
add_entry -password -p adb-to-kafka@ADS-KAFKA.LOCAL -k 1 -e arcfour-hmac
add_entry -password -p adb-to-kafka@ADS-KAFKA.LOCAL -k 1 -e camellia256-cts-cmac
add_entry -password -p adb-to-kafka@ADS-KAFKA.LOCAL -k 1  -e camellia128-cts-cmac
```

12. As the last command, send the following request to store a keytab file with the entered data. Then, type `q` to quit `ktutil`:

```
write_kt /var/lib/pxf/conf/kerberos/adb-to-kafka.service.keytab
```

13. To copy *adb-to-kafka.service.keytab* to all ADB segment hosts, run the following commands on coordinator in the given order:
    - PXF synchronization:

```
$ pxf cluster sync
```

    - PXF restart:

```
$ pxf cluster restart
```

14. Check the `adb-to-kafka` principal by sequentially running the following commands. When running the `kinit` command, you need to specify the principal password (`123` in the given example):

```
$ kinit adb-to-kafka@ADS-KAFKA.LOCAL
```

```
$ klist
```

The result indicates that a valid ticket is present:

```
Ticket cache: FILE:/tmp/krb5cc_2042
Default principal: adb-to-kafka@ADS-KAFKA.LOCAL

Valid starting       Expires              Service principal
11/29/2025 18:53:28  11/30/2025 18:53:28  krbtgt/ADS-KAFKA.LOCAL@ADS-KAFKA.LOCAL
```

15. Connect to the `adb` database on ADB coordinator (for example, via `psql`):

```
$ psql adb
```

16. Create an external table with the following structure in ADB:

```
CREATE WRITABLE EXTERNAL TABLE ext_adb_to_kafka_kerberos (a INT, b TEXT)
LOCATION (
    'pxf://topic_adb_to_kafka_kerberos?PROFILE=kafka&BOOTSTRAP_SERVERS=bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.
internal:9092,bds-ads3.ru-central1.
internal:9092&SECURITY_PROTOCOL=SASL_PLAINTEXT&SASL_MECHANISM=GSSAPI&KERBEROS_KEYTAB=/var/lib/pxf/conf/kerberos/adb-to-kafka.
service.keytab&KERBEROS_PRINCIPAL=adb-to-kafka&KERBEROS_SERVICE_NAME=kafka'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

17. Insert some data into the `ext_adb_to_kafka_kerberos` table:

```
INSERT INTO ext_adb_to_kafka_kerberos VALUES
(1,'kerberos');
```

18. In ADS, check that data from ADB has been successfully added. To do this, on the host where Kafka is installed, go to the root directory and run the following commands. When running the `kinit` command, specify a password for the `adb-to-kafka` principal (`123` in the given example):

```
$ kinit adb-to-kafka@ADS-KAFKA.LOCAL
$ export KAFKA_OPTS="-Djava.security.auth.login.config=/tmp/client.jaas"
$ bin/kafka-console-consumer.sh --topic topic_adb_to_kafka_kerberos --from-beginning --bootstrap-server bds-ads1.ru-central1.
internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092 -value-deserializer org.apache.kafka.common.
serialization.BytesDeserializer --property print.key=false --property print.value=true --property print.offset=true --consumer.
config /tmp/client.properties
```

The command output contains one message:

```
Offset:0        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":
["null","string"],"doc":""}]}\x00\xA6qBp\xF5;Q8\x035<\xFDB\xBF\xC6\xB3\x02\x18\x02\x02\x02\x10kerberos\xA6qBp\xF5;Q8\x035<\xFDB\x
BF\xC6\xB3
^C[2023-12-14 18:18:35,250] WARN [Principal=null]: TGT renewal thread has been interrupted and will exit. (org.apache.kafka.
common.security.kerberos.KerberosLogin)
Processed a total of 1 messages
```

> NOTE
>
> In the `--consumer.config` option, specify a path to the *client.properties* file that is described in step 4 above.

Authentication errors

If a Kerberos principal or keytab path are not specified correctly in the external table definition, the following error occurs when you try to add data to that table:

```
ERROR:  PXF server error : Failed to create new KafkaAdminClient  (seg0 10.92.38.140:10000 pid=2404402)
HINT:  Check the PXF logs located in the '/var/lib/pxf/logs' directory on host '0.0.0.0' or 'set client_min_messages=LOG'
for additional details.
```

Details can be found in PXF logs.

SASL_SSL security protocol with GSSAPI mechanism (Kerberos)

> NOTE
>
> Actions with certificates that are described in steps 2—8 can be automated via a bash script (see *generate.sh* in SSL channel security). However, the script is not recommended for a production environment.

1. Perform steps 1—14 that are described in the GSSAPI mechanism (Kerberos) section (if you have not passed them yet). Ensure that a valid ticket exists for the `adb-to-kafka` principal on the ADB side.
2. Generate an RSA key pair and create *keystore.jks* on each ADB and ADS host:

```
$ keytool -genkeypair -noprompt -keyalg RSA -ext san=dns:<hostname> -alias <hostname> -dname "CN=<hostname>, OU=AD, O=AD, L=MSK,
S=MO, C=RU" -keystore /tmp/keystore.jks -storepass bigdata -keypass bigdata -validity 360 -keysize 2048
```

where `<hostname>` — FQDN of the host where the command runs.

3. To export SSL certificates, run the following command on each ADB and ADS host:

```
$ keytool -exportcert -file /tmp/<hostname>.crt -keystore /tmp/keystore.jks -storepass bigdata -alias <hostname> -rfc
```

where `<hostname>` — FQDN of the host where the command runs.

The result is below (for the host with FQDN `bds-mdw.ru-central1.internal`):

```
Certificate stored in file </tmp/bds-mdw.ru-central1.internal.crt>
```

4. Copy certificates from each ADB and ADS host to the local machine. The following is an example for Windows:

```
$ pscp <user>@<hostname>:/tmp/<hostname>.crt /<local_folder>/
```

where:

- `<user>` — a username that is used to connect to the ADB/ADS host via SSH;
- `<hostname>` — an FQDN of the ADB/ADS host;
- `<local_folder>` — a directory that is created on the local machine to store all certificates.

For example, for FQDN `bds-mdw.ru-central1.internal` the command can look like this:

```
$ pscp dasha@bds-mdw:/tmp/bds-mdw.ru-central1.internal.crt /cert/
```

As a result, all certificates are stored in one place:

```
    Directory: C:\cert


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/25/2023  10:58 AM           1275 bds-ads1.ru-central1.internal.crt
-a----        12/25/2023  10:58 AM           1275 bds-ads2.ru-central1.internal.crt
-a----        12/25/2023  10:59 AM           1275 bds-ads3.ru-central1.internal.crt
-a----        12/25/2023  10:58 AM           1271 bds-mdw.ru-central1.internal.crt
-a----        12/25/2023  10:58 AM           1275 bds-sdw1.ru-central1.internal.crt
-a----        12/25/2023  10:58 AM           1275 bds-sdw2.ru-central1.internal.crt
```

5. Copy all certificates from the local machine to the */tmp* directory of each ADB and ADS host. To do this, run the following command for each host:

```
$ pscp /<local_folder>/*.crt <user>@<hostname>:/tmp/
```

For example, for ADB host with FQDN `bds-mdw.ru-central1.internal` the command can look like this:

```
$ pscp /cert/*.crt dasha@bds-mdw:/tmp/
```

6. Import all certificates into *truststore.jks* and *ca-bundle.crt* on each ADB/ADS host as follows:

```
$ keytool -importcert -noprompt -alias <hostname> -file /tmp/<hostname>.crt -keystore /tmp/truststore.jks -storepass bigdata
$ sudo bash -c "cat /tmp/<hostname>.crt >> /etc/ssl/certs/ca-bundle.crt"
```

where `<hostname>` — FQDN of each ADB/ADS host. In the given example, two commands listed above should be run 6 times on each of 6 ADB/ADS hosts (for each certificate).

7. Import *truststore.jks* to Java CA store by running the following command on each ADB/ADS host:

```
$ sudo keytool -importkeystore -noprompt -srckeystore /tmp/truststore.jks -destkeystore /etc/ssl/certs/java/cacerts -deststorepass
changeit -srcstorepass bigdata
```

Result:

```
Importing keystore /tmp/truststore.jks to /etc/ssl/certs/java/cacerts...
Entry for alias bds-mdw.ru-central1.internal successfully imported.
Entry for alias bds-sdw1.ru-central1.internal successfully imported.
Entry for alias bds-sdw2.ru-central1.internal successfully imported.
Entry for alias bds-ads1.ru-central1.internal successfully imported.
Entry for alias bds-ads2.ru-central1.internal successfully imported.
Entry for alias bds-ads3.ru-central1.internal successfully imported.
Import command completed:  6 entries successfully imported, 0 entries failed or cancelled
```

8. Create and import OpenSSL certificates by running the following commands on each ADB/ADS host:

```
$ sudo openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 -subj "/C=RU/ST=Denial/L=MSK/O=AD/CN=<hostname>" -keyout
/etc/ssl/host_cert.key  -out /etc/ssl/certs/host_cert.cert
$ sudo bash -c "cat /etc/ssl/certs/host_cert.cert >> /etc/ssl/certs/ca-bundle.crt"
```

where `<hostname>` — FQDN of the host where commands run.

9. Enable SSL in ADS according to the Start SSL on an ADS cluster using ADCM section.
10. Ensure that Kafka authentication is configured correctly by following steps from the Check the installed SSL section.
11. In ADS, create a new file */tmp/client_ssl.properties*:

```
$ sudo vi /tmp/client_ssl.properties
```

Add the file contents:

```
security.protocol=SASL_SSL
sasl.mechanism=GSSAPI
sasl.kerberos.service.name=kafka
```

12. Connect to the `adb` database on ADB coordinator under the `gpadmin` user (for example, via `psql`):

```
$ sudo su - gpadmin
```

```
$ psql adb
```

13. Create an external table with the following structure in ADB:

```
CREATE WRITABLE EXTERNAL TABLE ext_adb_to_kafka_ssl (a INT, b TEXT)
LOCATION (
    'pxf://topic_ssl?PROFILE=kafka&BOOTSTRAP_SERVERS=bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-
ads3.ru-central1.internal:9092&SECURITY_PROTOCOL=SASL_SSL&SASL_MECHANISM=GSSAPI&KERBEROS_KEYTAB=/var/lib/pxf/conf/kerberos/adb-to-
kafka.service.keytab&KERBEROS_PRINCIPAL=adb-to-kafka&KERBEROS_SERVICE_NAME=kafka&TRUSTSTORE_LOCATION=/tmp/truststore.
jks&TRUSTSTORE_PASSWORD=bigdata'
)
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_export');
```

14. Insert some data into the `ext_adb_to_kafka_ssl` table:

```
INSERT INTO ext_adb_to_kafka_ssl VALUES
(1,'ssl-test-data');
```

15. In ADS, check that data from ADB has been successfully added. To do this, on the host where Kafka is installed, go to the root directory and run the following commands. When running the `kinit` command, specify a password for the `adb-to-kafka` principal (`123` in the given example):

```
$ kinit adb-to-kafka@ADS-KAFKA.LOCAL
$ export KAFKA_OPTS="-Djava.security.auth.login.config=/tmp/client.jaas"
$ bin/kafka-console-consumer.sh --topic topic_ssl --from-beginning --bootstrap-server bds-ads1.ru-central1.internal:9092,bds-ads2.
ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092 -value-deserializer org.apache.kafka.common.serialization.
BytesDeserializer --property print.key=false --property print.value=true --property print.offset=true --consumer.config
/tmp/client_ssl.properties
```

The command output contains one message:

```
Offset:0        Obj\x01\x02\x16avro.schema\x88\x02{"type":"record","name":"row","fields":[{"name":"a","type":
["null","int"],"doc":""},{"name":"b","type":["null","string"],"doc":""}]}\x00 1I\xCDJ6|U\xBEw\x14\xD5\xA8\x5C>?
\x02"\x02\x02\x02\x1Assl-test-data 1I\xCDJ6|U\xBEw\x14\xD5\xA8\x5C>?
^C[2023-12-26 08:30:14,242] WARN [Principal=null]: TGT renewal thread has been interrupted and will exit. (org.apache.kafka.
common.security.kerberos.KerberosLogin)
Processed a total of 1 messages
```

> NOTE
>
> In the `--consumer.config` option, specify a path to the *client_ssl.properties* file that is described in step 11 above.

Authentication errors

If connection parameters are not specified correctly in the external table definition, the following error occurs when you try to add data to that table:

```
ERROR:  PXF server error : Failed to create new KafkaAdminClient  (seg1 10.92.38.12:10000 pid=2845590)
HINT:  Check the PXF logs located in the '/var/lib/pxf/logs' directory on host '0.0.0.0' or 'set client_min_messages=LOG'
for additional details.
```

Details can be found in PXF logs.

Kafka to ADB Connector installation

Daria Barysheva

To Table of Contents

To install Kafka to ADB Connector, follow the steps:

1. Make sure that the Kafka to ADB service is installed. For information on how to add services to ADB clusters, see Add services.

2. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

3. Check that the new extension `kadb_fdw` is available — via the `psql` meta-command `\dx`:

```
\dx
```

   The output contains the `kadb_fdw` extension:

```
                                          List of installed extensions
       Name        | Version |   Schema   |                             Description
-------------------+---------+------------+----------------------------------------------------------------------
 adb_fdw           | 1.0     | public     | foreign-data wrapper for remote ADB servers
 arenadata_toolkit | 1.0     | public     | extension is used for manipulation of objects created by adb-bundle
 dblink            | 1.2     | public     | connect to other PostgreSQL databases from within a database
 gp_exttable_fdw   | 1.0     | pg_catalog | External Table Foreign Data Wrapper for Greengage
 gp_toolkit        | 1.8     | gp_toolkit | various GPDB administrative views/functions
 kadb_fdw          | 0.16    | kadb       | Kafka-ADB foreign data wrapper
 plpgsql           | 1.0     | pg_catalog | PL/pgSQL procedural language
 pxf               | 2.1     | public     | Extension which allows to access unmanaged data
 pxf_fdw           | 1.0     | public     | PXF Foreign Data Wrapper for Greengage
 tkh_fdw           | 1.0     | public     | ClickHouse Foreign Data Wrapper for Greenplum
(10 rows)
```

4. You can also check that the new foreign data wrapper `kadb_fdw` is available using the following SQL query against the `pg_catalog.pg_foreign_data_wrapper` system table:

```
SELECT * FROM pg_catalog.pg_foreign_data_wrapper;
```

   Result:

```
  oid  |     fdwname     | fdwowner | fdwhandler | fdwvalidator | fdwacl |                                         fdwoptions
-------+-----------------+----------+------------+--------------+--------+-------------------------------------------------------
-------------------------------------
 13702 | gp_exttable_fdw |       10 |      13700 |        13701 |        |
 17124 | dblink_fdw      |       10 |          0 |        17123 |        |
 17175 | adb_fdw         |       10 |      17173 |        17174 |        |
 17187 | jdbc_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=jdbc,"mpp_execute=all segments"}
 17188 | hdfs_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=hdfs,"mpp_execute=all segments"}
 17189 | hive_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=hive,"mpp_execute=all segments"}
 17190 | hbase_pxf_fdw   |       10 |      17185 |        17186 |        | {protocol=hbase,"mpp_execute=all segments"}
 17191 | s3_pxf_fdw      |       10 |      17185 |        17186 |        | {protocol=s3,"mpp_execute=all segments"}
 17192 | gs_pxf_fdw      |       10 |      17185 |        17186 |        | {protocol=gs,"mpp_execute=all segments"}
 17193 | adl_pxf_fdw     |       10 |      17185 |        17186 |        | {protocol=adl,"mpp_execute=all segments"}
 17194 | wasbs_pxf_fdw   |       10 |      17185 |        17186 |        | {protocol=wasbs,"mpp_execute=all segments"}
 17195 | file_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=file,"mpp_execute=all segments"}
 17201 | tkh_fdw         |       10 |      17200 |        17199 |        | {protocol=tkh_clickhouse,"mpp_execute=all
segments",format=text}
 17210 | kadb_fdw        |       10 |      17209 |        17211 |        | {"mpp_execute=all
segments",k_initial_offset=0,k_automatic_offsets=true,k_latency_ms=2000}
(14 rows)
```

5. Ensure that the kadb schema is available — via the `psql` meta-command `\dnS+`:

```
\dnS+
```

   Result:

```
        Name        |  Owner  | Access privileges  |                         Description
--------------------+---------+--------------------+-------------------------------------------------------------
 arenadata_toolkit  | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 diskquota          | gpadmin |                    |
 gp_toolkit         | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 information_schema | gpadmin | gpadmin=UC/gpadmin+|
                    |         | =U/gpadmin         |
 kadb               | gpadmin |                    |
 pg_aoseg           | gpadmin |                    | Reserved schema for Append Only segment list and eof tables
 pg_bitmapindex     | gpadmin |                    | Reserved schema for internal relations of bitmap indexes
 pg_catalog         | gpadmin | gpadmin=UC/gpadmin+| system catalog schema
                    |         | =U/gpadmin         |
 pg_toast           | gpadmin |                    | reserved schema for TOAST tables
 public             | gpadmin | gpadmin=UC/gpadmin+| standard public schema
                    |         | =UC/gpadmin        |
(10 rows)
```

Kafka to ADB Connector configuration

Daria Barysheva

Contents

To Table of Contents

Objects to work with a connector
  - Server
  - Foreign table

kadb schema
  - kadb.offsets table
  - Functions

Supported formats
  - AVRO OCF
  - CSV
  - Text

Implementation details
  - Concurrent SELECT and global deadlock detector
  - Distribution of partitions across segments
  - Query termination
  - Duration of SELECT queries

Objects to work with a connector

To send data from Kafka to ADB via Kafka to ADB Connector, you should first create the following objects on the ADB cluster side:

1. Server — encapsulates connection information that a foreign data wrapper uses to access an external data source.
2. Foreign table — a table in ADB that defines the remote data structure. A foreign table has no storage in ADB, but can be used in queries just like a normal table.

To see how to use the listed objects to send data from ADS to ADB, refer to Kafka to ADB Connector usage examples.

IMPORTANT

- For each Kafka cluster, it is sufficient to create one server. However, there may be several foreign tables if you need, for example, to read data from multiple Kafka topics.
- Options that are listed below can be defined both on the server and foreign table sides. Table-level options take precedence.
- Values of all options should be defined in the text format (i.e. in single quotes `'`).
- Option names that contain `#` or `.` should be surrounded with double quotation marks `"`. Otherwise, quotes are not required for option names.

Server

To create a server, use the `CREATE SERVER` command. The basic command syntax is listed below:

```
CREATE SERVER <server_name> [ TYPE '<server_type>' ] [ VERSION '<server_version>' ]
    FOREIGN DATA WRAPPER <fdw_name>
    [ OPTIONS ( [ <option> '<value>' [, ... ]] ) ]
```

where:

- `<server_name>` — a server name in ADB. Should be unique within the current ADB database.
- `<server_type>` — an optional server type.
- `<server_version>` — an optional server version.
- `<fdw_name>` — a foreign data wrapper name. You should use the `kadb_fdw` foreign data wrapper, which is automatically created after the connector installation (see step 4 in Kafka to ADB Connector installation).
- `<option>` — server options that define the connection details. The options that are available for Kafka to ADB Connector are listed in the Server options table. Note that these options can be defined both at the server level and at the foreign table level. Options marked as required should be specified at least in one definition (server or table). Table-level options take precedence.
- `<value>` — option values.

NOTE

- To create a server, you need the `USAGE` privilege on the `kadb_fdw` foreign data wrapper. The user who creates the server becomes its owner.
- The full description of the `CREATE SERVER` command syntax is available in the Greenplum documentation.
- To edit a server definition, use the ALTER SERVER command. To delete a server, use DROP SERVER.

Server options

Name	Description	Default	Required
k_brokers	A comma-separated list of Kafka brokers, each of which is specified in the following format: `<hostname>:<port>` or `<IP>:<port>`. Alias of the librdkafka property `bootstrap.servers`	—	Yes
k_topic	A Kafka topic identifier (see Storage concepts in Kafka)	—	Yes
k_consumer_group	A Kafka consumer group identifier. Alias of the librdkafka property `group.id`	—	Yes
k_seg_batch	A maximum number of Kafka messages that can be retrieved by each ADB segment. Note that when queries with `LIMIT` conditions are executed, messages are still requested from Kafka in batches. As a result, offsets in the kadb.offsets table are set to the latest offsets of retrieved messages for each partition even if data of these messages are not present in the query result. Since the `k_seg_batch` option limits the number of messages that can be read, there may be partitions from which no messages are retrieved by a single particular `SELECT` against a foreign table. Positive integers are allowed	—	Yes
k_timeout_ms	A timeout for a message consumption request to Kafka (in milliseconds). Only messages available in Kafka during the period from the start of `SELECT` to the end of `k_timeout_ms` will be retrieved and presented as the `SELECT` result. Each segment consumes messages from all partitions assigned to it simultaneously. Some `SELECT` queries can finish sooner if the specified Kafka partition contains enough messages (as soon as `k_seg_batch` messages are read by each segment). Most of all, the query duration is affected by the Kafka partitions, in which there are not enough messages: the connector expects new messages to appear in these partitions for `k_timeout_ms` milliseconds. Pay attention that the actual duration of each `SELECT` query can be longer than the `k_timeout_ms` value. At some stages of `SELECT` queries, you cannot terminate them until the `k_timeout_ms` interval expires. Non-negative integers are allowed	—	Yes
format	Data serialization format. Possible values (case-insensitive): - avro - csv - text	—	Yes
k_latency_ms	A timeout for metadata requests to Kafka. Set the value of this option to the maximum expected latency (across all ADB segments) of any metadata request to Kafka. The longest metadata request is made by the `kadb.offsets_to_committed(<OID>)` function — you can use it to determine the minimal `k_latency_ms` value. Note that there are several metadata queries when `SELECT` to a foreign table is executed. If the timeout is exceeded, the query fails with an error. Non-negative integers are allowed	2000	No
k_initial_offset	The initial offset that is used for partitions that have no corresponding entries in the kadb.offsets table yet. The option is used when the `k_automatic_offsets` flag is set and by offset management functions. Non-negative integers are allowed	0	No

k_automatic_offsets	A flag the `true` value of which means the following:
	- Immediately before each `SELECT` to a foreign table:
	  - Add partitions present in Kafka and absent in the kadb.offsets table to a set of partitions to read data from.
	  - Automatically increase an initial offset of any partition to the lowest (earliest) offset available in Kafka. A `NOTICE` occurs when such increase takes place.
	- Immediately after each `SELECT` to a foreign table:
	  - Update the kadb.offsets table by adding partitions present in Kafka and absent in ADB (via `INSERT`).
	If `k_automatic_offsets=false`, an `ERROR` is raised when the smallest offset of an existing message in Kafka is greater than the offset in the kadb.offsets table (for any partition).
	Note that partitions present in Kafka and absent in the kadb.offsets table are not visible for users if `CURSOR` is used: as mentioned above, data is inserted into kadb.offsets after `SELECT` execution, while `CURSOR` queries are constantly in progress.
	After a successful `SELECT` query, offsets in the kadb.offsets table are updated regardless the `k_automatic_offsets` value — to reflect the number of messages read from Kafka
		true	No

avro_schema	An AVRO schema that should be used for deserialization. Requires the JSON format. Incoming messages are deserialized by Kafka to ADB Connector in one of the following ways: - If the `avro_schema` option is set, the user-provided schema is used for deserialization (incoming messages should still be in the AVRO OCF format). - Otherwise, a schema is extracted directly from incoming messages in the AVRO OCF format. Note that a user-provided schema cannot be validated. If actual and provided schemas do not correspond, deserialization fails with the following error: `ERROR: invalid memory alloc request size`. Due to this, the `avro_schema` option should be used only for performance reasons and after careful analyze	—	No
csv_quote	A character that is used as a quotation mark when CSV is parsed. Use a single character represented by one byte in the current encoding	"	No
csv_delimeter	A character that is used as a delimiter when CSV is parsed. Use a single character represented by one byte in the current encoding	,	No
csv_null	A string that represents `NULL` values in CSV	Empty string	No
csv_ignore_header	A flag that indicates whether to ignore (do not parse) the first record of each CSV message, i.e. consider it as a header	false	—
csv_attribute_trim_whitespace	A flag that indicates whether to trim trailing whitespaces at the beginning and the end of each attribute (field) in CSV records	true	—

librdkafka properties

In addition to abovementioned options, when declaring a server and a foreign table, you can specify librdkafka properties.

A name of each `librdkafka` property should be started with `#` and enclosed into double quotes `"`. For example, `"#bootstrap.servers"`, `"#security.protocol"`, and so on.

librdkafka configuration properties take precedence over their aliases mentioned above.

The following `librdkafka` properties are forbidden to be set:

- `enable.auto.commit`;
- `enable.partition.eof`;
- `plugin.library.paths`;
- `interceptors`;
- all properties whose names end with `_cb`.

Foreign table

To create a foreign table, use the `CREATE FOREIGN TABLE` command. The basic command syntax is listed below:

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] <table_name> ( [
    <column_name> <data_type> [ COLLATE <collation> ] [ <column_constraint> [ ... ] ]
      [, ... ]
] )
    SERVER <server_name>
  [ OPTIONS ( <option> '<value>' [, ... ] ) ]
```

where:

- `<table_name>` — a foreign table name in ADB.
- `<column_name>` — a column name.
- `<data_type>` — a column data type.
- `<collation>` — a column collation.
- `<column_constraint>` — a constraint defined at the column level. The name `<constraint_name>` is given optionally. The syntax:

```
[ CONSTRAINT <constraint_name> ]
{ NOT NULL |
  NULL |
  DEFAULT <default_expr> }
```

  Possible constraints:

  - `NOT NULL` — specifies that the column is not allowed to contain null values.
  - `NULL` — specifies that the column is allowed to contain null values. It is the default behavior (unless `NOT NULL` is specified).
  - `DEFAULT` — sets the default column value equal to `<default_expr>`.

- `<server_name>` — a server name.
- `<option>` — foreign table options. For Kafka to ADB Connector, all options defined at the server level can be overwritten at the foreign table level (partially or fully). When you specify an option for a server and a foreign table at the same time, the table level takes precedence.
- `<value>` — option values.

NOTE

- To create a foreign table, you need the `USAGE` privilege on the corresponding server and on all column data types that are used in the table definition. The user who creates the foreign table becomes its owner.
- The full description of the `CREATE FOREIGN TABLE` command syntax is available in the Greenplum documentation.
- To edit a foreign table definition, use the ALTER FOREIGN TABLE command. To drop a foreign table, use DROP FOREIGN TABLE.

kadb schema

During the Kafka to ADB Connector installation, the `kadb` schema is added to the default ADB database. This schema contains the kadb.offsets table and a set of functions that are used to synchronize offsets.

kadb.offsets table

The `kadb.offsets` table stores partition/offset mappings for each foreign table that has ever been created in the current ADB database. The `kadb.offsets` table provides the ability to store partitions outside of Kafka — on the consumer side. The `kadb.offsets` table structure is listed below.

Column	Description
ftoid	A foreign table OID
prt	A Kafka partition identifier
off	An offset

OID is used to identify each foreign table. To define a foreign table OID, use the following query:

```
SELECT '<schema>.<table>'::regclass::oid;
```

where:

- `<schema>` — a name of the schema, in which a foreign table is created. Can be omitted if a schema search path is properly configured.
- `<table>` — a foreign table name.

On each `SELECT` query to a foreign table with the given OID, a request is issued to `kadb.offsets`. Then messages are read from Kafka starting with the offset specified in the `kadb.offsets` table for the selected table OID and partition. For example, if the offset for some partition is set to `15`, the message with offset `15` will be read first from this Kafka partition.

A set of partitions and their offsets can be changed via common SQL queries to `kadb.offsets`. Additionally, you can use special functions for this purpose.

For new partitions that are not yet present in the `kadb.offsets` table, the initial offset is set to `0` by default. You can change that value via the `k_initial_offset` option.

After a successful `SELECT` query against a foreign table, offsets are updated in the `kadb.offsets` table automatically based on the data obtained from Kafka. For example, if the last message read from some Kafka partition has offset `25`, the `kadb.offsets.off` column for the selected partition and foreign table will contain `26`. And the next time the data is read from Kafka, `26` will be used as the initial offset.

CAUTION

Currently, the `kadb.offsets` table is not backed up via `gpbackup` since it is not considered as a configuration table. When using Kafka to ADB Connector, you need to reserve the offsets on your own.

Functions

Kafka to ADB Connector supports several functions to synchronize offsets in Kafka and appropriate offsets in the kadb.offsets table. Functions are located in the `kadb` schema. When calling the functions, remember the following:

- **None of the functions provides transaction guarantees for Kafka** — no assumptions can be made about what happens with offsets in Kafka before or after a function is called, even if that call is combined with a `SELECT` query to a foreign table in the same transaction.
- **Some functions are not atomic** — they do not create a "snapshot" of all Kafka partitions, the result is obtained from each partition independently at slightly different points in time.

Functions for offset management

Function	Parameters	Description	Atomic
kadb.commit_offsets(OID)	A foreign table OID	For a foreign table with the specified OID, commits offsets stored in `kadb.offsets` to Kafka	Yes
kadb.load_partitions(OID)	A foreign table OID	For a foreign table with the specified OID, loads partitions that exist in Kafka. The function result contains the following fields: - `ftoid` — a foreign table OID; - `prt` — a partition identifier; - `off` — the `k_initial_offset` option value.	No
kadb.partitions_obtain(OID)	A foreign table OID	For a foreign table with the specified OID, adds partitions returned by the `kadb.load_partitions(OID)` function to `kadb.offsets`. Only new partitions are added; existing ones are left unchanged	No
kadb.partitions_clean(OID)	A foreign table OID	For a foreign table with the specified OID, deletes all entries about partitions, that do not exist in Kafka, from the `kadb.offsets` table	No
kadb.partitions_reset(OID)	A foreign table OID	For a foreign table with the specified OID, deletes all entries from the `kadb.offsets` table and adds entries returned by `kadb.load_partitions(OID)` instead	No
kadb.load_offsets_at_timestamp(OID, BIGINT)	- A foreign table OID. - A timestamp — number of milliseconds since UNIX Epoch (UTC).	For a foreign table with the specified OID, loads the earliest offsets from Kafka, whose timestamps are greater or equal to the given timestamp. The result is returned only for partitions already present in the `kadb.offsets` table for the specified foreign table. The function result contains the following fields: - `ftoid` — a foreign table OID; - `prt` — a partition identifier; - `off` — the offset value found in Kafka.	Yes
kadb.offsets_to_timestamp(OID, BIGINT)	- A foreign table OID. - A timestamp — number of milliseconds since UNIX Epoch (UTC).	For a foreign table with the specified OID, changes offsets in the `kadb.offsets` table to the earliest offsets found in Kafka, whose timestamps are greater or equal to the given timestamp. Affects only partitions already present in `kadb.offsets` for the specified foreign table	Yes
kadb.load_offsets_earliest(OID)	Foreign table OID	For a foreign table with the specified OID, loads the earliest offsets from Kafka. The result is returned only for partitions already present in `kadb.offsets` for the specified foreign table. The function result contains the following fields: - `ftoid` — a foreign table OID; - `prt` — a partition identifier; - `off` — the offset value found in Kafka.	No
kadb.offsets_to_earliest(OID)	Foreign table OID	For a foreign table with the specified OID, changes offsets in the `kadb.offsets` table to the earliest offsets found in Kafka. Affects only partitions already present in `kadb.offsets` for the specified foreign table	No
kadb.load_offsets_latest(OID)	Foreign table OID	For a foreign table with the specified OID, loads the latest offsets from Kafka. The result is returned only for partitions already present in `kadb.offsets` for the specified foreign table. The function result contains the following fields: - `ftoid` — a foreign table OID; - `prt` — a partition identifier; - `off` — the offset value found in Kafka.	No
kadb.offsets_to_latest(OID)	Foreign table OID	For a foreign table with the specified OID, changes offsets in the `kadb.offsets` table to the latest offsets found in Kafka. Affects only partitions already present in `kadb.offsets` for the specified foreign table. As a result, subsequent `SELECT` queries to the selected foreign table will return messages added to the corresponding Kafka partitions after the `kadb.offsets_to_latest(OID)` function call	No
kadb.load_offsets_committed(OID)	Foreign table OID	For a foreign table with the specified OID, loads the latest committed offsets from Kafka. The result is returned only for partitions already present in `kadb.offsets` for the specified foreign table. The function result contains the following fields: - `ftoid` — a foreign table OID; - `prt` — a partition identifier; - `off` — the offset value found in Kafka.	Yes
kadb.offsets_to_committed(OID)	Foreign table OID	For a foreign table with the specified OID, changes offsets in the `kadb.offsets` table to the latest committed offsets found in Kafka. Affects only partitions already present in `kadb.offsets` for the specified foreign table	Yes

Supported formats

Kafka to ADB Connector supports deserialization of messages that were created in one of the following formats:

- AVRO OCF
- CSV
- Text

A deserialization method should be explicitly specified via the `format` option.

NOTE

Regardless of the format selected, only values of Kafka messages are deserialized. Message keys are ignored.

AVRO OCF

Kafka to ADB Connector supports the AVRO OCF serialization format with some limitations:

- AVRO schemas should contain only primitive data types. Complex types are not supported. However, there are two exceptions:
  - `union` of any primitive type with `null` (except for such unions with themselves, i.e. unions of `union` and `null`);
  - `fixed` type, which is processed in the same way as `bytes`.
- Logical types are supported.
- The foreign table definition in ADB should match the actual AVRO schema. It is also important that the order of table columns matches the order of schema fields.

The table below shows possible options for converting AVRO data types into PostgreSQL types in Kafka to ADB Connector.

Supported data type mappings

AVRO data type	PostgreSQL data type
string	TEXT, BPCHAR, VARCHAR
string	Custom PostgreSQL data type (e.g. `MONEY`). The conversion is the same as for user-provided textual input
null	Any PostgreSQL data type in a column without `NOT NULL` constraints
boolean	BOOLEAN
int	INTEGER
long	BIGINT
float	REAL
double	DOUBLE PRECISION
bytes, fixed	BYTEA
decimal	NUMERIC
date	DATE
time-millis, time-micros	TIME
timestamp-millis	TIMESTAMP(<N>), where `<N>` is equal to `1`, `2`, or `3`
timestamp-micros	TIMESTAMP; TIMESTAMP(<N>), where `<N> >= 4`
duration	INTERVAL

The following are examples of AVRO schemas that are valid for Kafka to ADB Connector.

Examples of valid AVRO schemas

```
{
  "name":"doc",
  "type":"record",
  "fields":[
    {
      "name":"id",
      "type":"int"
    },
    {
      "name":"text",
      "type":[
        "string",
        "null"
      ]
    },
    {
      "name":"issued_on",
      "type":"int",
      "logicalType":"date"
    }
  ]
}
```

```
{
  "name":"doc",
  "type":"record",
  "fields":[
    {
      "name":"d",
      "type":"int",
      "logicalType":"date"
    },
    {
      "name":"t_ms",
      "type":"int",
      "logicalType":"time-millis"
    },
    {
      "name":"t_us",
      "type":"long",
      "logicalType":"time-micros"
    },
    {
      "name":"ts_ms",
      "type":"long",
      "logicalType":"timestamp-millis"
    },
    {
      "name":"ts_us",
      "type":"long",
      "logicalType":"timestamp-micros"
    },
    {
      "name":"dur",
      "type":{
        "name":"dur_fixed",
        "type":"fixed",
        "size":12,
        "logicalType":"duration"
      }
    },
    {
      "name":"dec_1",
      "type":{
        "name":"dec_2_fixed",
        "type":"fixed",
        "size":6,
        "logicalType":"decimal"
      }
    },
    {
      "name":"dec_2",
      "type":{
        "name":"dec_2_fixed",
        "type":"bytes",
        "logicalType":"decimal",
        "precision":14,
        "scale":4
      }
    }
  ]
}
```

CSV

CSV format support is provided in Kafka to ADB Connector by lbcsv. As a result, all conventions about CSV used by that library are applied. The CSV specification is defined in RFC 4180.

Taking into account the abovementioned recommendations, Kafka to ADB Connector uses the following CSV parsing rules:

- CSV fields (attributes) are separated by a character that is specified via the `csv_delimeter` option.
- CSV rows (records) are separated by a newline character sequence.
- CSV fields can be enclosed into quotes specified via the `csv_quote` option.
- CSV fields that contain a delimiter, a quote, or a newline character should be quoted.
- Each quotation mark that is used inside of CSV fields should be escaped with a preceding quote character.
- Empty fields are parsed as `NULL` values.
- Empty rows are skipped.
- Leading and trailing whitespaces are removed from non-quoted fields (if the `csv_attribute_trim_whitespace` option is set).

CSV values can be converted to any PostgreSQL datatype. The conversion is the same as for `psql` textual input.

Text

The text format is used to deserialize data presented in Kafka messages as raw text. When you select that format, messages are processed by connector as follows:

- Each message read from Kafka is treated as one column (attribute) of a single tuple (row) in a foreign table.
- The data is parsed by PostgreSQL as user-provided textual data.
- Empty Kafka messages (with `0` length) are parsed as `NULL` values.

CAUTION

In order to use the `text` format for data deserialization, define exactly one column in the foreign table (of any PostgreSQL data type).

Implementation details

Below are the features of Kafka to ADB Connector implementation that have a significant impact on its usage.

Concurrent SELECT and global deadlock detector

As mentioned above, Kafka to ADB Connector refers to `kadb.offsets` at each `SELECT` query against foreign tables. The `kadb.offsets` table has a `DISTRIBUTED REPLICATED` type in ADB. Both `INSERT` and `UPDATE` queries are possible to that table. Thus, Greenplum limitations on concurrent transactions can impact `SELECT` queries to foreign tables.

The way concurrent (parallel) transactions are processed in Greenplum is defined by the current state of global deadlock detector.

When the global deadlock detector is disabled, each `UPDATE` query requires `ExclusiveLock`, which actually locks the entire table being updated. In Kafka to ADB Connector, this means that concurrent `SELECT` queries to different foreign tables are not possible. Such `SELECT` queries are executed sequentially, one at a time.

To allow multiple concurrent `SELECT` queries against different foreign tables, enable the global deadlock detector. With the detector enabled, each `UPDATE` query requires `RowExclusiveLock`, thus permitting multiple concurrent `UPDATE` queries to the `kadb.offsets` table, and hence multiple `SELECT` queries against foreign tables.

To enable the global deadlock detector, set the `gp_enable_global_deadlock_detector` GUC value to `on`:

```
$ gpconfig -c gp_enable_global_deadlock_detector -v on
```

Concurrent `SELECT` queries against a single foreign table are not possible. In some cases, such queries may succeed and even return correct results (the same for all concurrent queries). However, this is not guaranteed.

Distribution of partitions across segments

Each `SELECT` query against a foreign table considers only partitions that are present in the kadb.offsets table. The `kadb.offsets` contents can be modified before `SELECT` execution — automatically if the `k_automatic_offsets` option is set or manually by means of functions.

Partitions are distributed among ADB segments according to the following rules:

- Distribution is performed evenly across all segments. The number of partitions assigned to a segment does not differ by more than one with respect to other segments.
- The order of partitions (as returned by Kafka) and the order of ADB segments to which they are assigned match.

For example, in the ADB cluster that consists of `3` segments, partitions `[0, 2, 3, 4, 1]` returned from Kafka will be distributed as follows.

Segment	Partitions
1	[0, 2]
2	[3, 4]
3	[1]

Query termination

Each query that involves `librdkafka` functions or `SELECT` calls against a foreign table can be terminated by users before it completes. A query is guaranteed to terminate in less than one second after all ADB segments receive a query termination signal.

However, active connections to Kafka are not closed immediately after query termination. Under normal conditions, connections will be closed within the time specified in the `k_latency_ms` option. But when Kafka does not respond (e.g. if Kerberos authentication fails), the connection may remain active for indefinite period of time.

To ensure all resources are released, close the ADB session within which the query was sent.

Duration of SELECT queries

If a `SELECT` query against a foreign table is not interrupted, its maximum duration can be estimated as follows:

$$\mathrm{duration} = \mathrm{k\_latency\_ms} \cdot \left(1 + \left\lceil \frac{\text{number of partitions}}{\text{number of segments}} \right\rceil\right) + \mathrm{k\_timeout\_ms}$$

The `k_latency_ms` and `k_timeout_ms` options affect the `SELECT` duration significantly. Thus, these options should be set properly.

Kafka to ADB Connector usage examples

Daria Barysheva

Contents

To Table of Contents

PLAINTEXT security protocol (by default)

SASL_PLAINTEXT security protocol

- PLAIN mechanism
- GSSAPI mechanism (Kerberos)

SASL_SSL security protocol with GSSAPI mechanism (Kerberos)

This article describes how Kafka to ADB Connector can be used to read AVRO OCF messages written to Kafka topics (as a part of ADS) by ADB to Kafka Connector. The use of both connectors simultaneously is optional. The purpose of examples is only to show the ability to read AVRO OCF messages (this is the format in which messages are serialized by ADB to Kafka Connector). The following prerequisites are met:

- The ADS cluster is installed according to the Online installation guide on the following hosts: `bds-ads1.ru-central1.internal`, `bds-ads2.ru-central1.internal`, `bds-ads3.ru-central1.internal`.
- The ADB cluster is installed according to the Online installation guide.
- In ADB, the ***Kafka to ADB*** service is added and installed.
- The `adb` database exists in ADB.
- All ADB nodes have access to the `9092` port on all ADS nodes through which you plan to load data.
- In ADB, the ADB to Kafka Connector is installed.

PLAINTEXT security protocol (by default)

1. Perform steps 1—5 from the ***Without SERVER option*** section of the ADB to Kafka Connector usage examples article (if you have not passed them yet). As a result, the `topic_adb_to_kafka` topic should contain 5 messages.
2. Connect to the `adb` database of the ADB cluster under the `gpadmin` user (for example, via `psql`). Create the following objects in the given order.
   - The `kafka_server` server:

```
DROP SERVER IF EXISTS kafka_server CASCADE;
CREATE SERVER kafka_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092'
);
```

   - The `kafka_table` foreign table (note that the table structure should match the AVRO schema):

```
DROP FOREIGN TABLE IF EXISTS kafka_table;
CREATE FOREIGN TABLE kafka_table(a INT, b TEXT)
SERVER kafka_server
OPTIONS (
    format 'avro',
    k_topic 'topic_adb_to_kafka',
    k_consumer_group 'group',
    k_seg_batch '100',
    k_timeout_ms '1000'
);
```

3. In ADB, insert some data into the ext_adb_to_kafka table to send it to the `topic_adb_to_kafka` topic in ADS:

```
INSERT INTO ext_adb_to_kafka VALUES
(6,'test6'),
(7,'test7'),
(8,'test8'),
(9,'test9'),
(10,'test10');
```

4. Select data from the `kafka_table` foreign table:

```
SELECT * FROM kafka_table;
```

   The result contains some notifications that are described in Kafka to ADB Connector configuration (see the `k_initial_offset` and `k_automatic_offsets` options in the ***Server options*** table) and a list of messages read from Kafka:

```
NOTICE:  Kafka-ADB: Offset for partition 0 is not known, and is set to default value 0  (seg0 slice1 10.92.38.140:10000
pid=2323744)
 a  |   b
----+--------
  1 | test1
  2 | test2
  4 | test4
  3 | test3
  5 | test5
  7 | test7
  6 | test6
  8 | test8
  9 | test9
 10 | test10
(10 rows)
```

5. Check that the kadb.offsets table contains a new row:

```
SELECT * FROM kadb.offsets;
```

   Result:

```
 ftoid | prt | off
-------+-----+-----
 45426 |   0 |  10
(1 row)
```

   where `45426` is OID of the `kafka_table` foreign table, which can be defined as follows:

```
SELECT 'public.kafka_table'::regclass::oid;
```

```
  oid
-------
 45426
(1 row)
```

SASL_PLAINTEXT security protocol

PLAIN mechanism

1. Perform all steps from the ***PLAIN mechanism*** section of the ADB to Kafka Connector usage examples article (if you have not passed them yet). As a result, the `topic_adb_to_kafka_sasl` topic should contain 1 message.
2. Connect to the `adb` database of the ADB cluster under the `gpadmin` user (for example, via `psql`). Create the following objects in the given order.
   - The `kafka_server` server:

```
DROP SERVER IF EXISTS kafka_server CASCADE;
CREATE SERVER kafka_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092'
);
```

   - The `kafka_table_sasl` foreign table (note that compared to the previous example, the table definition contains librdkafka options specific for the `SASL_PLAINTEXT` protocol with the `PLAIN` mechanism):

```
DROP FOREIGN TABLE IF EXISTS kafka_table_sasl;
CREATE FOREIGN TABLE kafka_table_sasl(a INT, b TEXT)
SERVER kafka_server
OPTIONS (
    format 'avro',
    k_topic 'topic_adb_to_kafka_sasl',
    k_consumer_group 'group',
    k_seg_batch '100',
    k_timeout_ms '1000',
    "#security.protocol" 'sasl_plaintext',
    "#sasl.mechanism" 'PLAIN',
    "#sasl.username" 'adb-to-kafka',
    "#sasl.password" '123'
);
```

3. Select data from the `kafka_table_sasl` foreign table:

```
SELECT * FROM kafka_table_sasl;
```

   The result contains some notifications that are described in Kafka to ADB Connector configuration (see the `k_initial_offset` and `k_automatic_offsets` options in the ***Server options*** table) and a list of messages read from Kafka:

```
NOTICE:  Kafka-ADB: Offset for partition 0 is not known, and is set to default value 0  (seg0 slice1 10.92.38.140:10000 pid=8036)
 a |   b
---+-------
 1 | test1
(1 row)
```

Authentication errors

If user credentials are not specified correctly in the foreign table definition, the following error occurs when you try to select data from that table:

```
ERROR:  Kafka-ADB: [kafka] Failed to retrieve metadata for topic 'topic_adb_to_kafka_sasl': Local: Broker transport
failure [-195] [-1]
```

GSSAPI mechanism (Kerberos)

1. Perform all steps from the ***GSSAPI mechanism (Kerberos)*** section of the ADB to Kafka Connector usage examples article (if you have not passed them yet). As a result, the `topic_adb_to_kafka_kerberos` topic should contain 1 message.
2. Connect to the `adb` database of the ADB cluster under the `gpadmin` user (for example, via `psql`). Create the following objects in the given order.
   - The `kafka_server` server:

```
DROP SERVER IF EXISTS kafka_server CASCADE;
CREATE SERVER kafka_server
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092'
);
```

   - The `kafka_table_kerberos` foreign table (note that compared to the previous examples, the table definition contains librdkafka options specific for the `SASL_PLAINTEXT` protocol with the `GSSAPI` mechanism):

```
DROP FOREIGN TABLE IF EXISTS kafka_table_kerberos;
CREATE FOREIGN TABLE kafka_table_kerberos(a INT, b TEXT)
SERVER kafka_server
OPTIONS (
    format 'avro',
    k_topic 'topic_adb_to_kafka_kerberos',
    k_consumer_group 'group',
    k_seg_batch '100',
    k_timeout_ms '1000',
    "#security.protocol" 'sasl_plaintext',
    "#sasl.mechanism" 'GSSAPI',
    "#sasl.kerberos.keytab" '/var/lib/pxf/conf/kerberos/adb-to-kafka.service.keytab',
    "#sasl.kerberos.principal" 'adb-to-kafka',
    "#sasl.kerberos.service.name" 'kafka'
);
```

3. Select data from the `kafka_table_kerberos` foreign table:

```
SELECT * FROM kafka_table_kerberos;
```

   The result contains some notifications that are described in Kafka to ADB Connector configuration (see the `k_initial_offset` and `k_automatic_offsets` options in the ***Server options*** table) and a list of messages read from Kafka:

```
NOTICE:  Kafka-ADB: Offset for partition 0 is not known, and is set to default value 0  (seg0 slice1 10.92.38.140:10000
pid=2404402)
 a |    b
---+----------
 1 | kerberos
(1 row)
```

Authentication errors

If connection parameters (keytab, principal, or service name) are not specified correctly in the external table definition and there is no a valid ticket for the selected service in the ticket cache, the following error occurs when you try to select data from that table:

```
ERROR:  Kafka-ADB: [kafka] Failed to retrieve metadata for topic 'topic_adb_to_kafka_kerberos': Local: Broker transport
failure [-195] [-1]
```

SASL_SSL security protocol with GSSAPI mechanism (Kerberos)

1. Perform all steps from the ***SASL_SSL security protocol with GSSAPI mechanism (Kerberos)*** section of the ADB to Kafka Connector usage examples article (if you have not passed them yet). As a result, the `topic_ssl` topic should contain 1 message.
2. Connect to the `adb` database of the ADB cluster under the `gpadmin` user (for example, via `psql`). Create the following objects in the given order.
   - The `kafka_server_ssl` server (note that compared to the previous examples, the server definition should contain FQDN of ADS hosts):

```
DROP SERVER IF EXISTS kafka_server_ssl CASCADE;
CREATE SERVER kafka_server_ssl
FOREIGN DATA WRAPPER kadb_fdw
OPTIONS (
    k_brokers 'bds-ads1.ru-central1.internal:9092,bds-ads2.ru-central1.internal:9092,bds-ads3.ru-central1.internal:9092'
);
```

   - The `kafka_table_ssl` foreign table (note that compared to the previous examples, the table definition contains librdkafka options specific for the `SASL_SSL` protocol with the `GSSAPI` mechanism):

```
DROP FOREIGN TABLE IF EXISTS kafka_table_ssl;
CREATE FOREIGN TABLE kafka_table_ssl(a INT, b TEXT)
SERVER kafka_server_ssl
OPTIONS (
    format 'avro',
    k_topic 'topic_ssl',
    k_consumer_group 'group',
    k_seg_batch '100',
    k_timeout_ms '1000',
    "#security.protocol" 'sasl_ssl',
    "#sasl.mechanism" 'GSSAPI',
    "#sasl.kerberos.keytab" '/var/lib/pxf/conf/kerberos/adb-to-kafka.service.keytab',
    "#sasl.kerberos.principal" 'adb-to-kafka',
    "#sasl.kerberos.service.name" 'kafka',
    "#ssl.ca.location" '/etc/ssl/certs/ca-bundle.crt',
    "#ssl.certificate.location" '/etc/ssl/certs/host_cert.cert',
    "#ssl.key.password" 'bigdata'
);
```

3. Select data from the `kafka_table_ssl` foreign table:

```
SELECT * FROM kafka_table_ssl;
```

   The result contains some notifications that are described in Kafka to ADB Connector configuration (see the `k_initial_offset` and `k_automatic_offsets` options in the ***Server options*** table) and a list of messages read from Kafka:

```
NOTICE:  Kafka-ADB: Offset for partition 0 is not known, and is set to default value 0  (seg0 slice1 10.92.38.140:10000
pid=2835718)
 a |       b
---+---------------
 1 | ssl-test-data
(1 row)
```

Authentication errors

If certificate paths are not specified correctly in the foreign table definition, the following error occurs when you try to select data from that table:

```
ERROR:  Kafka-ADB: [kafka] Failed to create a Kafka consumer: ssl.key.location failed: error:0A080002:SSL routines::system
lib [5]
```

When specifying an incorrect keytab, principal, or service name, you get an error message identical to the one described above for the SASL_PLAINTEXT protocol with GSSAPI mechanism.

ADB to ADB Connector overview

Daria Barysheva

Contents

To Table of Contents

Features
Query execution schemas
- SELECT
- INSERT
Supported push-down predicates
Supported work modes
- Define the number of executors automatically
- Use the number of executors equal to the number of local segments
- Define the number of executors explicitly

Features

ADB to ADB Connector provides two-way interaction between ADB clusters. The connector is implemented on the basis of a foreign data wrapper and parallel retrieve cursors.

ADB to ADB Connector has the following features:

- Segments of a local cluster have the ability to select data directly from segments of a remote cluster in a parallel mode. For some queries (e.g. using aggregates), the "coordinator-coordinator" mode is supported.
- Transactional data insertion from local to remote clusters in the "coordinator-coordinator" mode is available. ADB to ADB Connector provides transaction support similar to postgres_fdw in the following way:
  - Start a transaction when opening a connection.
  - Implement inner transaction processing via SAVEPOINT.
  - Process transaction rollbacks.
  - Close parameterized queries.
- ADB to ADB Connector supports the ability to calculate the number of query executors (QE) automatically — see Supported work modes below. If users do not want to specify the number of segments in a remote cluster, they can set the `use_remote_estimate` option to `true` at the server or foreign table level — to automatically estimate this number based on the query (see ADB to ADB Connector configuration for more details). Also, changing the number of segments in a remote cluster does not affect ADB to ADB Connector.

IMPORTANT

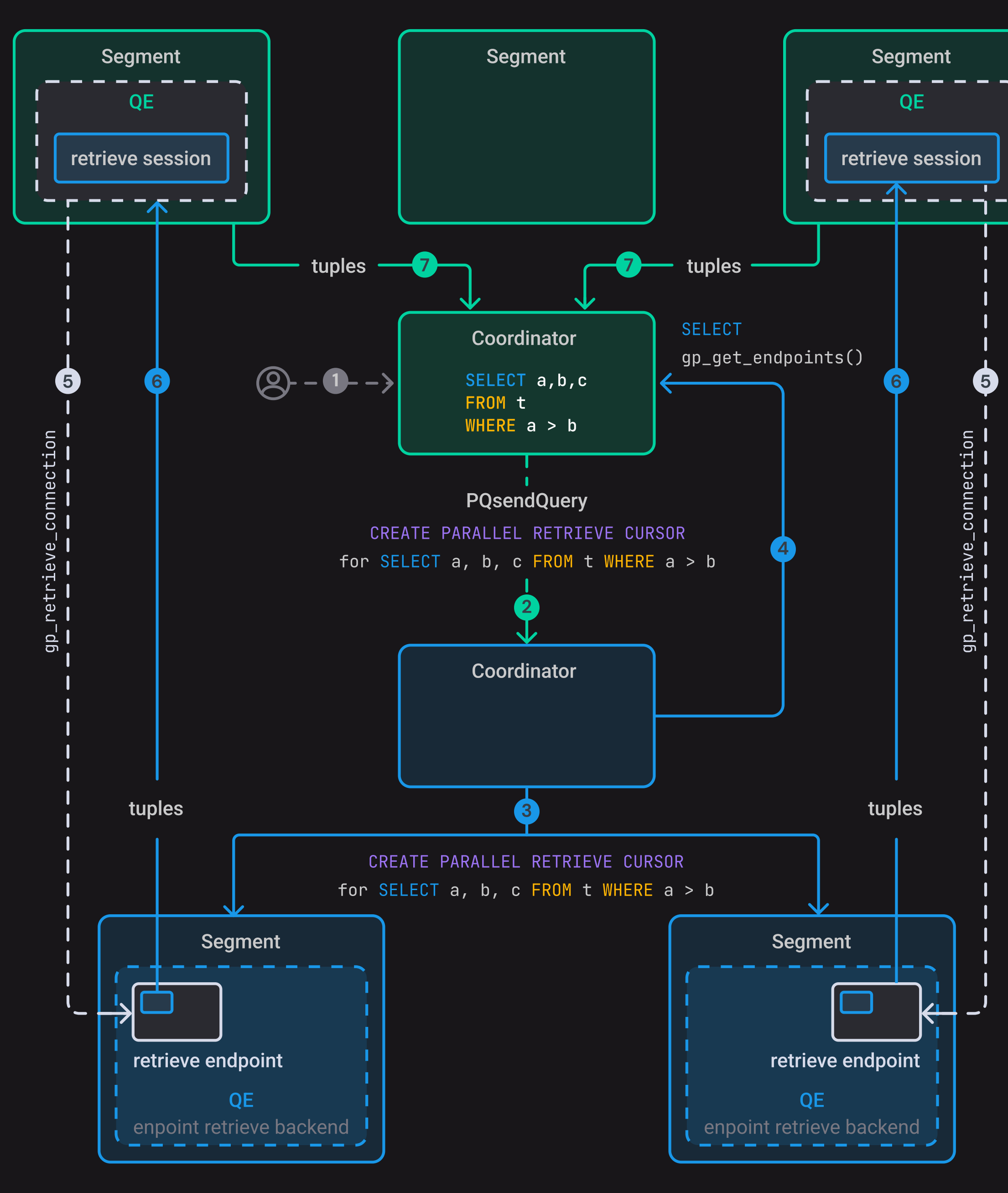
- ADB to ADB Connector is available in the Enterprise version of ADB (starting with 6.25.1.51 in ADB 6 and 7.4.0 in ADB 7).
- ADB to ADB Connector does not support the `UPDATE` and `DELETE` queries. Trying to call them will cause an error: `ERROR: adb_fdw. UPDATE and DELETE queries are forbidden`.
- Sending `INSERT` queries to a target (remote) cluster is not the main operating mode of ADB to ADB Connector and is not suitable for bulk data loading. A more effective strategy is to use the connector in the `SELECT` mode on a remote cluster, and then to call `INSERT` on its side. In this case, the remote cluster segments receive data in parallel directly from the local cluster segments, and then during the `INSERT` operation the remote cluster distributes data to its own segments.

Query execution schemas

SELECT

`SELECT` queries are processed by ADB to ADB Connector in the following sequence:

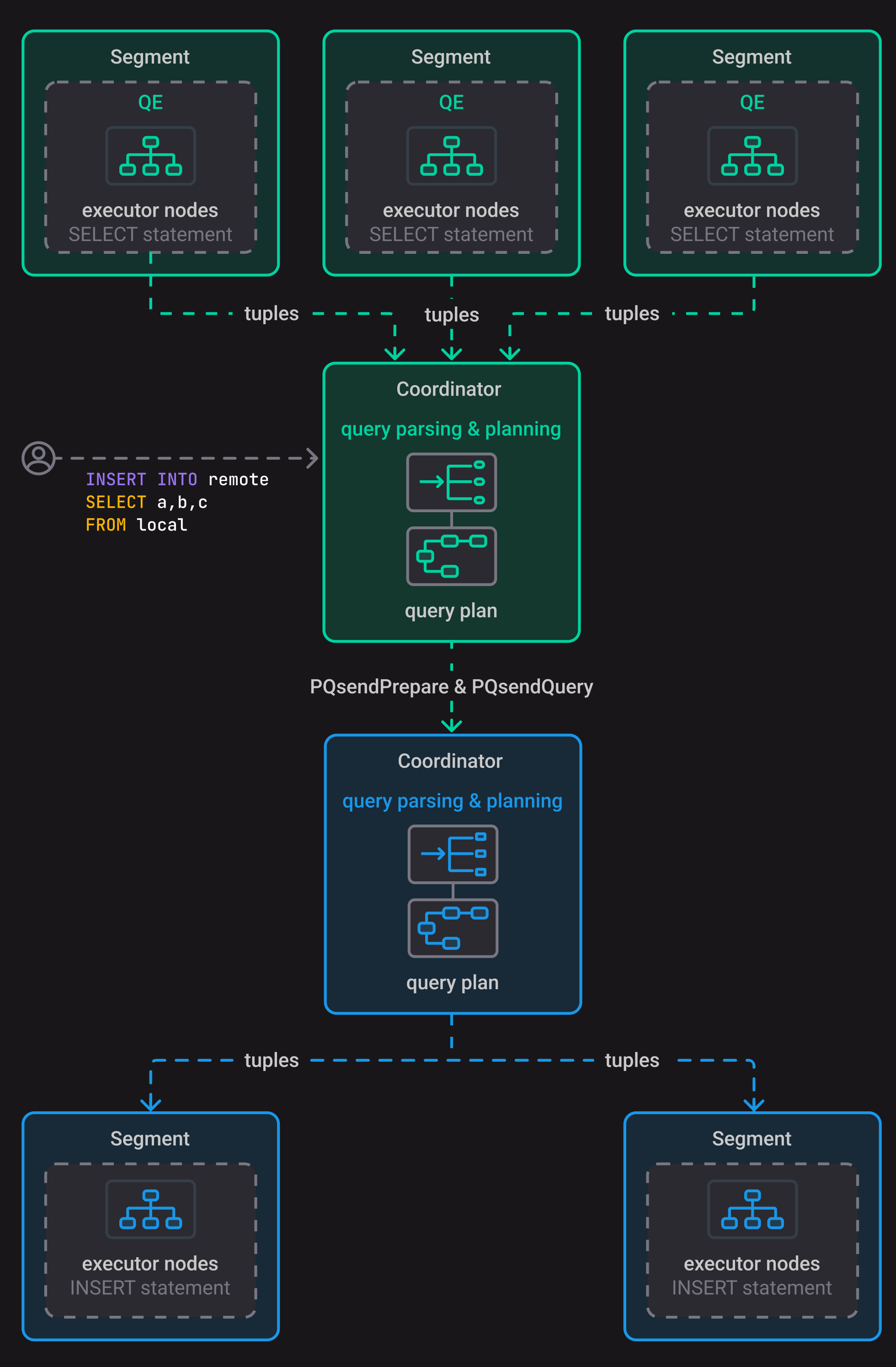
1. A user sends a `SELECT` query to the coordinator host of the local cluster.
2. The local coordinator requests creation of a parallel cursor on the coordinator of the remote cluster.
3. The remote coordinator creates endpoints.
4. As a result of calling the `gp_get_endpoints` function, the remote coordinator returns some service information about the endpoints created: their number, IP addresses and port numbers, as well as additional data required for authentication.
5. On the basis of the received information about the number of endpoints, an equal number of query executors (QE) is created on the segments of the local cluster. The retrieve sessions are started on local segments.
6. Local cluster segments start reading data directly from remote cluster segments.
7. Local cluster segments return data to the local coordinator in order to pass the query result to the user.


Segment
QE
retrieve session
Segment
Segment
QE
retrieve session
tuples
7
7
tuples
Coordinator
SELECT a,b,c
FROM t
WHERE a > b
SELECT
gp_get_endpoints()
1
5
6
6
5
gp_retrieve_connection
gp_retrieve_connection
PQsendQuery
CREATE PARALLEL RETRIEVE CURSOR
for SELECT a, b, c FROM t WHERE a > b
2
4
Coordinator
3
tuples
tuples
CREATE PARALLEL RETRIEVE CURSOR
for SELECT a, b, c FROM t WHERE a > b
Segment
retrieve endpoint
QE
enpoint retrieve backend
Segment
retrieve endpoint
QE
enpoint retrieve backend

SELECT execution schema

INSERT

`INSERT` queries are processed by ADB to ADB Connector in the "coordinator-coordinator" mode, which means the local and remote cluster segments do not interact directly with each other. Currently, this mode is the only guarantee of data consistency.


Segment
QE
executor nodes
SELECT statement
Segment
QE
executor nodes
SELECT statement
Segment
QE
executor nodes
SELECT statement
tuples
tuples
tuples
Coordinator
query parsing & planning
query plan
INSERT INTO remote
SELECT a,b,c
FROM local
PQsendPrepare & PQsendQuery
Coordinator
query parsing & planning
query plan
tuples
tuples
Segment
executor nodes
INSERT statement
Segment
executor nodes
INSERT statement

INSERT execution schema

Supported push-down predicates

Due to a number of limitations present in Greenplum 6, the current version of ADB to ADB Connector does not fully implement push-down predicate support for `SELECT` queries (with remote-side processing). The list of available predicates and their execution features are shown in the table below.

Predicate	Push-down support	Note
WHERE	Yes, with exceptions	Only predicates operating on built-in data types, operators, and functions are transferred to the remote cluster. Also, the operators and functions that are used in the `WHERE` clause should satisfy the `IMMUTABLE` condition. Internal system columns, except for `ctid`, are not transferred to the remote cluster
LIMIT	No	The condition is applied on the local cluster side
ORDER BY	No	The condition is applied on the local cluster side
Any aggregate function	No	The remote cluster returns all rows that meet the conditions of the `WHERE` clause (if they are present). Then the rows are processed within the aggregate function on the local cluster side
Any type of JOIN	No	The remote cluster returns all rows that meet the conditions of the `WHERE` clause (if they are present). Then the rows are processed within JOIN operator on the local cluster side

Supported work modes

NOTE

- For more information on belowmentioned options, see ADB to ADB Connector configuration.
- Examples of the ADB to ADB Connector usage can be found in ADB to ADB Connector usage examples.

Define the number of executors automatically

In this mode, the `use_remote_estimate` option is set to `true` at the server or foreign table level. As a result, the number of required query executors is calculated automatically — equal to the number of endpoints of the parallel cursor.

The server definition example is given below:

```
CREATE SERVER test_server
FOREIGN DATA WRAPPER adb_fdw
OPTIONS (
    host 'bds-mdw-ext.ru-central1.internal',
    port '5432',
    dbname 'adb',
    use_remote_estimate 'true');
```

Pros	Cons
The number of executors is calculated automatically. Users do not need to know this number in advance. Any changes in the number of remote segments are processed correctly — without the need to update the foreign table configuration.	A parallel cursor should be created and opened in a remote cluster during the query planning phase

Use the number of executors equal to the number of local segments

In this mode, the `use_remote_estimate` and `num_segments` options are not specified. As a result, the number of executors is static and equal to the number of segments in the local cluster. If the endpoints number for the query does not match the number of local segments, then the mapping to local executors is performed.

The server definition example is given below:

```
CREATE SERVER test_server
FOREIGN DATA WRAPPER adb_fdw
OPTIONS (
    host 'bds-mdw-ext.ru-central1.internal',
    port '5432',
    dbname 'adb');
```

Pros	Cons
There is no need to create and open a parallel cursor in a remote cluster during the query planning stage	If a remote cluster has a larger number of segments compared to a local cluster, some local cluster executors will get more endpoints which will cause a processing skew. The endpoints assigned to the same Query Executor will be processed sequentially, which, in turn, will reduce the processing parallelization level.

Define the number of executors explicitly

In this mode, the `use_remote_estimate` option is not specified (by default, it is `false`), but the `num_segments` option is defined at the server level, which explicitly specifies the number of executors. Unlike the previous scheme, you can define a number of executors other than the number of local segments — for example, equal to the number of segments in the remote cluster. If the endpoints number for the query does not match the number of local segments, then the mapping to local executors is performed.

The server definition example is given below:

```
CREATE SERVER test_server
FOREIGN DATA WRAPPER adb_fdw
OPTIONS (
    host 'bds-mdw-ext.ru-central1.internal',
    port '5432',
    dbname 'adb',
    num_segments '16');
```

Pros	Cons
There is no need to create and open a parallel cursor in a remote cluster during the query planning stage. The advantage over the previous scheme is the ability to eliminate a processing skew by explicitly matching the number of executors to the number of remote segments.	The number of query executors should be calculated manually. The automatic work mode does not have this drawback. The possibility of a processing skew during query execution is not completely excluded (in case of incorrect `num_segments` definition).

ADB to ADB Connector installation

Daria Barysheva

To Table of Contents

To install ADB to ADB Connector, follow the steps:

1. Make sure that the ADB to ADB service is installed. For information on how to add services to ADB clusters, see Add services.

IMPORTANT

The ADB to ADB service should be installed in both ADB clusters between which you plan to transfer data.

2. Connect to the database via `psql` (or any other client program):

```
$ psql adb
```

3. Check that the `adb_fdw` extension is available — via the `psql` meta-command `\dx`.

NOTE

If you want to connect to an ADB 6 cluster, you also need to make sure the `gp_parallel_retrieve_cursor` extension is installed on that cluster. Starting with ADB 7, this extension functionality has been integrated into the core system so it's no longer required for interaction between ADB 7 clusters.

```
\dx
```

The output contains the `adb_fdw` extension:

```
                                          List of installed extensions
       Name        | Version |   Schema   |                             Description
-------------------+---------+------------+---------------------------------------------------------------------
 adb_fdw           | 1.0     | public     | foreign-data wrapper for remote ADB servers
 arenadata_toolkit | 1.0     | public     | extension is used for manipulation of objects created by adb-bundle
 dblink            | 1.2     | public     | connect to other PostgreSQL databases from within a database
 gp_exttable_fdw   | 1.0     | pg_catalog | External Table Foreign Data Wrapper for Greengage
 gp_toolkit        | 1.8     | gp_toolkit | various GPDB administrative views/functions
 kadb_fdw          | 0.16    | kadb       | Kafka-ADB foreign data wrapper
 plpgsql           | 1.0     | pg_catalog | PL/pgSQL procedural language
 pxf               | 2.1     | public     | Extension which allows to access unmanaged data
 pxf_fdw           | 1.0     | public     | PXF Foreign Data Wrapper for Greengage
 tkh_fdw           | 1.0     | public     | ClickHouse Foreign Data Wrapper for Greenplum
(10 rows)
```

4. You can also check that the new foreign data wrapper `adb_fdw` is available using the following SQL query against the `pg_catalog.pg_foreign_data_wrapper` system table:

```
SELECT * FROM pg_catalog.pg_foreign_data_wrapper;
```

Result:

```
  oid  |     fdwname     | fdwowner | fdwhandler | fdwvalidator | fdwacl |                                          fdwoptions
-------+-----------------+----------+------------+--------------+--------+-------------------------------------------------------
-------------------------------------
 13702 | gp_exttable_fdw |       10 |      13700 |        13701 |        |
 17124 | dblink_fdw      |       10 |          0 |        17123 |        |
 17175 | adb_fdw         |       10 |      17173 |        17174 |        |
 17187 | jdbc_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=jdbc,"mpp_execute=all segments"}
 17188 | hdfs_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=hdfs,"mpp_execute=all segments"}
 17189 | hive_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=hive,"mpp_execute=all segments"}
 17190 | hbase_pxf_fdw   |       10 |      17185 |        17186 |        | {protocol=hbase,"mpp_execute=all segments"}
 17191 | s3_pxf_fdw      |       10 |      17185 |        17186 |        | {protocol=s3,"mpp_execute=all segments"}
 17192 | gs_pxf_fdw      |       10 |      17185 |        17186 |        | {protocol=gs,"mpp_execute=all segments"}
 17193 | adl_pxf_fdw     |       10 |      17185 |        17186 |        | {protocol=adl,"mpp_execute=all segments"}
 17194 | wasbs_pxf_fdw   |       10 |      17185 |        17186 |        | {protocol=wasbs,"mpp_execute=all segments"}
 17195 | file_pxf_fdw    |       10 |      17185 |        17186 |        | {protocol=file,"mpp_execute=all segments"}
 17201 | tkh_fdw         |       10 |      17200 |        17199 |        | {protocol=tkh_clickhouse,"mpp_execute=all
segments",format=text}
 17210 | kadb_fdw        |       10 |      17209 |        17211 |        | {"mpp_execute=all
segments",k_initial_offset=0,k_automatic_offsets=true,k_latency_ms=2000}
(14 rows)
```

ADB to ADB Connector configuration

Daria Barysheva

Contents

To Table of Contents

Server
User mapping
Foreign table

To send data from the local (source) ADB cluster to the remote (target) ADB cluster via ADB to ADB Connector, you should first create the following objects on the local cluster side:

1. Server — encapsulates connection information that a foreign data wrapper uses to access an external data source.
2. User mapping — provides user authentication to an external data source.
3. Foreign table — a table in ADB that defines the remote data structure. A foreign table has no storage in ADB, but can be used in queries just like a normal table.

To see how to use the listed objects to send data, refer to ADB to ADB Connector usage examples.

IMPORTANT

- For each target ADB cluster to which you plan to send queries from the source ADB cluster, it is sufficient to create one server and one user mapping. However, there may be several foreign tables if you need to send queries against various ADB tables.
- All options that are listed in the tables below (Server options, User mapping options, Foreign table options) should be defined in the text format (i.e. in single quotes `'`). The ***Type*** column in the tables shows what data type will be used when validating input values.

Server

To create a server, use the `CREATE SERVER` command. The basic command syntax is listed below:

```
CREATE SERVER <server_name> [ TYPE '<server_type>' ] [ VERSION '<server_version>' ]
    FOREIGN DATA WRAPPER <fdw_name>
    [ OPTIONS ( [ <option> '<value>' [, ... ]] ) ]
```

where:

- `<server_name>` — a server name in ADB. Should be unique within the current ADB database.
- `<server_type>` — an optional server type.
- `<server_version>` — an optional server version.
- `<fdw_name>` — a foreign data wrapper name. You should use the `adb_fdw` foreign data wrapper, which is automatically created after the connector installation (see step 4 in ADB to ADB Connector installation).
- `<option>` — server options that define the connection details. The options that are available for ADB to ADB Connector are listed in the Server options table. Note that these options (except `host`, `port`, `num_segments`) can be defined both at the server level and at the foreign table level. Options marked as required should be specified at least in one definition (server or table). Table-level options take precedence.
- `<value>` — option values.

NOTE

- To create a server, you need the `USAGE` privilege on the `adb_fdw` foreign data wrapper. The user who creates the server becomes its owner.
- The full description of the `CREATE SERVER` command syntax is available in the Greenplum documentation.
- To edit a server definition, use the ALTER SERVER command. To delete a server, use DROP SERVER.

Server options

Name	Type	Description	Default	Required
host	TEXT	An IP address or name of the remote ADB coordinator host. Validation rules for the `host` option are listed below	—	Yes
port	TEXT	A port number that is used to connect to the remote ADB coordinator host. Validation rules for the `port` option are listed below	—	Yes
dbname	TEXT	A name of the target database on the remote ADB cluster side	—	Yes
mpp_execute	TEXT	The distribution strategy for query execution. The only possible value is `coordinator`. The `mpp_execute` option cannot be changed. The execution strategy (`coordinator`, `all segments`, or `any`) is selected automatically by the ADB to ADB Connector, depending on the type of operation requested and the result of the query planning	coordinator	No
use_remote_estimate	BOOL	Indicates whether to use an external cluster for planning a query. Based on the flag, query costs are estimated and the number of executors on segments is calculated. If `use_remote_estimate = true`, a parallel cursor is created and the number of executors is determined according to the number of parallel cursor endpoints. Otherwise, the number of executors is taken from `num_segments`, and if this parameter is not filled — the current number of segments is used. For more information, see ADB to ADB Connector overview. Possible values: `true`, `false`	false	No
num_segments	INT	The number of executors on segments that are used to process data received from the parallel cursor endpoints. If the number of local cluster segments is smaller than the number of endpoints, some local cluster executors will get more endpoints (which will cause processing skew). Positive integers >= `1` are allowed	—	No
fdw_startup_cost	DOUBLE	The cost estimate for obtaining the first tuple from the data source. Positive double values are allowed	100.0	No
fdw_tuple_cost	DOUBLE	The cost estimate for obtaining one tuple from the data source. Positive double values are allowed	0.01	No
fetch_size	INT	The number of rows in the data batch read by the connector. Positive integers >= `1` are allowed	100	No
updatable	BOOL	Indicates the ability to change tables on the remote ADB cluster side via the `INSERT` command. Possible values: `true`, `false`	true	No

Validation rules for the host and port options

- The maximum length of the `host` value is 255 characters.
- You can use in the `host` value digits `0-9`, ASCII characters `a-zA-Z`, hyphens `-`, underscores `_`, and dots `.`.
- The maximum length of the `port` value is 5 characters.
- Only digits `0-9` are allowed in the `port` value.

User mapping

To create a user mapping, use the `CREATE USER MAPPING` command. The basic command syntax is listed below:

```
CREATE USER MAPPING FOR { <username> | USER | CURRENT_USER | PUBLIC }
    SERVER <server_name>
    [ OPTIONS ( <option> '<value>' [, ... ] ) ]
```

where:

- `<username>` — a user name in ADB. The following constants are allowed:
  - `USER` or `CURRENT_USER` — the current user who established the ADB connection.
  - `PUBLIC` — all present and future ADB users.
- `<server_name>` — a server name.
- `<option>` — user mapping options, which typically include a user name and a password to connect to an external data source. The options that are available for ADB to ADB Connector are listed in the User mapping options table.
- `<value>` — option values.

NOTE

- To create a user mapping, you should be the owner of the corresponding server.
- The full description of the `CREATE USER MAPPING` command syntax is available in the Greenplum documentation.
- To edit a user mapping definition, use the ALTER USER MAPPING command. To delete a user mapping, use DROP USER MAPPING.

User mapping options

Name	Type	Description	Default	Required
user	TEXT	A user name in the remote ADB database	—	Yes
password	TEXT	A user password in the remote ADB database	—	Yes

Provide access via pg_hba.conf

In addition to a user mapping creation, you should provide an access of the selected user to the remote ADB database from the local cluster. Add the following record to the *pg_hba.conf* file on the remote ADB cluster side:

```
host    <database_name>  <remote_user>       <local_coordinator_address>      password
```

where:

- `<database_name>` — a name of the target database in the remote ADB cluster. The `all` value means all databases.
- `<remote_user>` — a user name in the remote ADB database (see the `user` option above).
- `<local_coordinator_address>` — an IP address of the coordinator host in the local ADB cluster (from which you plan to send queries). Use a subnet number. For example, `10.92.17.84/32`.

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 remote ADB cluster. To apply changes, click ***Save*** and run the service action ***Reconfigure***.

Foreign table

To create a foreign table, use the `CREATE FOREIGN TABLE` command. The basic command syntax is listed below:

```
CREATE FOREIGN TABLE [ IF NOT EXISTS ] <table_name> ( [
    <column_name> <data_type> [ COLLATE <collation> ] [ <column_constraint> [ ... ] ]
      [, ... ]
] )
    SERVER <server_name>
  [ OPTIONS ( <option> '<value>' [, ... ] ) ]
```

where:

- `<table_name>` — a foreign table name in ADB.
- `<column_name>` — a column name.
- `<data_type>` — a column data type.
- `<collation>` — a column collation.
- `<column_constraint>` — a constraint defined at the column level. The name `<constraint_name>` is given optionally. The syntax:

```
[ CONSTRAINT <constraint_name> ]
{ NOT NULL |
  NULL |
  DEFAULT <default_expr> }
```

  Possible constraints:

  - `NOT NULL` — specifies that the column is not allowed to contain null values.
  - `NULL` — specifies that the column is allowed to contain null values. It is the default behavior (unless `NOT NULL` is specified).
  - `DEFAULT` — sets the default column value equal to `<default_expr>`.

- `<server_name>` — a server name.
- `<option>` — foreign table options. For ADB to ADB Connector, all options defined at the server level (except `host`, `port`, `num_segments`) can be overwritten at the foreign table level (partially or fully). When you specify an option for a server and a foreign table at the same time, the table level takes precedence. Additionally, at the foreign table level, three more options are available (see Foreign table options below).
- `<value>` — option values.

NOTE

- To create a foreign table, you need the `USAGE` privilege on the corresponding server and on all column data types that are used in the table definition. The user who creates the foreign table becomes its owner.
- The full description of the `CREATE FOREIGN TABLE` command syntax is available in the Greenplum documentation.
- To edit a foreign table definition, use the ALTER FOREIGN TABLE command. To drop a foreign table, use DROP FOREIGN TABLE.

Foreign table options

Name	Type	Description	Default	Required
schema_name	TEXT	A schema name that is used to access a table on the remote ADB cluster side. If the parameter is undefined, a schema of the `table_name` table is used	—	No
table_name	TEXT	A target table name on the remote ADB cluster side	—	Yes
column_name	TEXT	A column name of the target table `table_name`	—	No

ADB to ADB Connector usage examples

Daria Barysheva

Contents

To Table of Contents

Select data

Insert data

This article describes how to transfer data between two ADB clusters via ADB to ADB Connector. The following prerequisites are met:

- The local ADB cluster is installed according to the Online installation guide.
- The remote ADB cluster is installed according to the Online installation guide. The coordinator host is called `bds-mdw-ext.ru-central1.internal`.
- The `adb` database exists in both clusters.
- The `adb_to_adb` user with password `123` exists in the remote cluster.
- The `adb_to_adb` user can access the remote `adb` database from the local cluster. For this purpose, change the *pg_hba.conf* file according to the ADB to ADB connector configuration article:

```
host    all  adb_to_adb       10.92.17.84/32     password
```

  where `10.92.17.84/32` is IP address of the coordinator host in the local ADB cluster (with a subnet number).

- ADB to ADB Connector is installed in the local and remote clusters according to the ADB to ADB connector installation guide.

Select data

To check that the remote ADB cluster data can be read from the local cluster, follow the steps:

1. Connect to the `adb` database in the remote ADB cluster under the `gpadmin` user (for example, via `psql`). Create the `author` table:

```
CREATE TABLE author(id INT NOT NULL, name TEXT NOT NULL)
WITH (appendoptimized=true)
DISTRIBUTED BY(id);
```

2. Insert some data into the `author` table (on the remote cluster side):

```
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');
```

3. Grant necessary permissions on the new table `author` to the `adb_to_adb` user (on the remote cluster side):

```
GRANT SELECT, INSERT ON public.author TO adb_to_adb;
```

4. Connect to the `adb` database in the local ADB cluster under the `gpadmin` user (for example, via `psql`). Create the following objects in the given order.

   - The `test_server` server:

```
CREATE SERVER test_server
FOREIGN DATA WRAPPER adb_fdw
OPTIONS (
  host 'bds-mdw-ext.ru-central1.internal',
  port '5432',
  dbname 'adb',
  use_remote_estimate 'true');
```

   - The user mapping for the `gpadmin` user:

```
CREATE USER MAPPING FOR gpadmin
SERVER test_server
OPTIONS (user 'adb_to_adb', password '123');
```

   - The `test_table` foreign table:

```
CREATE FOREIGN TABLE test_table (id INT, name TEXT)
SERVER test_server
OPTIONS (table_name 'author');
```

5. Select data from the `test_table`. Ensure that the data added on the remote cluster side is available in the local cluster:

```
SELECT * FROM test_table ORDER BY id;
```

   Result:

```
 id |       name
----+-------------------
  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
(10 rows)
```

Insert data

To check that you can add data to the remote ADB database from the local cluster, follow the steps:

1. Perform steps 1—4 that are listed above (if you have not passed them yet).
2. Connect to the `adb` database in the local ADB cluster under the `gpadmin` user (for example, via `psql`). Add some data to the `test_table` foreign table:

```
INSERT INTO test_table(id, name) VALUES
(11, 'Fyodor Dostoevsky'),
(12, 'Alexander Pushkin'),
(13, 'Mikhail Bulgakov');
```

3. Connect to the `adb` database in the remote ADB cluster under the `gpadmin` user (for example, via `psql`). Check the presence of new tuples in the `author` table:

```
SELECT * FROM author ORDER BY id;
```

   The result contains new data received from the source ADB cluster:

```
 id |       name
----+-------------------
  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
 11 | Fyodor Dostoevsky
 12 | Alexander Pushkin
 13 | Mikhail Bulgakov
(13 rows)
```

Configuration parameters

Daria Barysheva

Contents

To Table of Contents

ADB
ADB Control
ADBM
Chrony
PXF

This article describes the parameters that can be configured for ADB 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 ***Show advanced*** flag being set.

ADB

Parameter	Description	Default value
Name of default database	A name of the database created by default. Subsequently, this database is used to create most extensions via ADB service actions in ADCM	adb
Use segment mirroring	Enables synchronous replication for primary data segments, i.e. adds mirror segments to the cluster. As a result, mirror segments provide higher availability for the stored data, but require twice more space	true
Type of mirroring	Defines the mirroring type: `group` or `spread`. The default value is `auto`, which selects the optimal type automatically based on the cluster configuration: if the number of segment hosts is greater than the number of primary segments per host (see ***Number of segments per host*** below), then the `spread` type is used. Otherwise, the `group` type is used	auto
ADB MADlib Extension	Defines whether to install MADlib. MADlib is an open-source library for scalable in-database analytics that provides data-parallel implementations of mathematical, statistical, and machine-learning methods for structured and unstructured data. It provides a suite of SQL-based algorithms for machine learning, data mining, and statistics that run at scale within a database engine with no need for transferring data between ADB and other tools	false
ADB PostGIS Extension	Defines whether to install PostGIS. PostGIS is a spatial database extension for PostgreSQL that allows Geographic Information Systems (GIS) objects to be stored in the database. PostGIS includes support for GiST-based R-Tree spatial indexes and functions for analysis and processing of GIS objects	false
ADB Auto Core Dump Service	Defines whether to install the Auto Core Dump service (`systemd-coredump`) — a plugin that enables core dump files for ADB processes. This feature is useful for debugging purposes. If enabled, the system collects core dumps and saves them in the following directory: */var/lib/systemd/coredump/*. When the maximum storage size is reached (1 GB), old files will be deleted	false
Diskquota Extension	Defines whether to install the `diskquota` extension, which allows Arenadata DB administrators to limit the amount of the disk space used by schemas or roles in the database	false

Advanced

> NOTE
>
> These parameters become visible after the ***Show advanced*** flag is set.

Parameter	Description	Default value
Number of segments per host	A number of primary segments on each segment host. If the parameter is not set, then `(<number of CPU cores>) / 2` is used. When you customize a number of primary segments per host, remember that the more parallel SQL queries you plan to run simultaneously — the fewer segments per host you need	(<number of CPU cores>) / 2
Segment host data directories	Directories that are used to store data on segment hosts. The ***Number of segments per host*** value should be a multiple of the number of directories	/data1
Masters host data directory	Directories that are used to store data on coordinator and standby coordinator hosts	/data1
Arenadata configs directory name	A name of the folder that contains Arenadata configuration files. It is located in the system user home directory	arenadata_configs
Sysctl parameters	Applies the Linux kernel parameter values (using sysctl options) listed in the ***Sysctl parameters [key:value]*** block. Expand the block to view or modify the values of the applied parameters. To add a custom parameter, click ***Add property***, enter the parameter name in the ***Enter field name*** and its value in the ***Enter field value*** field	false
Disable linux security module	Disables Security Enhanced Linux (SELinux). Modifying this parameter requires reboot of cluster hosts. If you modify the parameter prior to ADB installation, set the ***Reboot cluster servers after installation*** flag in the ***Install*** or ***Reinstall*** cluster action. If you add new hosts to ADB, set the ***Reboot new segment host after installation*** flag in the ***Move Host Segment*** cluster action or set the ***Reboot new servers after installation*** flag in the ***Expand*** or ***Init Standby Master*** cluster action. If the flag is not set, stop the cluster and reboot hosts manually	false
Disable IPv6	Disables IPv6 on all ADB hosts	false
Set up MaxStartups for sshd	Sets the `MaxStartups` parameter of the OpenSSH Daemon (sshd) to `1000:30:1022`	false
Set up GRUB for DBMS	Disables Transparent Huge Pages in the Linux kernel by specifying `transparent_hugepage=never` in the GRUB configuration file. Modifying this parameter requires reboot of cluster hosts. If you modify the parameter prior to ADB installation, set the ***Reboot cluster servers after installation*** flag in the ***Install*** or ***Reinstall*** cluster action. If you add new hosts to ADB, set the ***Reboot new segment host after installation*** flag in the ***Move Host Segment*** cluster action or set the ***Reboot new servers after installation*** flag in the ***Expand*** or ***Init Standby Master*** cluster action. If the flag is not set, stop the cluster and reboot hosts manually	false

Set up ulimits for DBMS user	Adds the following resource limits to */etc/security/limits.conf*:
	`gpadmin soft    nofile  524288`
	`gpadmin hard    nofile  524288`
	`gpadmin soft    nproc   150000`
	`gpadmin hard    nproc   150000`
	The `gpadmin` user name here is the default name of the system user; it can be changed using the ***System user name*** parameter (see below)
		false

System user GID	A unique identifier of the group that the system user belongs to. A system user should be created with the same UID and GID across all cluster hosts in order to reduce problems with access to the common network storages, containing ADB backup files	2042
Disable firewall	Disables `firewalld` on the cluster hosts	false
System user name	A system user name for ADB	gpadmin
System user UID	A unique identifier of the system user. A system user should be created with the same UID and GID across all cluster hosts in order to reduce problems with access to the common network storages, containing ADB backup files	2042
Edit /etc/hosts	Defines whether to edit the */etc/hosts* file during ADB installation to include hostnames of the coordinator, standby coordinator, and segment hosts. With the default value of `false`, you must manually edit the file or configure a DNS server to ensure hostname resolution between all cluster hosts. If the value is set to `true`, make sure to also set the ***Reboot cluster servers after installation*** flag in the ***Install*** or ***Reinstall*** cluster action	false
Amount of shared memory	The amount of shared memory allocated for query metrics (in KB). The maximum size is `131072`. Restart ADB to apply changes	5120
Maximum scan node's instrumentations	The maximum number of scan node's instrumentations per query (in shmem). Restart ADB to apply changes	300

Interconnect properties

> NOTE
>
> This section becomes visible after the ***Show advanced*** flag is set.

Parameter	Description	Default value
Interconnect type	An interconnect type that should be used. Possible values: - `udpifc` - `tcp` - `proxy`	udpifc
ic-proxy auto-discovery	Whether to use automatic proxy list generation during ic-proxy interconnect configuration. A port shift is defined in the ***ic-proxy port delta*** field (by default, `2000`)	true
ic-proxy port delta	Defines an integer value that is used to increase/decrease the proxy port value relative to the port of the associated postmaster backend. Applies only in conjunction with ***ic-proxy auto-discovery*** set to `true`. Allowed values: `[-60000, 0)`, `(0, 60000]`. Please do not use `0` as the parameter value, since it will cause the following error: `Check that auto-discovery or proxy delta or gp_interconnect_proxy_addresses guc is not empty`	2000
Run ic-proxy check	Whether to run an interconnect test	false

Use dedicated cluster network

Parameter	Description	Default value
Cluster network	A subnet that is used for cluster nodes interconnection. Requires the `<subnet>/<netmask>` notation, for example `192.168.1.0/24`. If this parameter is not set, then ADCM uses addresses with the default route. A host must have exactly one address in the range. Run ***Reinstall ADB*** to apply changes	—

ADB ident server

Parameter	Description	Default value
ADB ident server	Whether to install the Ident server	false
Default deny policy for ident server	Whether to deny access by default for all users for which no explicit rules have been defined in the configuration file of the Ident server (*/etc/oidentd.conf*)	true

Crontab

Parameter	Description	Default value
Crontab maintenance scripts	Crontab maintenance scripts of the ADB system user `gpadmin`. For more information, see Work with arenadata_toolkit	—
Delete old maintenance script logs	A number of days after which maintenance script logs should be deleted	30

SSL

Parameter	Description	Default value
SSL	Whether to use SSL certificates	false
SSL Ciphers	SSL ciphers	["HIGH","MEDIUM","+3DES","!aNULL"]
Server certificate	Server certificate contents in the CRT format (**.crt* file)	—
Server private key	Server private key contents (**.key* file)	—
Root certificate	Root certificate (Certificate Authority, CA) contents in the CRT format (**.crt* file)	—
Certificate revocation list	Certificates revoked by certificate authorities	—
Custom ssl pg_hba section	The custom part of the *pg_hba.conf* file (with SSL settings)	—

UI LDAP authentication

Parameter	Description	Default value
UI LDAP authentication	Defines whether to use LDAP authentication	false
Type	A type of the LDAP server. Possible values: - `MSAD` — Microsoft Active Directory; - `FreeIPA` — 389 Directory server as a part of FreeIPA.	MSAD
LDAP Host	A domain name or an IPv4 address of the LDAP server	—
LDAP Port	A port number on the LDAP server to connect to	—
Users baseDN	A limitation of the search scope of objects in the LDAP directory that is used in user search requests	—
Groups baseDN	A limitation of the search scope of objects in the LDAP directory that is used in group search requests	—
Users filter	An LDAP filter that is used to search users. Should be specified according to the RFC 2254	(&(|(objectClass=person)(objectClass=inetOrgPerson))(cn=*))
Groups filter	An LDAP filter that is used to search groups. Should be specified according to the RFC 2254	(&(|(objectClass=group)(objectClass=groupofnames)(objectClass=groupOfUniqueNames))(cn=*))
SA Login	A user name that is used for service requests to the LDAP server	—
Password	A user password that is used for service requests to the LDAP server	—
Lowercase login	Defines whether to convert user names to lowercase	false
Lowercase group	Defines whether to convert group names to lowercase	false
Sync interval	A `cron` expression that defines how often synchronization of ADB users and groups with an LDAP server should be performed. The pg-ldap-sync utility, which comes with the ADB bundle, is used for synchronization	0 * * * *

Other

Parameter	Description	Default value
Custom pg_hba section	A custom part of the *pg_hba.conf* file	—

ADB Control

External ADB Control server parameters

Parameter	Description	Default value
External ADB Control server parameters	Defines whether to use an external ADB Control server	false
ADB Control IP-address	An external ADB Control IP address. The property becomes available after setting the ***External ADB Control server parameters*** flag	—
External ADB Control port	An external ADB Control port. The property becomes available after setting the ***External ADB Control server parameters*** flag	81

UI component parameters

Parameter	Description	Default value
ADB user name	A user name that is used by ADB Control for connecting to ADB. Such access is required to obtain information about the cluster topology and to cancel requests initiated by users	adcc
Service password	The service API user password for interaction between ADB Control and ADCM. It is recommended to change the default value	—
Servername	A server name for connecting to the ADB Control via HTTP	—
Listening port	A port on which ADB Control UI is available via HTTP	81
Xmx ram parameter	A maximum Java heap size for the UI component	8
Log level	A logs level that is used by the UI component. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO

SSL

Parameter	Description	Default value
SSL	Defines whether to use SSL encryption in ADB Control	false
CA certificate	Server certificate authority (CA) file contents in the CRT format (**.crt* file)	—
Server certificate	Server certificate contents in the CRT format (**.crt* file)	—
Server private key	Server private key contents (**.key* file)	—
Verify system endpoints' certificates	A flag that indicates whether to verify certificates of system endpoints. When using the ***Verify system endpoints' certificates*** option in the ADB Control configuration with self-signed certificates, ensure that you added the root and intermediate CA certificates to the OS trusted root certificate stores. In order to use the ***Verify system endpoints' certificates*** option with your own certificates, for proper verification, add the ADB Control IP address to the `CN` field of the *san.cnf* file (`SAN`) and exclude DNS from the `alt_names` section (leave only IP)	false
Server JKS keystore path	A path to the keystore file in the JKS format for the ADB Control server part. For example, */opt/adcc/ssl/keystore.jks*	—
Server JKS keystore password	A password that is used for connecting to ***Server JKS keystore path***	—
Server JKS truststore path	A path to the truststore file in the JKS format for the ADB Control server part. For example, */opt/adcc/ssl/truststore.jks*	—
Server JKS truststore password	A password that is used for connecting to ***Server JKS truststore path***	—
Client JKS keystore path	A path to the keystore file in the JKS format for the ADB Control client part (ADB Control/ADBM agents). For example, */opt/ssl/keystore.jks*	—
Client JKS keystore password	A password that is used for connecting to ***Client JKS keystore path***	—
Client JKS truststore path	A path to the truststore file in the JKS format for the ADB Control client part (ADB Control/ADBM agents). For example, */opt/ssl/truststore.jks*	—
Client JKS truststore password	A password that is used for connecting to ***Client JKS truststore path***	—

UI LDAP authentication

Parameter	Description	Default value
UI LDAP authentication	Defines whether to use LDAP authentication	false
LDAP type	A type of the LDAP server. Possible values: - `MSAD` — Microsoft Active Directory; - `FreeIPA` — 389 Directory server as a part of FreeIPA.	MSAD
LDAP URI	A URI for connecting to the LDAP server. Requires the following format: `<schema>://<host>:<port>`. For example, `ldap://example.com:389`, `ldaps://example.com:636`. If a port number is not specified, the standard ports are used: `389` for LDAP and `636` for LDAPS	—
Internal CA certificate in PEM format (optional)	The internal CA certificate or certificate chain in PEM format that signed the LDAP server certificate	—
Users baseDN	A limitation of the search scope of objects in the LDAP directory that is used in users search requests	—
Groups baseDN	A limitation of the search scope of objects in the LDAP directory that is used in groups search requests	—
Users OUs	A list of `OU` for user search. If the option is set, users will be searched only in the specified OUs, otherwise — within full ***Users baseDN***. Enter each `OU` value on a separate line using the `ou=<name>` format (e.g. `ou=users1`)	—
Groups OUs	A list of `OU` for group search. If the option is set, groups will be searched only in the specified OUs, otherwise — within full ***Groups baseDN***. Enter each `OU` value on a separate line using the `ou=<name>` format (e.g. `ou=groups1`)	—
Group	A group name that is used for service requests to the LDAP server	—
Login	A user name that is used for service requests to the LDAP server	—
Password	A user password that is used for service requests to the LDAP server	—
Size limit	A maximum number of records that can be returned by the LDAP server	1000
Lowercase login	Defines whether to convert user names to lowercase	false

Backend parameters

Parameter	Description	Default value
Xmx ram parameter	A maximum Java heap size for the backend component	8
Log level	A logs level that is used by the backend component. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO

Registry parameters

Parameter	Description	Default value
Xmx ram parameter	A maximum Java heap size for the registry component	2
Log level	A logs level that is used by the registry component. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO

Database parameters

Parameter	Description	Default value
Database name	A name of the PostgreSQL database where ADB Control stores data (Query DB)	adcc
Username	A user name that is used for connecting to the PostgreSQL database	postgres
Password	A user password that is used for connecting to the PostgreSQL database	—
Listening port	A listening port for the PostgreSQL database	5433
Data dir	A full path to the directory where ADB Control stores PostgreSQL data. ADB Control uses the default Docker volume directory unless otherwise specified	—

> CAUTION
>
> The ***Data dir*** directory should be empty before database initialization during ADB Control installation.

External database parameters

Parameter	Description	Default value
External database parameters	Defines whether to use an external PostgreSQL database as Query DB for ADB Control	false
Server ip	An IP address of the server where the external PostgreSQL database is stored	—
External database port	A listening port of the server where the external PostgreSQL database is stored	5432
Database name	A name of the external PostgreSQL database	adcc
Username	A user name that is used for connecting to the external PostgreSQL database	postgres
Password	A user password that is used for connecting to the external PostgreSQL database	—

Database migration parameters

Parameter	Description	Default value
Migration completion timeout	A migration completion timeout (in minutes)	5

ADQM database parameters

Parameter	Description	Default value
Data dir	A full path to the directory where ADB Control stores ClickHouse data. ADB Control uses the default Docker volume directory unless otherwise specified	—
Database name	A name of the ClickHouse database that is used as Metrics DB in ADB Control	adcc
Username	A user name that is used for connecting to the ClickHouse database	default
Password	A user password that is used for connecting to the ClickHouse database	—

External ADQM parameters

Parameter	Description	Default value
External ADQM parameters	Defines whether to use an external ClickHouse database as Metrics DB for ADB Control	false
ADQM address and port [ip_address:port]	An address of the server where the external ClickHouse database is stored. Starting with ADB Control 4.8.8, you can use clustered ClickHouse as an external database for ADB Control installation (in addition to a standalone node). To do this: 1. Create a logical cluster called `adcc_cluster` in ClickHouse. 2. On the ADB Control service configuration page in ADCM, expand a node with the parameter name ***ADQM address and port [ip_address:port]***. 3. Click ***Add property***. 4. In the dialog that opens, enter an IP address in the ***Enter field name*** field and a port number in the ***Enter field value*** field. 5. Click ***Apply*** to save changes. 6. Repeat the previous steps for each ClickHouse node.	—
Database name	A name of the external ClickHouse database	adcc
Username	A user name that is used for connecting to the external ClickHouse database	—
Password	A user password that is used for connecting to the external ClickHouse database	—

Agent parameters

Parameter	Description	Default value
Xmx ram parameter	A maximum Java heap size for ADB Control agents	2
Log level	A logs level that is used by ADB Control agents. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO
Servers list update timeout	A timeout between refreshing information about available ADB Control Backend Servers instances (in ms)	5000

ADB Control scheduler param

Parameter	Description	Default value
Xmx ram parameter	A maximum Java heap size for ADB Control scheduler	2
Log level	A logs level that is used by ADB Control scheduler. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO

Use dedicated network

> NOTE
>
> This section becomes visible after the ***Show advanced*** flag is set.

Parameter	Description	Default value
Dedicated network	A subnet that is used for ADB Control/ADBM interconnection. Requires the `<subnet>/<netmask>` notation, for example `192.168.1.0/24`. If this parameter is not set, then ADCM uses addresses with the default route. A host must have exactly one address in the range. Run the ***Reconfigure & Restart*** ADB Control action to apply changes. If the ADB Control IP address is used in the `SAN` certificate section, you should do the following before running the ***Reconfigure & Restart*** action when enabling or disabling a dedicated network for ADB Control/ADBM. In the ***HTTPS parameters enable*** section on the ADB Control configuration page in the ADCM UI, either reset the self-signed key and certificate (if they were automatically generated via a bundle) or set new key and certificate (if you added the current ones yourself)	—

ADBM

Parameter	Description
Password for adbm system-user	The password for the `adbm` user account, which is used in service API for collecting information about storage and cluster modifications that is displayed on the ***Audit*** page. It is recommended to change the default value. The password must contain at least 12 characters

Backend parameters

Parameter	Description	Default value
Xmx ram parameter	A maximum Java heap size for the ADBM backend	8
Log level	A logs level that is used by the ADBM backend. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO

Agent parameters

Parameter	Description	Default value
Xmx ram parameter	A maximum Java heap size for the ADBM agent	2
Log level	A logs level that is used by the ADBM agent. Possible values are `TRACE`, `DEBUG`, `INFO`, `WARN`, `ERROR`	INFO

External database parameters

Parameter	Description	Default value
External database parameters	Defines whether to use an external PostgreSQL database for ADBM	false
Server ip	An IP address of the server where the external database is stored	—
External database port	A listening port of the server where the external database is stored	5432
Database name	A name of the external database	adbm
Username	A user name that is used for connecting to the external database	postgres
Password	A user password that is used for connecting to the external database	—

Database migration parameters

Parameter	Description	Default value
Migration completion timeout	A migration completion timeout (in minutes)	3
Xmx ram parameter GB	The maximum memory pool allocated for a Java Virtual Machine (JVM) in GB	1

Chrony

Parameter	Description	Default value
NTP servers	Addresses of valid NTP servers. For example, `0.ru.pool.ntp.org`, `1.ru.pool.ntp.org`. If this parameter is set, then coordinator takes the time from NTP servers, standby coordinator — from coordinator or NTP servers, segment hosts — from coordinator or standby coordinator. If the parameter is not set or all NTP servers become unavailable, then coordinator uses its local clocks, standby coordinator uses the coordinator or local clocks, and segment hosts get the time from coordinator or standby coordinator	—

PXF

Parameter	Description	Default value
Use custom JAVA_HOME	Defines whether to use a custom `JAVA_HOME` path	false
JAVA_HOME	A custom `JAVA_HOME` path. Should lead to the JDK directory. The property becomes available after setting the ***Use custom JAVA_HOME*** flag. If empty, then the default system value is used: - */usr/lib/jvm/java-17-openjdk-amd64* (Ubuntu) - */usr/lib/jvm/jre-17-openjdk* (RED OS) - */usr/lib/jvm/jre-17* (AltLinux) For Astra Linux, the ***JAVA_HOME*** field is mandatory starting with ADB 6.27.1.58	—
PXF_JVM_OPTS	Sets `PXF_JVM_OPTS`	-Xmx2g -Xms1g
PXF_OOM_KILL	Defines whether the PXF service should automatically terminate when the PXF JVM detects an out-of-memory condition	false
PXF_OOM_DUMP_PATH	The absolute path to a file where you want the PXF service to write Java heap dumps when the PXF JVM detects an out-of-memory condition. If you specify a directory, the PXF JVM will write a new file to the directory every time it detects an out-of-memory condition. Leave empty to prevent the creation of heap dump files	—
PXF log4j properties	log4j properties for PXF	—
Use custom PXF_BASE	Defines whether to use a custom `PXF_BASE` path	false
PXF_BASE	A custom `PXF_BASE` path for PXF installation. The property becomes available after setting the ***Use custom PXF_BASE*** flag	/var/lib/pxf
Encryption	Defines whether to use encryption	false
Keystore file	An absolute path to the JKS keystore file. For example, */var/lib/pxf/conf/pxfkeystore.jks*	—
Keystore password	A JKS keystore password	—
Encryption key alias	An encryption key alias. Do not enter a private key in this field	—

SSL

Parameter	Description	Default value
SSL	Defines whether to use SSL encryption in PXF	false
CA certificate	Server certificate authority (CA) file contents in the CRT format (**.crt* file)	—
Server certificate	Server certificate contents in the CRT format (**.crt* file)	—
Server private key	Server private key contents (**.key* file)	—
Server JKS keystore password	A password that is used for connecting to ***Server JKS keystore path***	—
Server JKS truststore password	A password that is used for connecting to ***Server JKS truststore path***	—
Server JKS keystore path	A path to the keystore file in the JKS format for the PXF server part. For example, */var/lib/pxf/conf/ssl/keystore.jks*	—
Server JKS truststore path	A path to the truststore file in the JKS format for the PXF server part. For example, */var/lib/pxf/conf/ssl/truststore.jks*	—

Supported extensions

Daria Barysheva

To Table of Contents

Extensions are special modules that provide additional database functionality by bundling several SQL objects into a package to use them as a single unit. Extensions provide various functions, operators, data types, etc.

Below is a full list of extensions that are available in Arenadata DB.

Extension name	Version	Description	Enterprise only	Deployment type
adb_fdw	1.0	Implements a foreign data wrapper that provides two-way interaction between ADB clusters	+	ADB to ADB Connector
address_standardizer	3.3.2	Rule-based address standardizer that is used to parse an address into constituent elements	-	ADB service → Install PostGIS action
address_standardizer_data_us	3.3.2	Provides sample rule tables for US address data	-	ADB service → Install PostGIS action
arenadata_toolkit	1.0	Used for manipulating the arenadata_toolkit schema and its objects	-	ADB service
btree_gin	1.3	Provides support for indexing common data types in GIN	-	`CREATE EXTENSION btree_gin;`
citext	1.6	Implements a data type for case-insensitive character strings	-	`CREATE EXTENSION citext;`
dblink	1.2	Supports connections to other PostgreSQL databases from within a database session	-	ADB service
diskquota	2.3	Allows you to limit the amount of disk space used by schemas, roles, or tablespaces in a database	-	ADB service → Install diskquota action
file_fdw	1.0	Implements the foreign data wrapper for server-side flat files	-	`CREATE EXTENSION file_fdw;`
fuzzystrmatch	1.1	Provides functions to determine similarities and distance (e.g. Levenshtein) between strings based on various algorithms	-	`CREATE EXTENSION fuzzystrmatch;`
gp_distribution_policy	1.0	Allows you to check distribution policies in a cluster	-	`CREATE EXTENSION gp_distribution_policy;`
gp_exttable_fdw	1.0	Converts an external table to a foreign table	-	`CREATE EXTENSION gp_exttable_fdw;`
gp_internal_tools	1.0.0	Provides different internal tools for Greengage DB	-	`CREATE EXTENSION gp_internal_tools;`
gp_legacy_string_agg	1.0.0	Implements the legacy single-argument function `string_agg` in Greengage DB	-	`CREATE EXTENSION gp_legacy_string_agg;`
gp_sparse_vector	1.0.1	Implements SParse vectors in Greengage DB	-	`CREATE EXTENSION gp_sparse_vector;`
gp_toolkit	1.8	Provides information from system catalogs, cluster logs, and environment status	-	`CREATE EXTENSION gp_toolkit;`
gpadcc	0.8	Used by the ADB Control service	+	ADB Control service
hstore	1.6	Implements a data type for storing sets of key/value pairs within a single data field	-	`CREATE EXTENSION hstore;`
isn	1.2	Provides support for international product numbering standards	-	`CREATE EXTENSION isn;`
kadb_fdw	0.16	Implements a foreign data wrapper that provides interaction between Kafka and ADB	+	Kafka to ADB Connector
ltree	1.1	Implements a data type for hierarchical tree-like structures	-	`CREATE EXTENSION ltree;`
madlib	2.1.0	A scalable in-database analytics library	-	ADB service → Install MADlib action
orafce	4.9	Implements SQL functions and operators that emulate a subset of functions and packages from the Oracle database	-	`CREATE EXTENSION orafce;`
pageinspect	1.9	Used to inspect the contents of database pages at a low level	-	`CREATE EXTENSION pageinspect;`
pg_buffercache	1.4.1	Provides real-time access to the contents of the shared buffer cache	-	`CREATE EXTENSION pg_buffercache;`
pg_hint_plan	1.3.9	Allows controlling the query planner’s execution plan by using hints embedded in SQL comments	-	`CREATE EXTENSION pg_hint_plan;`

pg_stat_statements	1.7	Allows tracking planning and execution statistics of all SQL statements executed by a server	-	In the `adb` database, run:
				`CREATE EXTENSION pg_stat_statements;`
				On the coordinator host, add `pg_stat_statements` to the `shared_preload_libraries` server configuration parameter, and then restart the DBMS:
				`$ gpstop -r`

pg_trgm	1.4	Provides text similarity measurement and index searching based on trigram matching	-	`CREATE EXTENSION pg_trgm;`
pgcrypto	1.3	Implements cryptographic functions (encryption/decryption)	-	`CREATE EXTENSION pgcrypto;`
plperl	1.0	Provides support for PL/Perl procedural language	-	`CREATE EXTENSION plperl;`
plperlu	1.0	Provides support for PL/PerlU untrusted procedural language	-	`CREATE EXTENSION plperlu;`
plpgsql	1.0	Provides support for PL/pgSQL procedural language	-	ADB service
plpython3u	1.0	Provides support for PL/Python3U untrusted procedural language	-	`CREATE EXTENSION plpython3u;`
plpythonu	1.0	Provides support for PL/PythonU untrusted procedural language	-	`CREATE EXTENSION plpythonu;`
postgis	3.3.2	Provides PostGIS geometry, geography, and raster spatial types and functions	-	ADB service → Install PostGIS action
postgis_raster	3.3.2	Provides support for raster data	-	ADB service → Install PostGIS action
postgis_sfcgal	3.3.2	Implements PostGIS SFCGAL functions	-	ADB service → Install PostGIS action
postgis_tiger_geocoder	3.3.2	Provides the PostGIS TIGER geocoder and reverse geocoder	-	ADB service → Install PostGIS action
postgres_fdw	1.0	Implements a foreign data wrapper for remote PostgreSQL servers	-	`CREATE EXTENSION postgres_fdw;`
pxf	2.1	Allows you to access unmanaged data	-	PXF service → ***Install*** action
pxf_fdw	1.0	Implements a foreign data wrapper for PXF	-	PXF service → ***Install*** action
sslinfo	1.2	Provides information about SSL certificates	-	`CREATE EXTENSION sslinfo;`
tablefunc	1.1	Provides functions that return tables (e.g. `crosstab`)	+	`CREATE EXTENSION tablefunc;`
tkh_fdw	1.0	Implements a foreign data wrapper that provides two-way interaction between ADB and ClickHouse	+	ADB ClickHouse Connector
uuid-ossp	1.1	Provides functions to generate universally unique identifiers (UUIDs)	-	`CREATE EXTENSION "uuid-ossp";`

SQL commands not supported in ADB

Daria Barysheva

To Table of Contents

Though Arenadata DB is based on PostgreSQL, some SQL commands and operators are not supported in ADB. This is due to the features of distributed databases. Below is a list of SQL commands that are currently missing in ADB.

NOTE

For more information on all SQL commands (including supported ones), see the Greenplum documentation.

SQL command
ALTER SYSTEM
ALTER TRIGGER
ALTER PUBLICATION
ALTER SUBSCRIPTION
COMMIT PREPARED
CREATE CONSTRAINT TRIGGER
CREATE PUBLICATION
CREATE SUBSCRIPTION
CREATE TRIGGER
DROP OWNED
DROP PUBLICATION
DROP SUBSCRIPTION
DROP TRIGGER
`LATERAL` JOIN type
PREPARE TRANSACTION
ROLLBACK PREPARED
SET CONSTRAINTS

ADB 7 releases

Daria Barysheva

Contents

To Table of Contents

7.4.1

7.4.0

7.2.0

7.4.1

7.4.1.1

Date: 23.12.2025

New features

- Implemented ADBM 2.8.1
- Implemented ADB Control 4.15.1

Bug fixes

- The ***Move Host Segment***, ***Expand***, and ***Init Standby Master*** cluster actions reboot only new hosts now
- Fixed an issue with switching to the standby coordinator when using the ***force activation*** mode
- Fixed an issue where the ADB to ADB connector could hang during parallel retrieve cursor closure
- Fixed an issue where reading from a foreign table within a transaction could fail
- Fixed an issue with dispatching `DROP` commands to segments when the object is missing on the coordinator
- Fixed an issue where temporary files could remain in the system if a query with a materialized CTE was interrupted by an error

7.4.0

7.4.0.1

Date: 28.11.2025

New features

- The release includes the following components: ADBM, ADB Control, and connectors (ADB ClickHouse, ADB to Kafka and Kafka to ADB, ADB to ADB). Offline installation is supported
- Migrated ADB to Greengage DB
- DBeaver has been adapted to work with Greengage DB. Use version 25.2.3 or later
- Added support for Linux cgroup v2, allowing you to limit disk throughput
- Implemented orphaned file cleanup
- Added the ability to define a distribution key for `Redistribute Motion` when using foreign tables
- Added support for LDAP authentication
- Added support for LLVM 18
- Implemented TLS options for PXF external tables and foreign data wrappers (FDWs)
- Implemented ADBM 2.8.0
- Implemented ADB Control 4.15.0

Improvements

- Refactored `arenadata_toolkit`. The extension has been aligned with the latest Greengage DB architecture, and several components have been reorganized with part of the functionality moved into the `gp_toolkit` extension
- Implemented a precheck to ensure a standby coordinator cannot be added when mirroring is not enabled
- Refactored OS parameter handling and added a precheck to validate required system settings
- Added support for selecting `group` or `spread` mirroring
- Added SSL support for connections between `pxf_fdw` and the PXF service
- Enabled support for the `DEXTRA_DYNAMIC_MEMORY_DEBUG` macro
- Disallowed inlining of CTEs containing DML operations
- Raised a planner error for `SELECT INTO` queries containing modifying CTEs
- Enabled JSON output for `EXPLAIN ANALYZE` statistics
- Prevented WAL removal during recovery when `archive_mode=always`
- Added an explicit error when specifying `ENCODING` for foreign-table columns
- Added `EXPLAIN ANALYZE` statistics for queries with DML inside CTEs
- Added a default transformation flag for `gpfdist` applied automatically when a table has no transformation level defined
- Ensured `gpcloud` compatibility with OpenSSL 1.1.1
- Added a check to ensure the previous expand action completed successfully before starting the next one

Bug fixes

- Fixed an issue where temporary schemas were not cleaned up after segment failures
- Fixed an issue where `pg_dump` generated an invalid `DROP EXTERNAL TABLE` statement that prevented proper restoration
- Fixed `gplogfilter` so that time-range filters correctly skip irrelevant files
- Fixed an issue where disk space was not promptly reclaimed after dropping temporary tables
- Fixed an issue where GUCs on segments were not updated when running `gpstop -u`
- Fixed a crash when running `EXPLAIN (ANALYZE, FORMAT JSON)` for `gp_toolkit.gp_table_size_skew_coefficients()`
- Fixed an issue where inserts failed if `default_tablespace` pointed to a non-existing tablespace
- Fixed incorrect calculation of space used by spill files when multiple processes run in the same session
- Fixed an issue where `VACUUM` could corrupt the bitmap index of an AO table on mirror segments
- Fixed an issue where LDAP connections ignored `hostssl` and did not use SSL for the specified users
- Fixed an issue preventing the use of wildcard characters in interconnect addresses
- Fixed an issue in the `tablefunc` extension that prevented using DML queries
- Fixed incorrect memory allocation when generating statistics for hash join queries

Misc

- MADlib is now delivered as an extension

7.2.0

7.2.0.1

Date: 19.09.2024

Improvements

- The output of the `EXPLAIN` command in JSON mode has been fixed to provide more accurate query execution details
- GPDB compilation will now fail if compiler warnings occur, improving build quality
- Implemented error reporting for invalid operations involving CTEs in the `SELECT INTO` statements
- Interruption of LDAP requests by signals will no longer be treated as an authentication error, enhancing stability
- Official support for Ubuntu 22.04 on x86-64 platform has been added
- The query analysis logic has been moved to the PostgreSQL core, improving performance and integration
- Simplified the `dbid` detection logic to improve the reliability of the `pg_rewind` command
- Optimized the row counting mechanism in replicated tables for DML operations
- Updated the data distribution logic to ensure proper handling of combined hash specifications
- Python 3.11 is now supported, enhancing container operations
- Updated the CTE processing mechanism to always materialize data if it is used outside of `SELECT`
- Incorporated the latest `timezone` data updates for more accurate time zone management
- Updated JIT support to enhance query performance in GPDB
- Added documentation for running JIT tests and updated LLVM support to the latest versions
- Fixed Dockerfile for proper execution of the GPDB 7 test suite and added support for Java 11

Bug fixes

- Resolved resource leaks during errors while executing the `COPY TO/FROM PROGRAM` commands
- Addressed compilation warnings to improve compatibility with modern compilers
- Corrected an issue that occurred when switching the access method to columnar for tables
- Resolved an issue that caused a segmentation fault when calculating compression ratios for AO tables
- Resolved an issue where temporary schemas were not properly cleaned up after segment failures
- Addressed errors related to incorrect row access during the analysis of AOCO tables
- The `gpstop -u` command logic has been updated to ensure that GUC parameters are properly applied on segments

ADB 7 known issues

Anton Vasilev

To Table of Contents

Issue	From	Fixed	Workaround

Ansible task hangs indefinitely on a specific host during package installation or APT cache update	7.4.0	—	Solution:
			1. On the target host, identify the processes locked by APT:
			`$ ps aux | grep apt`
			2. Terminate found processes:
			`$ kill <PID>`
			Once the hanging processes are terminated, the task will retry automatically. Make sure to terminate the hanging processes on all affected hosts — in this case, the ADCM action will resume its operation, and you can restart it in the ADCM UI.
			As a prevention measure, disable and stop `apt-daily-upgrade.timer`, `apt-daily.timer`, and `apt-daily.service` to prevent background auto-updates from locking APT:
			`$ sudo systemctl disable apt-daily-upgrade.timer apt-daily.timer apt-daily.service`
			`$ sudo systemctl stop apt-daily-upgrade.timer apt-daily.timer apt-daily.service`

ADB 6 releases

Daria Barysheva

Contents

To Table of Contents

- 6.29.1
- 6.29.0
- 6.27.1
- 6.26.2
- 6.26.0
- 6.25.2
- 6.25.1
- 6.24.3
- 6.23.5
- 6.23.3
- 6.22.1
- 6.22.0
- 6.21.1
- 6.21.0
- 6.20.4
- 6.20.3
- 6.20.1
- 6.19.3
- 6.19.1
- 6.18.2
- 6.18.0
- 6.17.5
- 6.17.2
- 6.17.1
- 6.16.3
- 6.16.2
- 6.16.1
- 6.16.0
- 6.15.0
- 6.14.1
- 6.14.0
- 6.13.0
- 6.12.1

6.29.1

6.29.1.1

Date: 08.10.2025

New features

- Added the ability to define a distribution key for `Redistribute Motion` while using foreign tables
- Implemented the `--on-error-continue` mode for the `gpbackup` component, enabling the restore process to continue upon encountering a table error during DataDomain recovery
- Added support for AltLinux 10.2 SP. AltLinux 8.6 is no longer supported
- Implemented ADBM 2.6.6

Bug fixes

- Fixed a bug that caused increased memory consumption when exporting data via PXF through an external table
- Fixed a bug where a hanging process continued reading data via PXF on a segment after its parent process on the master host was interrupted

6.29.0

6.29.0.3

Date: 17.09.2025

New features

- Added support for Astra Linux 1.7.6 "Voronezh" (for the Enterprise edition of ADB)
- Implemented the orphaned file cleanup functionality
- Added the ability to set `USE_LOCAL_PXF_TIMEZONE_READ` and `USE_LOCAL_PXF_TIMEZONE_WRITE` in the *pxf-site.xml* file
- Implemented ADB Control 4.14.2
- Implemented ADBM 2.6.5

Improvements

- Backported the Dynamic Shared Areas (DSA) module
- Removed the Materialize operation of sorted internal path in Merge Join for the Postgres optimizer
- Implemented password escaping in PXF, ADB Control, and ADBM

Bug fixes

- Fixed an issue with `ALTER TABLE … SET WITH (reorganize = true)`, which resulted in the table type being changed from heap to AOC and in data loss when the `gp_default_storage_options` configuration parameter was set to `appendonly=true, orientation=column`
- Fixed an issue that caused the `COPY` command to hang while working with column-oriented tables where all columns had been dropped
- Fixed an issue with `gp_check_functions` when working with custom tablespaces
- Fixed `gpperfmon ident error` in the Precheck cluster action

Misc

- This is the first ADB release based on Greengage DB

6.27.1

6.27.1.67

Date: 12.08.2025

New features

- Implemented ADBM 2.5.7

6.27.1.66

Date: 05.08.2025

New features

- Implemented ADB Control 4.13.3
- Implemented ADBM 2.5.5

Bug fixes

- Fixed: the `archive` process lost segments after switching roles
- Fixed the error of updating ADB from 6.25.2.52 to 6.27.1.61 on ALT 8.4 SP

6.27.1.63

Date: 23.06.2025

New features

- Added support for Hashi_vault implemented in ADCM
- Added HTTPS support for ET
- Implemented ADBM 2.5.0
- Implemented ADB Control 4.13.0

Improvements

- Migrated from GnuTLS to OpenSSL
- Improved monitoring of memory consumption by technical processes
- Optimized query plan creation for "edge" cases: temporary tables without collected statistics that use the `isNull` filter
- For ADB to ADB connector, optimized the planner to work with queries that contain the `IS NULL` predicate
- Optimized GPORCA for correlated subqueries with grouping
- Added the ability to use special symbols like `“$%&{%[”` in custom LDAP passwords

Bug fixes

- In the `diskquota` configuration file, started removing records that are related to deleted tables or schemas
- Improved readability and relevance of the error message when restoring from multi-page WAL files that contain broken pages
- Fixed the negative query cost in JSON output
- Fixed the error of ADB updates from versions 6.24.3.47 and earlier, caused by the presence of the mandatory field `JAVA_HOME`
- Fixed the `gpperfmon` stop after switching to standby master
- Fixed the need to disable `Authentication type` for ADB update
- Implemented early cleanup of unused generated data, for example, when using `LIMIT` in a query

Misc/Internal

- Starting with ADB 6.27.1.63, the minimal ADCM version is 2.6.0
- Extended restrictions on using multiple inline CTE within a single data query
- Fixed the Java version check during installation
- The ADB upgrade to 6.27.1.63 is only tested for ADB 6.26.2.55 and later. Upgrades from earlier versions are not tested

6.27.1.61

Date: 14.04.2025

Bug fixes

- Fixed the session variables (GUC) sync bug that caused data loss on committed transactions

6.27.1.60

Date: 07.03.2025

New features

- Upgraded `gpbackup` to 1.30.5_arenadata18
- Upgraded PXF to 6.12.2_arenadata23
- Upgraded `diskquota` to 2.3.0_arenadata8
- Upgraded `gptkh` to 1.1.6
- Upgraded `pgbackrest` to 2.52_arenadata8
- Added the `--jobs` option to `gprestore` for parallel restore and improved helper synchronization
- Introduced the endpoint for online DB object size tracking with minimal performance impact
- Added mutual TLS support for secure ADB-PXF communication
- Added the new SSL management action for the PXF service in the ADB bundle
- Implemented ADBM 2.4.0

- Implemented ADB Control 4.12.0

Improvements

- Reduced S3 metadata calls by skipping unnecessary file validations and marker deletions
- Improved PXF fragment distribution by shifting from modulo to prime-based division
- Updated the cluster check to use `preferred_role` for balanced segment distribution
- Skipped `crontab` comment lines in configuration tasks
- Fixed full uninstall of PXF to preserve package configuration directories
- Patched docker-compose for compatibility with docker-engine-27.1.1 on ALTLinux 8
- Refactored `diskquota` initialization to reduce memory consumption at startup
- Optimized workfile compression memory usage with a dedicated zstd memory context and properbuffer cleanup

Bug fixes

- Fixed the `SIGSEGV` errors on segments
- Resolved the `Init standby master` crash by manually installing python-lxml
- Fixed the Hive connectivity error in PXF 6.11 by ensuring the correct Kerberos-enabled UGI is used
- Resolved timing issues in `gpbackup_helper` so that pipes are created before `gprestore` reads them
- Fixed the `SIGSEGV` error in the ClickHouse FDW during the `INSERT` operations
- Corrected the PXF installation failure on non-ADB hosts by restricting PXF tasks to ADB nodes
- Enhanced synchronization between helpers and `gprestore` to avoid using deleted pipes during parallel restores

Misc/Internal

- Revised Docker *daemon.json* updates via regex for safe basic option changes
- Standardized the oident service name to `oidentd.service` for consistent install/uninstall

6.27.1.59

Date: 27.12.2024

New features

- Introduced timestamp partitioning in PXF
- Enabled flexible selection of fragmenter, accessor, and resolver in PXF
- Added SSL connectivity options for external databases in ADB Control (Metrics DB, Query DB, Offload DB)
- Added the Move Host Segment action to prepare and move segments to new hosts
- Implemented ADBM 2.3.2

Implemented ADB Control 4.11.0

Improvements

- Refined the `diskquota` extension: reduced memory consumption and improved stability
- Improved error handling logic when working with S3 external tables
- Allowed Ident authentication for `gpperfmon`
- Streamlined the fault injector configuration
- Implemented the pattern-based validation for fields that accept IP or domain names (ADB Control/ADBM)
- Implemented the ability to support `SERIAL` columns and default constraints in external tables

Bug fixes

- Fixed missing prototype warnings and compilation issues in GPDB
- Resolved the `high VMEM usage` false alarms
- Addressed incorrect baseurl formation in **.repo* files on RedOS 7.3
- Fixed errors that occurred during the uninstallation of ADB Control components
- Prevented statistics from being erroneously erased by `VACUUM` in AO tables

Misc/Internal

- Removed redundant `extern` keywords for cleaner and more consistent code
- Removed the etcd component from ADBM

6.27.1.58

Date: 14.11.2024

New features

- Upgraded `gpbackup` to 1.30.5_arenadata16
- Upgraded `diskquota` to 2.3.0_arenadata6
- Upgraded `kafka-greenplum-pxf-connector` to 1.1.0-pxf-6.x
- Upgraded `pgbackrest` to 2.52_arenadata6
- Upgraded `tkh-connector` to 1.5.0
- Full interconnect support for the Proxy type has been integrated into ADB (see ***ADB → Interconnect properties*** in Configuration parameters)
- ADB is now compatible with RedOS 7.3
- Configuration files are now validated without using log files
- PXF is now compatible with ADH version 3.3.6

- Implemented ADBM 2.2.3
- Implemented ADB Control 4.10.3

Improvements

- The Tkhemali/ADB ClickHouse Connectors now support messages larger than 4096 characters
- Edge case handling in `gpbackup` has been improved
- `gprestore` now uses the actual `attnum` values from the catalog during restoration, ensuring data integrity
- A new filter option is available in `pgbackrest` for the `archive-get` and `restore` commands
- `gpbackup_s3_plugin` now includes the `logger_verbosity` option
- Nested exception information is now available in PXF logs when needed
- The skip-file usage in `gprestore` has been enhanced
- `gprestore` now shows actual error messages more frequently
- The PXF platform has been updated to use Spring Boot 3.3.2
- An option to save the `gpexpand` logs prior to rollback has been introduced
- ADB and all its components now require Java 17
- *.bash_profile* is now supported on Astra and Ubuntu

Bug fixes

- Log rotation scripts now better accommodate database names
- `gpexpand` has been optimized to handle partitioned tables that span multiple schemas more effectively
- The logging behavior in `gpperfmon` has been unified
- The PLContainer Docker instance is fully operational again
- Optimizations for `MIN`/`MAX` are no longer applied to inner plans of lateral joins
- GUC sync will no longer forget about temp tables
- `gpbackup` and `gprestore` no longer hang if helper processes fail
- JSON output is now available for the `EXPLAIN ANALYZE` statistics
- Schema types are now correctly backed up
- The Kafka Connector automatically sets up `pxf-profile` after installation
- The ADB ClickHouse Connector now handles cyrillic characters correctly
- Restoration of `arenadata_toolkit` has been improved
- Improvements in PXF ensure user impersonation is handled more accurately
- A delete action for failed ADBM instances has been added to the bundle
- ETags will be parsed case-insensitive now

- Support for over 512 fields has been added to the ADB ClickHouse Connector
- The *core-app.log* files are now included in the log rotation
- Clusters running in the standalone mode with extensions now shut down correctly
- Software package cache updates automatically when needed

Misc/Internal

- In CentOS and Astra Linux, before installation of ADB 6.27.1.58 or upgrade to this version, you should manually install Java 17 on cluster hosts and fill in the ***JAVA_HOME*** parameter on the cluster configuration page

6.27.1.57

Date: 29.08.2024

New features

- Significantly enhanced and extended the performance and scope of PXF push-down
- Upgraded `gptkh` to 1.1.5
- Upgraded `pgbackrest` to 2.52
- Upgraded PXF to 6.10.1_arenadata15
- Upgraded `gpbackup` to 1.30.5_arenadata15
- Implemented ADBM 2.1.2
- Implemented ADB Control 4.9.1

Improvements

- Queries on unresponsive segments do not longer hang indefinitely
- `gprestore` is now more tolerant of plugin outputs sent to stderr
- PXF now supports push-down filters for the following types: `NUMERIC`, `TIMESTAMP`, and `TIMESTAMPTZ`
- PXF now supports years with more than four digits
- PXF now supports push-down for immutable and some stable functions
- `gpadmin` logs now rotate automatically
- Updated timezones to the latest version

Bug fixes

- The `COPY` command now handles partitioned tables more accurately
- The `COPY` command now can use the `client_connection_check_interval` GUC
- `gprestore` no longer creates a large number of zombie processes

- `SEGFAULT` on the master node are now less likely
- `gprestore` now handles missing or failed tables more effectively

6.27.1.56

Date: 27.06.2024

New features

- Upgraded `pgbouncer` to 1.22.1
- Upgraded PXF to 6.10.1_arenadata14
- Upgraded `gpbackup` to 1.30.3_arenadata14
- Upgraded `gptkh` to 1.1.4
- Upgraded `tkh-connector` to 1.4.1
- Added the ability to separate tablespaces for temporary tables from the spill/sort temporary file storage by setting the new GUC `temp_spill_files_tablespaces`
- ADB now supports Ubuntu OS
- The ADB to Kafka Connector now supports SSL
- Added the ADBM CLI to simplify backup and restore processes
- Implemented ADBM 2.0.4
- Implemented ADB Control 4.8.8

Improvements

- ADB bundle: the ***Advanced*** section has been removed from the cluster action ***Expand***
- ADB bundle: the ***Advanced*** section in the ADB service configuration is now smarter
- `gpfdist` now has the default transformation flag
- Grafana dashboards and ADCM API now include more thread-related metrics for PXF
- Added aliases to the ORCA plan (`EXPLAIN`)
- ORCA requires fewer columns for delete operations
- `gpexpand` now handles already expanded tables properly
- PXF can better export JSON/JSONB fields to Postgres
- PXF will never ignore the *jdbc-site.xml* settings to determine the JDBC driver
- Improved performance of the DDBoost plugin
- Improved the ADB performance on CPUs with the Power architecture
- `gprestore` works better with the `--cluster-resize` parameter
- Improved security of the ADB source code

- ADB Control now supports external ADQM
- ADB Control and ADBM no longer use Nginx
- ADB bundle: the *pg_hba.conf* file is now validated before applying
- ADB bundle: when SQL files are used instead of text to run SQL commands, the `ON_ERROR_STOP` option is added

Bug fixes

- The ADB Control connection URLs automatically switch to standby when it is activated
- ORCA generates a plan for queries with `GROUP BY` by a non-primary key
- ORCA generates correct plans for queries where `EXCEPT` is above `LEFT JOIN`
- Improved the locking strategy for partitioned tables
- The restore process will not remove old WAL files if `archive_mode=always`
- LDAP authorization requests are processed more correctly
- The `LC_MONETARY` and `LC_TIME` settings are now properly synchronized across query processes
- The `adb_collect_table_stats` query now works for partitioned tables
- Column definitions in the `gpbackup` metadata will no longer double
- The `COPY FROM PROGRAM` command now properly closes the program in case of an error
- PostGIS extensions now work during the cluster installation tasks
- The `ALTER TABLE ADD PARTITION` command now handles dropped columns more properly
- Improved performance when working with ClickHouse
- `gpbackup` has a lower probability of Out Of Memory (OOM) errors
- The address availability check for UDP interconnect has been reinstated
- Improved the `gpbackup` speed
- `gprestore` now handles errors better
- ADB Control no longer ignores changes in the scheduler configuration files
- Improved the resource release callback logic for `tkh_fdw`
- The ADB Control reconfiguration no longer fails after adding incorrect options
- The PXF locale can now be configured via ADCM
- Improved handling `tkh_fdw` with the `CDB_MOTION_DEBUG` flag

6.26.2

6.26.2.55

Date: 11.04.2024

New features

- Synced with upstream Greenplum Database 6.26.2
- Upgraded `pgbackrest` to 2.50_arenadata4
- Upgraded `tkh-connector` to 1.4.1
- Upgraded PlanChecker to 1.1.1
- Upgraded `gptkh` to 1.1.3
- Upgraded PXF to 6.9.0_arenadata13
- Upgraded `gpbackup` to 1.30.3_arenadata10
- Upgraded `librdkafka` to 1.9.2
- Added the PXF cluster option to reload profiles
- Added the option to change the `PXF_BASE` directory
- Added `tablespace_location` to `arenadata_toolkit.db_files_current`
- Added the `parallel_processes` option to the ***Redistribute*** cluster action
- Implemented ADBM 1.7.3
- Implemented ADB Control 4.7.5

Improvements

- Unset the `gpadmin` default password in the ADB bundle
- Reset ADB on Astra Linux from default Axiom Java to openjdk
- Replaced Java 8 with Java 11 in all components of Arenadata DB
- Added a date to the logback parameters pattern
- Added the ability to use a subdirectory for a custom Docker registry
- Implemented toast size calculation for `adb_relation_storage_size`
- Added support for Greenplum specific query tree nodes to query jumbling logic
- Removed the confusing option ***Force Enable Monitoring*** from PXF service parameters
- Added the LDAP settings check in the ADB service during the ***Precheck***, ***Check***, ***Install***, ***Reinstall*** actions

Bug fixes

- ADB ClickHouse Connector did not use the `USER MAPPING` on segments
- The *adcc_agent_monitor* script did not exist on new hosts after the expand
- The ADB bundle option ***custom enterprise repo*** did not work with Astra Linux
- `db_datfrozenxid_alerter` returned the `script is already running` error

Scheduler app logs were not written to files

The ADB bundle error `Could not find the requested service diamond` after removing ADM cluster

The ADB bundle on Astra Linux ignored the `gpinitsystem` error

The *operation.py* scripts failed with a DB name having contained special characters

Backup Manager pages were displayed on UI when ADBM was not installed

Host system metrics were always empty

The `gpbackup_s3_plugin` and `gpbackup_helper` processes remained after running `gprestore` unsuccessfully

Fixed the progress bar concurrency issue with `gpbackup`

Fixed possible issues detected in ORCA by svacer

Fixed possible memory leaks detected by svacer

Fixed core dump in `transformValuesClause` when there were no columns

Fixed incorrect memory assumption for hash join statistics

Fixed the address availability check when setting up UDP sockets in IPv6 disabled environments

`diskquota` could work incorrectly because of a huge memory leak if tuple copies were not freed after use

Disk space was not reclaimed to OS right after drop of a temporary heap relation

`gpbackup` did not use a shared snapshot for the first table in worker processes

Fixed data restore with the `--backup-dir` and `--resize-cluster` parameters to a cluster with fewer segments

Fixed the empty `query_text` field in the `queries_history` table

ORCA extracted all columns of an AO table when deleted

ORCA generated an incorrect plan with the `EXISTS` clause for a partitioned table

Fixed inconsistencies with Shared Scans in a postgres planner

Fixed the `unable to login` error that occurred when a server used `tset`

Fixed missing system attributes for an AO table

Fixed inserting the `#` symbol into a table on the ClickHouse side

6.26.0

6.26.0.53

Date: 28.12.2023

New features

Synced with upstream Greenplum Database 6.26.0

Added new metrics for PXF Monitoring: `pxf status` and `uptime`

Added the ability to choose a strategy for vacuuming in `arenadata_toolkit`

- Added support for `plpython3u`
- Upgraded PlanChecker to 1.1.0
- Upgraded the *adb-community/clickhouse-server* image to 22.8.6.71-alpine
- Upgraded `gptkh` to 1.1.2
- Upgraded PXF to 6.8.0_arenadata12
- Upgraded `gpbackup` to 1.29.4_arenadata7
- Upgraded `pgbouncer` to 1.21.0
- Upgraded `pgbackrest` to 2.47_arenadata4
- Upgraded `adb_ddp_plugin` to 1.1.1
- Upgraded `diskquota` to 2.3.0_arenadata5
- Upgraded `tkh-connector` to 1.4
- Upgraded `plcontainer` to 2.3.1
- Implemented ADB Control 4.6.4
- Implemented ADBM 1.6.3

Improvements

- Replaced the inefficient `pg_catalog.pg_partitions` view in `arenadata_toolkit`
- Added ADB Control and ADBM descriptions to the main pages of corresponding services in ADCM
- Renamed the `Use nobarrier` configuration parameter for master and segments to avoid ambiguity
- Added switching from Cgroups v2 to v1 for Astra Linux 1.7.5 and higher
- Optimized `gp_toolkit.gp_size_of_table_disk`
- The `pxf_fdw` and `tkh_fdw` extensions are created in all databases when Tkhemali is complemented with ADB ClickHouse Connector

Bug fixes

- `gplogfilter` did not skip files if time filters were used
- Fixed the `extra data after last expected column` error with the `gpperfmon.queries_now_fast` table
- The coordinator did not cancel a PXF query on segments
- The `gp_fastsequence` index could be corrupted during data recovery
- ADB Control installation (via the ***Install*** service action) failed if ADB cluster had no the ***Monitoring clients*** service
- It was not possible to delete a role that had a `GRANT` on the PXF protocol
- `gprestore` failed if the `tkh_fdw extension` extension was installed
- ADB ClickHouse Connector failed when trying to insert data using `mpp_execute = any` with staging tables enabled

- Fixed the `gprestore` error when inserting data into `pg_statistic` for tables after partition exchange
- `diskquota` did not delete entries about expired temporary tables for interrupted sessions
- Metafiles were not fully restored in `adb_ddp_plugin`
- Fixed the error when inserting a partitioned table into a leaf partition
- Data inserting via ADB ClickHouse Connector resulted into `SIGSEGV`
- `gprestore --resize-cluster` failed when trying to restore `DISTRIBUTED REPLICATED` tables
- `gprestore --resize-cluster` failed when trying to restore empty tables
- Fixed a non-quoting `*.csv` filter in a log compression script

6.25.2

6.25.2.52

Date: 02.11.2023

New features

- Synced with upstream Greenplum Database 6.25.2
- Upgraded `gpbackup` to 1.29.4
- Added support for AstraLinux 1.7 SE "Orel" (for the Enterprise edition of ADB)
- Added support for Docker CE
- Added the option to choose custom JAVA_HOME for cluster
- Added the Parameter archiving feature — ability to export ADB settings according to the specified schedule
- Allowed PXF log output in the JSON format
- Added verification of reserved characters for database names
- Ensured correct ADB deployment in case of custom `umask`
- Added checking residual gpdb processes during the upgrade action
- Implemented ADBM 1.5.2
- Implemented ADB Control 4.5.3

Improvements

- Added calculation of the root partition size to `arenadata_toolkit.adb_skew_coefficients`
- Improved the ADB audit query
- `gpbackup`: excluded from a backup process setting all subpartitions when the `--leaf-partition-data` and `--exclude <root_partition>` options are used (to avoid pointing out every subpartition)
- Added a persistent volume for ClickHouse in ADB Control
- Reset sysctl parameters to recommended values

- Refactored closing connection to PXF, including fdw

Bug fixes

- Fixed: a `SELECT` query to `arenadata_toolkit.__db_files_current` failed
- Fixed: ADB to ADB Connector installation failed due to the lack of extension files on segments
- Fixed incorrect filling of the `submit` field in the `queries_history` table for `gpperfmon`
- Fixed: reboot timeout was too small to fit physical servers
- Fixed: a `cron` string was doubled for `addc_container_monitor`
- Fixed: CA cert had extra quotes in name
- Fixed: `diskquota` freezed in the paused state after upgrade
- Fixed ORCA invalid processing of nested SubLinks under the aggregates

6.25.1

6.25.1.51

Date: 31.08.2023

New features

- Implemented ADB to ADB Connector 1.0.0
- Added the ***Uninstall*** action for Madlib
- Implemented ADBM 1.4.0
- Implemented ADB Control 4.4.0

Improvements

- `gpbackup`: removed the timeout for `gprestore` that caused the connection error if `COPY` from gpdb didn't connect to the pipe
- Replaced `gp_log_system` for `gp_toolkit.__gp_log_master_ext` in the *pg_hba* validation query

Bug fixes

- Fixed ORCA invalid processing of nested SubLinks referenced in the `GROUP BY` clause
- Excluded reconfiguring of Diamond if a cluster has no monitoring
- Fixed the error with an empty value of the ADB Control Docker address in *pg_hba.conf*
- Fixed the error that occurred when the session on the master node was ended, while the sessions on the segments continued to work
- Removed dynamic linking of zstd for `gp_zstd_compression.so` to avoid a memory leak during workfiles compression
- Fixed: `VACUUM` made all segments inaccessible in case of a concurrent transaction
- Fixed Explicit Redistribute logic for queries with SubPlans

6.25.1.49

Date: 10.08.2023

New features

- Synced with upstream Greenplum Database 6.25.1
- Upgraded `pgbackrest` to 2.47
- Upgraded `gpbackup` to 1.29.1
- Upgraded `gpbackup_s3_plugin` to 1.10.0
- Upgraded `madlib` to 1.21.0
- Added LDAP integration and synchronization with ADB from bundle
- Added the ***Enable mirroring*** action to the ADB service
- Added the `pgbouncer-adb6` RPM with LDAP native authentication
- Added the ***Uninstall*** action for PostGIS
- Added the ***Uninstall*** action for `diskquota`
- `gpbackup`: added the `--report-dir` parameter to `gprestore`
- Implemented ADBM 1.3.3
- Implemented ADB Control 4.3.3

Improvements

- Timezone is taken into account in the *operation_log* filenames for `arenadata_toolkit`
- Added support for the `EXPLAIN ANALYZE` queries that contain cursor declarations
- Reduced memory consumption by avoiding ORCA initialization on query executors where ORCA is never used

Bug fixes

- Fixed `Unhandled ORCA memory pool init error` in OOM conditions
- `gpbackup`: table grants didn't appear after the restore in a new schema
- `gpbackup`: wrong DDL were created for foreign tables using `kadb_fdw`
- Fixed a segmentation fault during a dispatch interrupt
- Fixed update of ADB Control/ADBM versions in UI after cluster upgrade
- The `EXECUTOR` role for standby left only after process of obtaining standby from Registry
- Fixed inconsistency of `adcc-agents` with external ADB Control server endpoint

6.24.3

6.24.3.48

Date: 20.07.2023

New features

- Implemented ADBM 1.3.2
- Implemented ADB Control 4.3.2

Improvements

- Postgres config files are backed up for the purpose of ADBM Copy (Disaster Recovery)

Bug fixes

- Fixed: cluster was marked `Down` in ADBM when master was on the same host as segment

6.24.3.47

Date: 30.06.2023

New features

- Synced with upstream Greenplum Database 6.24.3
- Upgraded `pgbackrest` to 2.45
- Upgraded `gpbackup` to 1.28
- Upgraded `gpbackup_s3_plugin` to 1.9
- Upgraded PXF to 6.6
- ADB ClickHouse Connector went out of beta to GA
- ADB ClickHouse Connector: implemented segment scans to support select capability
- Implemented the `adb_collect_table_stats` function to handle a problem with NULL values in `db_files_current`
- Implemented the arenadata_toolkit.adb_skew_coefficients view to monitor data skews
- Added SASL_PLAINTEXT and SASL_SSL to ADB Kafka connector
- Implemented ADBM 1.3.0
- Implemented ADB Control 4.3.1

Improvements

- SSH connection timeout was increased to ensure spare node installation with ADB bundle
- ADB ClickHouse Connector: avoided using of short-living context for edatum in scans
- ADB ClickHouse Connector: avoided removing of one resource release callback inside another

- ADB ClickHouse Connector: changed the `send_delay_ms` option to `send_delay` in the `X-GP-OPTIONS-SEND_DELAY` header
- ADB ClickHouse Connector: improved initial shard selection based on `segno` for insert queries based on round-robin
- ADB bundle was changed to use rebranded service names of ADB ClickHouse Connector (ex — Tkhemali) and ADB Control (ex — ADCC)
- `adb_ddp_plugin` was optimized with more performant restores by means of filtered offsets
- Migrated the `arendata_toolkit` schema to the `arenadata_toolkit` extension module

Bug fixes

- Fixed the `gpstop` error `No such process` with a locale other than `en_US`
- Fixed `SEGFAULT` when executing a query using `JOIN LATERAL`
- Fixed `SEGFAULT` when executing a competing update in a trigger
- Fixed PXF migrator in an ADB bundle
- Fixed the FQDN problem that occurred during the ***Expand*** action
- Fixed: ORCA set improper Flow for Shared Scan nodes during translation to a physical plan
- ADB ClickHouse Connector: fixed the uninitialized variable error inside the options module
- Fixed creating the `diskquota` extension with the global option `appendonly=true`

6.23.5

6.23.5.45

Date: 31.05.2023

New features

- Synced with upstream Greenplum Database 6.23.5
- Upgraded `diskquota` to 2.2.1

Improvements

- Limited the find command depth when `diskquota` upgrading

Bug fixes

- Fixed `diskquota` 2.x to allow a huge number of rows in `diskquota.table_size`
- PXF curl error buffer: a transfer closed with external read data remaining
- Fixed: `diskquota` hung and blocked the cluster stopping

6.23.3

6.23.3.44

Date: 27.04.2023

New features

- Synced with upstream Greenplum Database 6.23.3
- Upgraded PXF to 6.5.0
- Upgraded `diskquota` to 2.2.0
- Implemented ADB ClickHouse Connector. You can use the new connector version in the beta mode simultaneously with the previous version Tkhemali Connector 1.X
- Started to process the `IN` predicate for filter push-down purposes in PXF JDBC
- Enabled SSL between client and master
- Enabled addition of Monitoring Clients to new hosts after the ***Expand*** action
- Added PXF Monitoring Grafana Dashboards
- Added the ability to expand monitoring agents in the ADB bundle
- Arenadata DB Command Center (ADBCC) is renamed to Arenadata DB Control (ADB Control)
- Implemented ADBM 1.2.1
- Implemented ADB Control 4.2.1

Improvements

- Refactored the logs compression script
- Added additional checks for the ADB state when the ***Install***/***Reinstall*** ADBM actions are being performed
- Removed redundant Partition Selector nodes from the `DELETE` operation when anti-join is used to filter for deletion

Bug fixes

- It was impossible to upgrade `diskquota` directly from 1.0 to 2.1 or later
- PXF: fixed the error that occurred if the year contained more than 4 digits
- ADCC monitoring could not be installed when "one shot" cluster installation was used
- Fixed the incorrect IP for standby in the *arenadata_segments_monitor.sh* script after the cluster upgrade
- The `gpcheckcat` test failed after syncing with 6.23
- ADCC client did not re-register on the ADCC server with LDAP when running ***Reconfigure & Restart***
- The PANIC error in logs with the `could not fsync file` text on MIRROR after moving a database to another tablespace
- Disabled the `gp_enable_gpperfmon` GUC if ADCC >= 4.1 is being installed/upgraded
- After restarting a master server or segment host, Grafana transmitted incorrect data
- Corrected the error output for `gpfdist` when inserting large rows
- Fixed issues with the external cluster DB addition into *pg_hba.conf* during the ***Reconfigure & Restart*** action

- It was impossible to uninstall the ADCC service due to the `Edit pg_hba` error
- Fixed issues with the *arenadata_segments_monitor.sh* script
- Fixed the missing (yet claimed) `tablefunc` extension of version 1.1
- The ADBM actual mapping state was not checked during the ***Install*** ADBM service action
- Fixed: monitoring scripts could exhaust a connection pool
- The connection cluster was not created automatically after the bundle installation with LDAP
- After the ***Expand*** action, the unusual data appeared on a new segment in *data_analyze*
- Fixed the incorrect `never executed` label in query plans
- Fixed: the ADB cluster update process failed
- Fixed the uninformative error that occurred when the reload process failed
- The high memory consumption could lead to OOM for AOCS tables and NLJ

6.22.1

6.22.1.41

Date: 15.02.2023

New features

- Upgraded `pgbouncer` to 1.18
- Upgraded `gpbackup` to 1.27
- Upgraded `plcontainer` to 2.2
- Implemented a buffer parameter in the `gpcheckperf` utility
- ADB bundle: added a checkbox to echo ADB-specific metrics to Monitoring Clients
- Implemented ADBM 1.1.0
- Implemented ADBCC 4.1.0

Improvements

- ADB bundle: removed the `gppefmon` dependency from ADBCC and refactored the ADBCC configs
- ADB bundle: renamed `Encryption Key` to `Alias` in the PXF service
- Added a precaution annotation to */home/gpadmin/arenadata_configs/postgresql.conf*
- Made migration of the `arenadata_toolkit` objects optional and unchecked by default

Bug fixes

- Components in ADB did not change their state to `installed` after installation
- High memory utilization for queries with SQL subqueries

- The service query *collect_table_stats.sql* periodically hung
- The PXF encryption error: `Encryption library was not found in /var/lib/pxf/lib/ directory`
- It was impossible to add the Chrony Secondary component on a new standby host
- PXF did not start with enabled option encryption: `Implement check for keystore file is exist`
- Refactored *arenadata_segments_monitor.sh* to avoid `crontab` for a standby node
- `EXPLAIN` provided a wrong rows count for a WindowAgg node
- The invalid byte sequence for encoding UTF8 in logs during accessing `arenadata_toolkit.db_files`
- GPORCA produced a bogus plan for queries with CTE during handling distribution for Sequence children
- The `SIGSEGV` error occurred when ADB read a PXF external table
- Incorrect `json` type processing
- The issue with changing an ADBCC service configuration file via ADCM
- Fixed a system database name for ADBM
- Fixed known issues for ADB 6.22.1.40

Misc/Internal

- Placed the `plcontainer` Docker images (python2/python3) in the Arenadata public registry

6.22.1.40

Date: 06.12.2022

New features

- Synced with upstream Greenplum Database 6.22.1
- Upgraded `gpbackup` to 1.26.0.7
- Upgraded PXF to 6.4.2.4
- Upgraded `kafka-adb` to 0.16
- Added `encryption` 1.0.0
- Added `adbm` 1.0.0
- Upgraded `geospatial` to 2.5.4.1
- ADB bundle: implemented the `pg_hba custom section` in the ADB service configuration
- ADB bundle: added the ADBM service and the ADBM configuration page
- ADB bundle: added an option to the ADB configuration to store a Docker certificate for a trust self-signed registry
- Obfuscated `jdbc.password` in the PXF *jdbc-site.xml* file
- Added the ability to deserialize AVRO messages in `kafka-adb`

Implemented ADBCC 4.0.0

Added a new system for ADB binary backups management — Arenadata DB Backup Manager (ADBM)

Improvements

Made optimizations in `arenadata_toolkit.db_files_history`

Bug fixes

The error occurred during the PostGIS extension upgrade in ADB 6.22.0

The `CREATE EXTENSION` command in explicit transactions failed during assertion on executors

Fixed the cross-slice interaction detection for subplans

The error with ***Precheck*** and ***Check*** actions after ADB installation

The ADBCC client did not re-register on the ADBCC server when ***Reconfigure & Restart*** was being applied

PXF might not catch a malformed chunked encoded response

The postmaster process could exhaust all signal slots under the intensive connections pressure

PXF hung indefinitely when it executed queries using the `maximumPoolSize = 1` parameter

Changed logic for ADB `checkstatus`

6.22.0

6.22.0.38

Date: 20.10.2022

New features

Synced with upstream Greenplum Database 6.22.0

Added support for AltLinux 8.4 SP. AltLinux 8.2 SP is now deprecated

Added Data Domain Boost 1.0.0 (`adb_ddp_plugin`)

Upgraded `gptkh` to 0.14

Upgraded `diskquota` to 2.0.1

Upgraded PXF to 6.4.2.3

PXF: allowed setting of Oracle parallel instructions

PXF: enabled a filter pushdown status in PXF JDBC logs

Implemented ADBCC 3.7.0

Bug fixes

PostgreSQL planner produced a bogus plan for queries against replicated tables with `SIRV` functions

The `SIGSEGV` error occurred when ADB read a PXF external table

Fixed the lost parameter `delete` in a `cron` log rotation script

Refactored Planchecker to use an external ADBCC database

The Postmaster process might exhaust all signal slots under the intensive connections pressure

`gpbackup`: fixed the metadata order so that now `gprestore` can restore functions after the tables that are used in functions as a returning type

Fixed the projection of boolean qualifiers which attributes are not in the `SELECT` list

PXF refactoring:

- Fixed reading of configuration files after the PXF cluster sync.
- Added the PXF *bin* to `PATH` of the `gpadmin` user.
- Installed PXF to standby.

Applying the `INSERT INTO` command to a replicated table in combination with the `SELECT FROM` query against another replicated table with sequence generation failed

`entrydb` processed wrong slot resetting on moving a query to a resource group

A failure occurred when DynamicSeqScan had a subPlan

Failed to exchange a partition with a table having a dropped column that the table was distributed by

The `SIGPIPE` error occurred during fetching the external table data via the extended protocol

gptkh: fixed fetching of the actual `system.tables` columns in ClickHouse (according to the ClickHouse version)

Misc/Internal

Added a registry URL and arenadata-enterprise repository to the bundle configuration

6.21.1

6.21.1.36

Date: 30.08.2022

New features

Synced with upstream Greenplum Database 6.21.1

Enabled core dump files for ADB processes

Implemented ADBCC 3.6.0

Bug fixes

Tkhemali could not process text fields with a new line character

Read data stream is released now when a partition scan is over

The `EvalPlanQual` routine erroneously arose for the `DELETE` operation over a partitioned append-optimized relation

`gp_toolkit.pg_resgroup_move_query` could lose RG slots

- `auto_explain` is not running now anywhere besides a dispatcher process
- `ALTER TABLE` erased `pg_appendonly` values
- Fixed a gang loss during running a transaction when `gp_dist_wait_status` was being called within a transaction
- A database process entry initialized a whole plan tree

6.21.0

6.21.0.35

Date: 21.07.2022

New features

- Synced with upstream Greenplum Database 6.21.0
- Implemented ADBCC 3.5.1

Improvements

- Optimized DML queries against partitioned tables to avoid further planning if a partition was pruned
- Excluded the `gpmon` background process from the shared memory user list
- Refactored the `gpdb` specific code for the dispatch result waiting
- Implemented a fallback to PostgreSQL for an empty target list in CTE producer

Bug fixes

- Fixed the cross-slice interaction detection for subplans
- Copying subplans for partitions by a planner broke the subplans order
- `gpinitsystem` failed on a machine with a custom locale
- `gptkh` could not parse engine settings with a dot symbol
- `kafka-adb`: re-added *kafka-client-2.5.0.jar* to the `kadb_fdw` connector

6.20.4

6.20.4.34

Date: 07.07.2022

New features

- Synced with upstream Greenplum Database 6.20.4
- ADB bundle: added the ability to specify a cluster network to distinguish a high-speed network for the interconnect purpose

Bug fixes

- ADBCC: the ***Container Memory Usage*** panel was blank under certain conditions

6.20.3

6.20.3.34

Date: 19.05.2022

New features

- Synced with upstream Greenplum Database 6.20.3

Bug fixes

- PXF: when the PXF service migrated, the rights to the directory were not being checked
- PXF: the `PXF_JVM_OPTS` parameters could not be changed
- The *gpexpand_schema.tar* archive was created in the directory */home/gpadmin* and could overflow the space
- A compatibility platform was incorrect for community bundles in AltLinux

6.20.1

6.20.1.33

Date: 11.04.2022

New features

- Synced with upstream Greenplum Database 6.20.1
- PXF: added PXF 6.3.0 to the ADB bundle (with ability to upgrade from PXF 5.x)
- PXF: activated a PXF cluster sync command
- PXF: added the PXF *bin* directory to the `PATH` variable
- PXF: added the ability to override data types mapping in external tables for PXF
- Added Tkhemali Connector 1.0.1 with a lot of refactoring
- Added the ***Delete Tkhemali*** action
- Added the ***Delete Kafka-to-ADB*** action
- Added the ***Delete ADB-to-Kafka*** action
- Added the `-Dliquibase.releaseLocks=true` parameter to the migration container for the ***Reconfigure & Restart*** action
- Implemented ADBCC 3.4.0

Improvements

- Made editing of */etc/hosts* optional

Bug fixes

- PXF: error with JDBC Hive when one of the fields was a reserved word
- GPORCA failed on assert during transformation `LEFT JOIN` to `INNER JOIN`
- The aggregate mode did not handle properly query finish commands from QD (Query Dispatcher)
- `kafka-adb`: the storage option for an offset table is set explicitly now to avoid overriding
- ADBCC installation failed if a cluster name consisted of more than one word
- Interleaving of the append-optimized partitioned table removal and the `pg_total_relation_size` call for one of its partitions potentially caused `SIGSERV`
- gptkh: fixed the `tnx()` function error that occurred when the data was being inserted into a partitioned table with the partitions number more than 1
- Tkhemali: column names with spaces were not handled properly
- GPORCA produced invalid plans for a common table expression (CTE) with replicated tables
- GPORCA: the `Gather motion` merging is enabled now for singleton on a segment
- Monitor scripts were not added to `crontab` on the ***Activate Standby*** action
- Backend server was displayed as IP while agent was displayed as FQDN on the ***Version*** page

6.19.3

6.19.3.32

Date: 09.03.2022

New features

- Synced with upstream Greenplum Database 6.19.3
- Implemented ADBCC 3.3.1

6.19.1

6.19.1.31

Date: 17.02.2022

New features

- Synced with upstream Greenplum Database 6.19.1
- Hid a server name in the HTTPS group ADBCC service
- Added ADB Loader tools for RHEL 8
- Added the ability to deploy maintenance scripts for several databases
- Enabled the `backlog_lock_waits` GUC
- Added the ***Delete PXF*** action

Implemented ADBCC 3.3.0

Improvements

Moved the ***Disable firewall*** option to the cluster level from the ADB service

Bug fixes

Wrong column binding was used to extract values from a SharedScan node

gptkh: the distributed table `test2_tmp_3054` looked at itself

gptkh: an error in parsing settings for distributed tables

Coordinator did not use quote escaping for GUC values when restored on QE

The sshd option `MaxStartups` had to be added before `Match`

Misc/Internal

Built the docker-compose RPM package 1.29.2

6.18.2

6.18.2.30

Date: 20.12.2021

New features

Synced with upstream Greenplum Database 6.18.2

[6X backport] Implemented `archive_mode` always

[6X backport] Zero fill pages skipped by a force WAL switch

Added Kafka ADB connector 1.0.4

Added `plcontainer` 2.1.5

Implemented ADBCC 3.2.5

Bug fixes

Power: an unexpected subplans order caused `subselect_gp` test fails

6.18.0

6.18.0.29

Date: 18.11.2021

New features

- Implemented ADBCC 3.2.4

Bug fixes

- When accessing from an append-optimized table, Index returned the wrong result
- kafka-fdw: fixed garbage in output at least for the text format
- `gpbackup` handled an empty tables set for an incremental data backup properly
- `gpbackup` hung forever after it was interrupted

6.18.0.28

Date: 12.11.2021

New features

- Synced with upstream Greenplum Database 6.18.0
- `gpbackup`: added an explicit order of tables by using `pg_class.relpages`
- PXF: added the partitioning query support for Sybase
- Implemented ADBCC 3.2.2

Bug fixes

- The `SIGSEGV` unexpected error on Reindex operation
- `madlib` on ppc64le arch is now included into a build
- `gpbackup` could dump the partitioned table data twice
- The `Not enough memory reserved for the statement` error occurred when the PL/pgSQL function was called multiple times for a common table expression (CTE) `SELECT` statement against a table with a lot of partitions

6.17.5

6.17.5.26

Date: 30.09.2021

New features

- Synced with upstream Greenplum Database 6.17.5
- ADB bundle: added the ***Delete*** service action for Monitoring Clients
- ADB bundle: added the `Ready to upgrade` status for a bundle upgrade action

Bug fixes

- Fixed low CPU performance on Power with newly added `CGLAGS` build options

6.17.2

6.17.2.26

Date: 07.09.2021

New features

- Synced with upstream Greenplum Database 6.17.2
- Synced with PXF 5.16.3
- Synced with `gpbackup` 1.21.0
- Synced with `gpbackup_s3_plugin` 1.7.0
- Added a build for Power8 LE platform (ppc64le arch)
- Implemented ADBCC 3.1.3

6.17.1

6.17.1.25

Date: 17.08.2021

New features

- Synced with upstream Greenplum Database 6.17.1
- Implemented the ZSTD compression support for `gpbackup`
- `adcc-extension`: started to send a schema name in the Node message
- ADB bundle: external database connection for ADBCC
- Implemented ADBCC 3.1.0

Bug fixes

- kafka-fdw: segments failed after applying `SELECT` to a KafkaToADB external table with batch size more than `40000 - <msg_count> * 40` bytes (where `<msg_count>` is a count of messages)
- PostgreSQL query optimizer built a bad plan for replicated tables with indexes
- `adcc-extension`: the expected segment distributed transaction context was `Segment Prepared`
- `adcc-extension`: there was no possibility to differ QueryStatus for slices executed on master
- Power: a bogus compiler error during `pg_upgrade` compilation

CVE

- CVE-2020-25695

6.16.3

6.16.3.24

Date: 03.08.2021

New features

- Synced with upstream Greenplum Database 6.16.3
- [6X backport] `POLLHUP` / `POLLRDHUP` are now detected while queries are running
- `adcc-extension`: started to retrieve and send an error text
- ADB bundle: defined adjustable parameters for ADBCC
- Implemented ADBCC 3.0.3

Bug fixes

- Fixed `ERROR: header checksum does not match`
- [6X backport] Restricted execution for multi-level correlated queries
- `adcc-extension`: `security_barrier` views

6.16.2

6.16.2.21

Date: 24.06.2021

New features

- Synced with upstream Greenplum Database 6.16.2
- [6X Backport] Shrinking of relation segment files to zero on `TRUNCATE` and `DELETE`
- ADBCC: added JVM arguments for logging

Improvements

- PXF: removed the tuple count check for JDBC queries `INSERT`

Bug fixes

- `gpbackup` failed when foreign tables were present in the database since it tried to acquire a lock on a foreign table

6.16.1

6.16.1.20

Date: 03.06.2021

New features

- Synced with upstream Greenplum Database 6.16.1
- Implemented ADBCC 2.1.1
- Kafka ADB Connector: allowed users to set custom `librdkafka` options
- [6X backport] GPORCA fails with `SIGSEGV` on queries to view against a table with dropped columns
- [6X backport] The `elog` function ignores multibyte encoding when it truncates a long message

Improvements

- ADB bundle: set a netcat timeout for sending monitoring metrics in AltLinux

Bug fixes

- The PartitionSelector mechanism did not return tuples in case of the type mismatch
- `adcc-extension`: unknown node with `id=5` for a subplan in values
- `adcc-extension`: a locally planned mark was lost in node status messages for queries with custom aggregate
- Wrong CCNTs for plans dispatched after InitPlans were executed
- Removed redundant `Gpmon_Incr_Rows_Out()` calls in internal sort functions
- An invalid variable specified in the segment directory path check in ADCM

6.16.0

6.16.0.19

Date: 17.05.2021

New features

- Synced with upstream Greenplum Database 6.16.0
- Added the `gp_enable_gpperfmon=on` parameter to master and segment servers for ADBCC

Improvements

- Implemented the `diskquota` extension update

6.15.0

6.15.0.18

Date: 28.04.2021

New features

- Added the `PXF_CONF` and the `PXF_HOME` environment variables to PXF hosts
- Kafka ADB connector: implemented signal handlers to interrupt consuming

Ported ADB to Alt Linux 8.2

Improvements

[6X backport PR 11625] pg_rewind: avoided removing files from the *log* directory

Bug fixes

Invalid values in the `prodataaccess` attribute of the `pg_proc` relation after the cluster initialization/upgrade

Kafka ADB connector did not consider partitions empty when a query for their watermark offsets failed after a timeout

[6X backport] GPORCA produced an incorrect plan for the `NOT IN` clause

6.15.0.17

Date: 13.04.2021

New features

Implemented ADBCC 2.0

Bug fixes

A broken URL in the ADB bundle service description

6.15.0.16

Date: 24.03.2021

New features

Synced with upstream Greenplum Database 6.15.0

Implemented switchover from master to standby via ADCM

Bug fixes

Kafka ADB connector: converting AVRO dates before UNIX epoch to the Greenplum Database representation

Kafka ADB connector: the Greenplum Database representation of AVRO double numbers that have more than 6 digits after a radix point

Kafka ADB connector: wrong interpretation for AVRO timestamps below zero

6.14.1

6.14.1.14

Date: 03.03.2021

New features

Synced with upstream Greenplum Database 6.14.1

Added a link to the home ADBCC page, which is activated when a logo is clicked

Improvements

Removed online loading of static resources from the ADBCC web interface

Bug fixes

- Resource group waiting queue corruption
- The `pg_aocsseg` inconsistency after the column addition rollback
- A cluster failed after the single segment host lost due to the FTS misbehavior
- A multiline text in the CSV output from `gplogfilter` was incorrect

6.14.0

6.14.0.13.b2

Date: 24.02.2021

Bug fixes

Fixed incorrect links in repositories of the 6.14.0.13.b1 release

6.14.0.13.b1

Date: 15.02.2021

New features

Synced with upstream Greenplum Database 6.14.0

6.13.0

6.13.0.12

Date: 21.01.2021

New features

- Synced with upstream Greenplum Database 6.13.0
- Supported auxiliary relations for append-optimized tables to be used by the `pgstattuple` extension to exactly estimate the bloat of those relations

Improvements

- Tkhemali-connector: avoided the intermediate conversion to a byte array
- Removed the obsolete batching options in ADQM connector (since the `TEXT` is used now)
- Reduced logging of `gpstate`

Bug fixes

- Greenplum Database failed after the developer GUCs of PostgreSQL being activated
- GPORCA optimizer built inefficient query plans
- Column-wise statistics target parameters did not apply to partitioning tables
- Error with `gpactivatestandby` when running it too fast after the master power off
- Error in parsing decimal parts for time types
- Scaling applied incorrectly to decimal values with a length less than a precision

Misc/Internal

- Bumped to the latest version of additional components

6.12.1

6.12.1.11

Date: 21.12.2020

New features

- Synced with upstream Greenplum Database 6.12.1
- Added the offset function to Kafka connector
- Added the rest committed function to Kafka ADB connector
- Implemented the text format for Kafka ADB connector
- Provided the `JVM_OPTS` setting for PXF
- Supported AVRO logical types in Kafka ADB connector
- Switched to the new way of PostGIS installation
- Started to use `rd_kafka_query_watermark_offsets` to validate partition-offset pairs in Kafka ADB connector

Bug fixes

- ADBCC disk space usage troubles in case of core dumps
- The `missing data for column` error in `gprestore`
- The `limit of distributed transactions has been reached` error on standby master
- The `WindowFunc with winref 1 assigned to WindowAgg with winref 2 (nodeWindowAgg.c:2264)` error
- Wrong behavior with a grouping function
- An exclusive lock on any partitioned table locked the `pg_partitions` view
- A server locked the `kadb.offsets` table

ADB 6 known issues

Daria Barysheva

To Table of Contents

Issue	From	Fixed	Workaround
RED OS repositories contain the `pgbackrest` package, which conflicts with `pgbackrest` supplied by Arenadata repositories with ADBM	—	—	Do not install or update `pgbackrest` supplied by RED OS repositories. The package is installed and updated during ADBM installation and upgrades
In case of requests that have a slice executing on the coordinator, spill file entries in `gp_toolkit.gp_workfile_entries` and `gp_toolkit.gp_workfile_usage_per_query` might be duplicated (with different PIDs), resulting in incorrect total size calculation	—	—	Use `SELECT DISTINCT ON (segid, sess_id, command_cnt, prefix)` in queries against the mentioned views to remove duplicate entries. For example:

- Get information about all spill files:

```
SELECT DISTINCT ON (segid, sess_id, command_cnt, prefix)
    pid, sess_id, segid, command_cnt, size, numfiles, prefix
    FROM gp_toolkit.gp_workfile_entries
    ORDER BY segid, sess_id, command_cnt, prefix, pid;
```

- Get total space usage for different sessions and segments:

```
SELECT sess_id, segid, SUM(size) AS total_size, SUM(numfiles) AS total_files FROM
(
    SELECT DISTINCT ON (segid, sess_id, command_cnt, prefix)
    segid, size, numfiles
    FROM gp_toolkit.gp_workfile_entries
) t
GROUP BY segid, sess_id ORDER BY segid, sess_id;
```

There is an error in the description of the `CPUSET` option of the `ALTER RESOURCE GROUP` command in the VMWare documentation. Actually, the `segment_cores` value comes first, and then the `master_cores` value	—	—	Use the correct description provided in the first table column
For the `timestamp with time zone` type, PXF does not always perform push-down	6.27.1.57	—	To ensure the correct handling of the `timestamp with time zone` type, an external table definition must include the `date_wide_range=true` parameter
Some `ulimit` parameters (e.g. `nofile limit`) may be not set when a cluster is started via the `gpstart` console utility under the `gpadmin` user (occurs in Astra Linux)	6.25.2.52.b1	—	When switching to `gpadmin`, do not use `sudo -iu gpadmin`. Use `sudo su` instead: `$ sudo su - gpadmin`
If you made a backup via `gpbackup` for ADB cluster of a version preceding 6.25.2.52.b1 and try to restore a database from that backup in ADB 6.25.2.52.b1 via `gprestore`, you can get the following error: `[CRITICAL]:-ERROR: language "plpythonu" already exists (SQLSTATE 42710)`	6.25.2.52.b1	—	Drop `PROCEDURAL LANGUAGE plpythonu` in a target database before running `gprestore`
If you made a backup via `gpbackup` for ADB cluster of a version preceding 6.25.2.52.b1 and try to restore a database from that backup in ADB 6.25.2.52.b1 via `gprestore`, you can get the following error: `[CRITICAL]:-ERROR: function public.dblink_connect_no_auth(text) does not exist (SQLSTATE 42883)`	6.25.2.52.b1	—	1. Run `gprestore` with the `--metadata-only` and `--on-error-continue` options. 2. Check that there are not any critical errors in *error log* (if errors are present, add the necessary relations manually). 3. Run `gprestore` with the `--data-only` option to restore data.

If you made a backup via `gpbackup` for ADB cluster of a version preceding 6.25.2.52.b1 and try to restore a database from that backup in ADB 6.25.2.52.b1 via `gprestore`, you can get the following errors:
`[CRITICAL]:-ERROR: incompatible library "/usr/lib/gpdb/lib/postgresql/gptkh.so": version mismatch (dfmgr.c:367) (SQLSTATE XX000)`
or
`[CRITICAL]:-ERROR: required extension "pxf_fdw" is not installed (SQLSTATE 42704)`
	6.25.2.52.b1	—	Update the extension packages `gptkh` and `pxf_fdw` in each database that uses these extensions and then update the extensions.
			To restore a backup, you only need to update the packages

The `SELECT` query to `arenadata_toolkit.__db_files_current` fails with the following error:
`psql:/tmp/tmpLWiv6c:1:`
`ERROR:  illegal rescan of motion node: invalid plan (nodeMotion.c:1712)  (seg105 slice5 10.183.104.153:10003 pid=22864) (nodeMotion.c:1712)`
`HINT:  Likely caused by bad NL-join, try setting enable_nestloop to off`
	6.25.1.49	—	1. Via `psql`, connect to the database where you gather statistics (e.g. `adb`) under a superuser:
			`$ psql adb -U <superuser>`
			2. Run the following command to alter the function:
			`ALTER FUNCTION arenadata_toolkit.adb_get_relfilenodes(tablespace_oid OID) ROWS 30000000;`

The ADB bundle update action can fail with the following error: `Failed: invalid syntax, for details look pg_hba.conf on master host`	6.24.3.48	—	1. Backup logs from the *pg_log* directory. 2. Clear or compress log files. 3. Restart ADB.

After updating ADB to version 6.24.3.48.b1 or higher, you can get the following error when using `gpbackup` / `gprestore`:
`[CRITICAL]:-table backups has no column named segment_count`
The error occurs due to the fact that in `gpbackup` of version 1.28.1 the `segment_count` field was added to the `backups` table of the service SQLite database `backup_history.db`.
This SQLite database is used to:
- Find a previous completed backup with the same options to create an incremental backup if the `--from-timestamp` parameter is omitted.
- During the restore process, find a plugin version (e.g. `gpbackup_s3_plugin`) that was used to create the specified backup.
Another side effect of the column absence is the incorrect value for a number of segments in backups that were already made. It leads to an error during a resize restore operation
	6.24.3.48.b1	—	Use one of the following options (each one comes with certain limitations):
			- **Option 1**. Use the `--from-timestamp` and `--no-history` `gpbackup` parameters together for incremental backups. This option is not helpful when a plugin is used.
			- **Option 2**. Drop the old history database file *backup_history.db* after upgrade to `gpbackup` of version 1.28.1 or higher:
			`$ rm $MASTER_DATA_DIR/gpseg-1/gpbackup_history.db`
			You have to proceed with full backups to restore data from incremental backups and provide the backup plugin work.
			- **Option 3**. Alter the `backups` table schema in the `backup_history.db` SQLite database:
			`$ sqlite3 $MASTER_DATA_DIR/gpseg-1/gpbackup_history.db`
			`ALTER TABLE backups ADD COLUMN segment_count INT;`
			`pragma table_info('backups');`
			You have to provide a valid number of segments if the resize restore operation is requested.

When data is selected from ClickHouse via ADB ClickHouse Connector, the following error occurs if the password for the `default` user is defined: `DB::Exception: default: Authentication failed: password is incorrect, or there is no user with such name.`	6.24.3.47	—	Do not use a password for the `default` user
Due to technical limitations in the ADB DDBoost plugin, the `restore_subset` plugin option is automatically set to `off` when the `on-error-continue` option is specified	6.22.0.38	—	Do not use the `on-error-continue` option during data restore
Expanding a cluster may fail if fully qualified domain names (FQDN) are used for the cluster hosts and the ***Check array*** flag is set	6.16.1.20	—	Unset the ***Check array*** flag during the cluster expansion process if FQDNs are used in the cluster
When resource management based on resource groups is active, the memory allotted to a segment host is equally shared by active primary segments. Assigning of memory to primary segments happens when the segment takes the primary role. The initial memory allotment to a primary segment is calculated on starting a cluster (with `gpstart`) and does not change, even when number of primary segments increases during the failover situation. This may result in a segment host utilizing more memory than the `gp_resource_group_memory_limit` setting permits. For more information, refer to the VMWare documentation	6.12.1.11	—	Restart your cluster to recalculate memory allotment in case of increased number of primary segments on hosts (e.g. after failover)
There is an error in the `COPY` command description in the VMWare documentation. The `FILL MISSING FIELDS` option should be replaced with `FILL_MISSING_FIELDS`	6.12.1.11	—	Use the correct option provided in the first table column
ADB Monitoring may raise an issue for metrics storage if RAID is used with the following error in */var/log/diamond/diamond.log*: `error [MainThread] ATTRIBUTE_NAME not found in second column of smartctl output between lines 4 and 9`	6.12.1.11	—	Set */etc/diamond/collectors/SmartCollector.conf* parameter `enabled = False`
`diskquota` may lose table statistics when being paused during ADB cluster restart. As a result, quotas may be violated. `diskquota` calculates table sizes and stores that information in the `diskquota.table_size` table periodically with a pause equal to `diskquota.naptime` (2 seconds by default). If you restart the ADB cluster during this interval of time, then `diskquota` will lose all changes that have occurred since the last data save to the `diskquota.table_size` table. For example, you could make a large insert into the tables, restart the cluster, and `diskquota` would not know about such a large insert. This happens because the `diskquota.table_size` table does not store information about data insert. Until these tables are active again, `diskquota` will not update their size. It can be useful to reinitialize `diskquota` by calling the `diskquota.init_table_size_table` function, but on large clusters this can take a significant amount of time	6.12.1.11	—	No workaround. There is no way to reliably persist in-memory changes from coordinator and segments when processes get a termination signal
The `diskquota.pause()` function description in the VMWare documentation is not correct. The right one is: `diskquota.pause()` makes bgworkers skip refreshing quotas entirely to avoid issues with wasting computation resources. Table sizes are updated correctly after resume	6.12.1.11	—	Use the correct description provided in the first table column
The formula for specifying the maximum number of table segments from the VMWare documentation is not correct. The right one is: `diskquota.max_table_segments = <maximum_number_of_tables> * ((<number_of_segments> mod 100) + 1) * 100`	6.12.1.11	—	Use the correct formula provided in the first table column
When trying to insert a row into a leaf partition of a partitioned table, you get the following error: `SQL Error [23514]: ERROR: trying to insert row into wrong partition` The issue reveals itself when some columns are dropped from a partitioned table and then partitions are added via `ADD PARTITION` or `EXCHANGE PARTITION`	6.12.1.11	—	Recreate a table or, if possible, do not drop columns from a partitioned table if you are going to add new partitions after that
In CentOS and Astra Linux, you cannot install or update ADB to 6.27.1.58 without manual installation of Java 17	6.27.1.58	6.27.1.58	Before ADB installation/update, install Java 17 on cluster hosts and fill in the ***JAVA_HOME*** parameter on the cluster configuration page
ADB to ADB Connector installation fails due to the lack of extension files on segments	6.25.1.51.b1	6.25.2.52.b1	Install the *adb-fdw* package for all ADB segment nodes manually, e.g. install it en masse via running `gpssh` from master or run installation on each node

During installation of the ADB cluster (version < 6.23.3.44) with external ADB Control (version from 6.23.3.44 to 6.25.1.49), you get inconsistency of agents with server endpoint and the failed bundle task:
`AssertionError: Action Install finished execution with unexpected result - 'failed'. Expected - 'success'`
`  TASK [adcc_client : Add ADB connection to ADCC]`
	6.23.3.44	6.25.1.49	No workaround. Only upgrade to the ***Fixed*** version resolves the issue

In ADB 6 starting with 6.22.1.41, Madlib uses a new path for its libraries — */usr/local/madlib-adb6*. After upgrading from previous ADB versions, Madlib stops working	6.22.1.41	N/A	1. Backup the `madlib` schema if necessary. 2. Drop the `madlib` schema in the cascade mode. 3. Run the ***Install Madlib*** action for ADB service via ADCM. 4. Restore schema objects from a backup.

The shared library *diskquota-2.0.so* doesn't appear in `shared_preload_libraries` during upgrade with the following *pg_log* error on master: `FATAL: could not access file ""diskquota"": no such file or directory`	6.22.1.41	6.24.3.48	1. Ensure that the dry-run of the `sed` utility works correctly:
			`$ sed 's/diskquota/diskquota-2.0/g' /data1/*/gpseg*/postgresql.conf`
			2. Run the `sed` utility with the `-i` parameter to apply changes:
			`$ sed -i 's/diskquota/diskquota-2.0/g' /data1/*/gpseg*/postgresql.conf`

During installation of the *encryption* module, you get the following error: `ERROR: Encryption library was not found in /var/lib/pxf/lib/ directory`	6.22.1.40	6.22.1.41	Copy the necessary library from */usr/lib/pxf/lib/*: `$ cp /usr/lib/pxf/lib/encryption-1.0.0-exec.jar /var/lib/pxf/lib/encryption-1.0.0-exec.jar`

If you upgrade your cluster to 6.22.1.40, remember that during this process the `arenadata_toolkit.db_files_history` table is being recreated with loading partitions/compression options and all data into the new table. This process may take a long time for a huge table	6.22.1.40	6.22.1.41	If you do not want to include the `arenadata_toolkit.db_files_history` table migration into the upgrade process, rename this table before upgrade and return the source name after:
			1. Run the query:
			`ALTER TABLE arenadata_toolkit.db_files_history RENAME TO db_files_history_skip;`
			2. Run the ***Upgrade*** cluster action via ADCM and wait for a successful upgrade.
			3. Run the query:
			`ALTER TABLE arenadata_toolkit.db_files_history_skip RENAME TO db_files_history;`

If you upgrade your cluster to 6.22.1.40 with installed ADCC and the ***UI LDAP authentication*** parameter switched on, the newer ADB server is not registered in ADCC and its version on the ***Information*** page in ADCC remains the same as for the older ADB server. It does not affect any functionality	6.22.1.40	6.22.1.41	1. Uninstall ADCC before upgrading the ADB cluster to 6.22.1.40. 2. Run ***Upgrade***. 3. Reinstall ADCC after upgrade.
[6X issue 13067] Gradual memory leak on mirror segments	6.19.1.31	6.19.3.32	Increase monitoring for memory consumption in a cluster and restart a cluster during the maintenance if possible. This issue will be patched in the next release
`madlib` on Power8 (ppc64le arch) breaks a `gpdb` build	6.17.2.26	6.17.5.26	No workaround. `madlib` is not included into a Power8 (ppc64le arch) build
kafka-fdw of versions 0.11-0.12 (they were ported in 6.15.0.17) does not work correctly when a batch size exceeds `40000 - <msg_count> * 40` bytes (where `<msg_count>` is a count of messages). In this case, the `SELECT` operation from the KafkaToADB external table causes `SEGFAULT`. The fix on the issue is expected in the next ADB release	6.16.3.24	6.17.1.25	Do not use a batch size larger than `40000 - <msg_count> * 40` bytes if you read more than one message. There are no limits for reading a single message
The `gpbackup` utility of versions 1.20.1 — 1.20.4 does not work correctly when foreign tables are present in a database. Backup fails with the `SQLSTATE 42809` error. The fix on the issue is expected in the next ADB release	6.16.1.20	6.16.2.21	Replace the current `gpbackup` with version 1.20.0 which is included in ADB 6.13.0.12
Attempt to upgrade a cluster with `diskquota` enabled and `gp_default_storage_options='appendoptimized=true'` results in the following error: `ERROR: "append-only tables do not support unique indexes"`	6.12.1.11	6.25.1.49	1. Set `gp_default_storage_options='appendoptimized=false'` before upgrade. 2. Upgrade a cluster. 3. Revert to `gp_default_storage_options='appendoptimized=true'`.
The ***Upgrade*** action fails if `appendoptimized` is set to `true` in the `gp_default_storage_options` parameter for any database	6.12.1.11	6.24.3.48.b1	Revert the `gp_default_storage_options` value to `appendoptimized=false` before the upgrade and set it to `true` after
`gpperfmon` can flood with logs to *$MASTER_DATA_DIR/gpseg-1/gpperfmon/q*.txt* even if it is not in use	6.12.1.11	6.23.5.45.b1	1. Switch off the `gp_enable_gpperfmon` GUC: `$ gpconfig -c gp_enable_gpperfmon -v off` 2. Purge the *$MASTER_DATA_DIR/gpseg-1/gpperfmon/* directory.

When you expand ADB by running the ***Expand*** cluster action, you can get errors if cluster hosts were registered with full qualified domain names (FQDN). In that case, the `hostname` and `address` columns of the `gp_segment_configuration` table contain different values, which may cause errors during the ***Expand*** action	6.12.1.11	6.23.3.44	Before you expand an ADB cluster:
			1. Stop ADB, for example, via the following command:
			`$ gpstop -M fast`
			2. Start a master segment in the utility mode:
			`$ gpstart -m`
			3. Set the `allow_system_table_mods` GUC value to `true`:
			`$ psql adb`
			`SET allow_system_table_mods = 'true';`
			4. Fix `gp_segment_configuration` so that the `hostname` and `address` column values are the same:
			`UPDATE gp_segment_configuration SET hostname = address;`
			5. Stop and start ADB in a standard mode:
			`$ gpstop -m`
			`$ gpstart`

ADB 5 releases

Daria Barysheva

Contents

To Table of Contents

5.29.11
5.29.8
5.29.6
5.29.5
5.29.4
5.29.3
5.29.2
5.29.1
5.29.0
5.28.14
5.28.13
5.28.11
5.28.9
5.28.7
5.28.6

5.29.11

5.29.11.19

Date: 16.06.2023

New features

Synced with upstream Greenplum Database 5.29.11

Improvements

Upgraded `tkh-connector` and `gptkh` to the latest PXF 5 versions

Bug fixes

Fixed `FATAL: could not open relation` after the segments switchover

Fixed the `gptkh` extension creation when extension scripts dropped anything

Fixed `ERROR: gp_relation_node has more entries than pg_aocsseg for relations`

5.29.8

5.29.8.18

Date: 13.10.2022

New features

Synced with upstream Greenplum Database 5.29.8

Improvements

Removed relcache in tablespaces located in non-default locations

5.29.6

5.29.6.18

Date: 15.06.2022

New features

Synced with upstream Greenplum Database 5.29.6

5.29.5

5.29.5.18

Date: 18.04.2022

New features

Synced with upstream Greenplum Database 5.29.5

5.29.4

5.29.4.18

Date: 03.03.2022

New features

Synced with upstream Greenplum Database 5.29.4

5.29.3

5.29.3.18

Date: 24.01.2022

New features

Synced with upstream Greenplum Database 5.29.3

Bug fixes

The `starve_case isolation2` test failed with ORCA

5.29.2

5.29.2.17

Date: 22.12.2021

New features

Synced with upstream Greenplum Database 5.29.2

5.29.1

5.29.1.17

Date: 26.11.2021

New features

Synced with upstream Greenplum Database 5.29.1

Updated `gpbackup` to 1.22.0_arenadata5

5.29.0

5.29.0.16

Date: 03.11.2021

New features

Synced with upstream Greenplum Database 5.29.0

5.28.14

5.28.14.16

Date: 30.08.2021

New features

Synced with upstream Greenplum Database 5.28.14

Updated `pgbouncer` to 1.16.0

Updated PXF to 5.16.3_arenadata2

Updated `gpbackup` to 1.21.0_arenadata3

Updated `gpbackup_s3_plugin` to 1.7.0_arenadata1

Bug fixes

Restricted execution for multi-level correlated queries

5.28.13

5.28.13.15

New features

Synced with upstream Greenplum Database 5.28.13

5.28.11

5.28.11.15

New features

Synced with upstream Greenplum Database 5.28.11

Bug fixes

Index scan in correlated subqueries produced incorrect results

5.28.9

5.28.9.14

New features

Synced with upstream Greenplum Database 5.28.9

Bug fixes

[5X backport] The *elog* function ignored multi-byte encoding when truncating long messages

[5X backport] The PartitionSelector mechanism did not return tuples in case of the type mismatch

5.28.7

5.28.7.13

New features

Synced with upstream Greenplum Database 5.28.7

Bug fixes

Fixed a broken URL in the ADB bundle service description

[5X backport] ORCA produced an incorrect plan for the `NOT IN` clause

5.28.6

5.28.6.11

New features

Synced with upstream Greenplum Database 5.28.6

Bug fixes

Fixed the `gpexpand` execution error that occurred for tables with dots in their names

Fixed the waiting queue corruption for resource groups

Fixed `SEGFAULT` that occurred during the table redistribution phase of the `gpexpand` operation

ADB Control releases

Daria Barysheva

Contents

To Table of Contents

- 4.15.1
- 4.15.0
- 4.14.2
- 4.13.3
- 4.13.0
- 4.12.0
- 4.11.0
- 4.10.3
- 4.9.1
- 4.8.8
- 4.7.5
- 4.6.4
- 4.5.3
- 4.4.0
- 4.3.3
- 4.3.2
- 4.3.1
- 4.2.1
- 4.1.0
- 4.0.0
- 3.7.0
- 3.6.0
- 3.5.1
- 3.4.0
- 3.3.1
- 3.3.0
- 3.2.5
- 3.2.4
- 3.2.2
- 3.1.3
- 3.1.0
- 3.0.3

4.15.1

Date: 23.12.2025

Bug fixes

Fixed SQL ID generation to use the `BIGINT` data type instead of `INTEGER`

4.15.0

Date: 28.11.2025

New features

ADB Control now supports ADB 7

Improvements

Added sorting by the ***Times accessed*** and ***Last accessed*** parameters in ***Audit → ADB Control → Relations***

Bug fixes

Resolved an issue with escaping characters in ***Users → Access***

Fixed overlapping symbols on the ***Dashboard*** page

Corrected system metrics calculations

Fixed plan nodes color and display issues

4.14.2

Date: 17.09.2025

New features

Implemented query grouping for representation of queries with the execution time shorter than the system metrics collection threshold

Improvements

Optimized SQL for the Data cleanup job

Bug fixes

Fixed incorrect display of the empty list of modified parameters on the Audit page

Fixed filtering tables on slow internet connection

4.13.3

Date: 05.08.2025

New features

Added processing and keeping in Vault the following data:

- ADBM user credentials.
- Truststore password (for this, you need to set ***Use PEM CA for vault*** to `true` in the ***Vault integration*** section of the ADB cluster configuration).

Improvements

Added the Hazelcast configuration to advanced bundle settings (see ADB Control → Advanced parameters → ADB Control agent hazelcast properties) to improve a Hazelcast cluster stability, which influenced the `adcc` agent. This configuration also allows you to manage the case of using ADB Control on ADB cluster hosts with two or more networks set up

Added the usage of `spring.config.additional-location` to avoid errors during upgrade on clusters with customized advanced parameters

Bug fixes

Fixed the error that occurred during the query plan text decompression

Fixed: a JKS password with specific parameters could prevent the ADB Control start

4.13.0

Date: 23.06.2025

Improvements

Set the resource group queuing timeout (`gp_resource_group_queuing_timeout`) for the `adcc` user connections

Added auto refresh of LDAP group mappings on ADB Control start to avoid authentication errors for LDAP users

Implemented the memory accounting context cleanup to avoid the `Out of memory` error on the `SELECT * FROM adcc_spills()` command execution

Implemented working with ClickHouse in high availability (HA) mode

Implemented separate truststore and keystore in ADB Control

Bug fixes

LDAPS users could not log in to ADB Control if Vault was used in ADB

Misc/Internal

In H2 2025, ADB Control and ADBM will be released in a separate bundle and will require new setup. Upgrade from the previous ADB bundles will not be available

4.12.0

Date: 07.03.2025

New features

Expanded the spill metrics set:

- Added ***Absolute total spill files at cluster*** and ***Skew of current spill value between segments*** (see ***Spill files*** in Monitoring of commands → System metrics collected for commands).
- Added the spill metrics collection mode selection (see General settings → Metrics).

Improvements

Added a filter for the ***Ended*** column on the ***Monitoring → Commands → History*** page

Bug fixes

Filtering by the ***Submitted*** column on the ***Monitoring → Commands → History*** page returned a result of filtering by the ***Ended*** column

The ADB Control agent generated a large number of threads in some cases

4.11.0

Date: 27.12.2024

New features

Changed collection and calculation of CPU and I/O metrics to make calculation of differences by processes in buckets more precise

Added SSL support for external databases used in ADB Control and ADBM (Metrics DB, Query DB, Offload DB)

Changed relative metrics calculation. Skew for all metrics is now calculated based on data from all segments, not only the ones that are active during the command execution. CPU average usage percent summary calculation is changed to consider logical segment mapping on hosts

Improvements

Optimized a time frame for segment system metrics refreshment in command details

Eliminated the direct `GpRole` usage on the ADB Control side for the Greenplum 7 support

Added validation for fields with domain names or IP addresses in the ADB Control/ADBM service configuration

Added a custom temp directory for the `adcc_scheduler` Docker service to eliminate errors caused by the `libzstd` inavailability

Bug fixes

Fixed: a modal window opened instead of the main login page

4.10.3

Date: 14.11.2024

New features

Added support of LDAPS for ADB Control users

Added the ability to configure a dedicated subnet for ADB Control and ADBM (see ***ADB Control → Use dedicated network*** in Configuration parameters)

Changed display of erroneous slices in the Plan & progress section on the page with command details

Improvements

Reworked processing a command counter in the ADB Control extension

In the `adcc` extension, added optimizations to the process of the `query_band` usage that help to avoid errors in `SELECT FROM get_stat_activity()`

Upgraded Spring Boot to 3.2.3

Extended the `sql_id` type for compatibility with ADB 7

Changed CPU usage calculation to average instead of sum on the Dashboard page

Updated names of some elements on the ***Dashboard*** page

Added the custom ***Login mount point*** parameter for integration with the Vault storage (see ***Vault integration → Login mount point*** in Configure a cluster)

Unified API conventions for the ADB Control and ADBM HTTP REST services, including datetime ranges

Bug fixes

The session registered under one user moved to another user if cookies were not cleared

After transaction errors, plan nodes were still in the `Executing` status

For some commands with errors, an error text was not displayed in the command details

The metrics offload job failed after ADB upgrade to 6.27.1.56

The `adcc` extension queries could delay cluster stop

4.9.1

Date: 29.08.2024

New features

Added the ability to connect to an external Hashicorp Vault instance and store secrets in it

Added display of blocked sessions and commands waiting to be queued within resource groups

Improvements

Improved the layout of charts on the Dashboard page

Added pre-checks for cluster actions to ensure ADB Control has enough privileges in the external database

Extended filtering and sorting by segment system metrics on the command details page

Removed the outdated authentication mechanism for system users

Bug fixes

The ***Information*** page did not display the cluster info

Fixed escaping of special characters for values entered in filters with partial search

Fixed the `executing` status on command plan nodes after transaction cancellation

Fixed: a failed attempt to log in to ADB as LDAP user was registered in ADB Control as the `ADB` authorization type instead of `ADB LDAP`

Fixed occasional fails of applying a metrics offload configuration

4.8.8

Date: 27.06.2024

New features

Added support for clustered ClickHouse as an external database for ADB Control installation

Added the new role — ADBM Administrator

Reworked the Configuration → Clusters tab

The PlanChecker service now has a configuration file

Improvements

The CPU average usage metric is now averaged across processes instead of cluster segments

A connection URL is now changed in ADB Control after standby is activated in ADB

Added re-creation of the ClickHouse container volume on the ADB Control service upgrade

Improved the container security by enabling `ReadonlyRootfs` and the `no-new-privileges` options

Removed Nginx from ADB Control and made the ADB Control UI accessible with SSH tunneling in isolated environments

Added masking passwords to the migration service logs

Inserts to `audit_auth` are now splitted into smaller batches to avoid shutdown of Postgres in ADB Control

Moved the cluster spill metrics to a separate table to avoid deadlocks in the ADB Control Postgres instance

Added a relation schema name to the plan node details

Bug fixes

Fixed absence of elapsed time on plan slices for queries in progress

The `CPU Total` metric had a wrong calculation basis

The `Spill Skew` value sometimes shown more than 100%

The session table blinked on update in the UI

Pagination did not work on the ***Resource group → Hosts*** tab

Fixed the migration error that occurred on configuring ADB Control with external Postgres

4.7.5

Date: 11.04.2024

New features

Added the DB credentials setup option for ADB Control and ADBM service configurations

Introduced the query tagging feature in ADB. Tags are now visible in ADB Control

Added ability to set GUCs that manage ADB Control behavior

Refactored the detailed plan progress view for convenience

Added session management of ADB Control users. Session interruption is now available for users with the `Admin` and `Owner` roles. Additionally, owners can set the maximum duration for user sessions (see the User session timeout option)

Improvements

Improved Docker security with container user restrictions

Reworked external database interaction with ADB Control from security perspective: sensitive settings are now made by a customer

Added the LDAP group list actualization button

Updated PlanChecker components: deleted JQuery, bumped Bootstrap to 5.3.3 for security purposes

Added the ***Session*** field to the ***Audit***/***Operations*** pages

Bug fixes

Fixed the display of ***Total CPU usage percent*** metric on the ***Dashboard*** page: decimal value was displayed instead of percentage

Fixed the absence of highlighting for selected filters

Removed the ability to set incorrect values in the calendar

Fixed the rights setting for *keystore.jks*, which caused the `Permission denied` error on agents

Fixed: the `CPU Avg` metric for some queries was marked as `Infinity` or `NaN`

Fixed: information for some segments was absent in command details in case of a large number of cluster segments

4.6.4

Date: 28.12.2023

New features

Introduced an option to configure SSL/TLS for interaction of ADB Control/ADBM services

Added the ***Application name*** parameter for each session on the Monitoring → Sessions tab

Improvements

Implemented dynamic loading of an LDAP group list based on the user input on the Users → LDAP group mapping tab

Moved charts to a separate Dashboard page

Implemented changes in API and a system user management process to improve security of interaction between ADB Control and ADCM, and the process of registering external ADB clusters

Modified TTL (time-to-live) for ClickHouse tables to provide lower disk space consumption

Bug fixes

Fixed different values of the ***Session*** field displayed for ***Auth*** and ***Logout*** audit events within the same session on the Audit → Authorizations tab

Fixed the ***Export sessions*** checkbox absence on the Configuration → Job policy → Metrics offload tab

Fixed the ***Start time*** field mapping on the ***Monitoring → Sessions*** tab

Fixed aggregation of metrics for cluster `Summary` charts

4.5.3

Date: 02.11.2023

New features

Added the Sessions tab to the ***Monitoring*** page with the session termination option

Improvements

Improved a process of filtering in agents for gathering system metrics

Added an automated cluster archivation when a new cluster with the same JDBC URL is added

Removed the ***Create connection*** button from the ***Configuration → Clusters*** page due to the existing feature of automated adding external clusters via ADCM

Support for SSL/TLS in gRPC services was added

Bug fixes

A user IP address was not displayed in the ***Host*** column on the ***Audit → Operations*** and ***Audit → Autorizations*** pages

Fixed an absence of a record for a denied action attempt on the ***Audit → Operations*** page

4.4.0

Date: 31.08.2023

New features

The custom timezone setting is now available on the ***Configuration*** tab

New parameters for the metrics offloading job (`Export job`) were added

Improvements

Changed the order of side menu items

Deleting the last user with the `Owner` role is now prohibited

Bug fixes

The ***Show*** and ***Frequency*** settings for jobs and actions lists were not saved after switching to another page

Fixed the error of filtering by ***Job group*** on the ***Jobs → Audit*** page

4.3.3

Date: 10.08.2023

New features

ADB is now available as a target database for exporting metrics from ADB Control

Improvements

Filters for ***Object name***, ***Username***, ***Host*** columns on the Audit → Operations and Audit → Authorizations pages are now case insensitive and partial (allow you to search by the first characters entered)

Bug fixes

Password for an offload database was shown on the ***Audit → Operations*** page

Sorting by columns was not available on the ***Jobs*** page

User role was displayed as a digit on the ***Audit → Operations*** page

After selecting a cluster, agents for other clusters were displayed on the ***Information*** page

It was impossible to switch to the next list page in the ***Resource groups → Overview*** section if there were more than 5 segments in the list

4.3.2

Date: 20.07.2023

Improvements

New tables and filters were implemented

Bug fixes

Error when changing the user role

Error when launching the metrics offload job

Execution time was not showed in plan nodes after the command finished

Details for relation audit were not showing

Audit operation for updating the resource group configuration was not registered

4.3.1

Date: 30.06.2023

New features

Audit for unsuccessful ADB logon attempts was added

Audit for ADB Control users management actions is available now

Checking that a new password differs from the current one was implemented

Added an archiving action for a cluster to stop monitoring of the selected cluster with the ability to activate it again

Added filtering of queries shorter than a threshold on the adcc-agent side to improve agents stability

SQL signature (`sql_id`) was added for every command

Improvements

Start time/***End time*** filters are now visible by default on ***Commands History***, ***Transactions History***, and ***Backups*** pages

Status names formatting in filters and table columns is unified

Bug fixes

No message was displayed when a connection was successfully created

Toggle button for a table was enabled when there was no further information

Reset button didn't reset sorting

Displayed `NaN` for ***Avg load rate / sec*** on the ADCC agent information page

Cluster information data didn't update immediately after switching the cluster on ***Actions***, ***Backups***, ***Restores*** tabs

Black screen was displayed when switching to another cluster from restore details

Backup details action row was not displayed

Page path not changed when switching tabs

Performance metrics list for commands/transactions had duplicates

4.2.1

Date: 27.04.2023

New features

Arenadata DB Command Center (ADBCC) is renamed to Arenadata DB Control (ADB Control)

Added the ability to export metrics to an external database for a long-term storage

Added the CSRF (Cross-Site Request Forgery) protection support

Added a forced password change at the first user authentication in the system

Added the temporary blocking of the user account after several failed authentication attempts

Added calculation of total monitoring metrics. For active commands and transactions, `cpu-total`, `read-total`, and `write-total` are calculated at the current time from the start of execution

Added pagination for the ***Job monitoring*** page

Bug fixes

Fixed checking the name of the user that is being created. Only latin characters, numbers, and special characters in the user name are available

Fixed: no pop-up window when receiving an error

Fixed: a query could become "frozen" in the unknown status after 14-16 seconds

Misc/Internal

Decreased the log level in `gpadcc`

4.1.0

Date: 15.02.2023

New features

Using ADBCC along with `gpperfmon` within the same ADB cluster is now supported

LDAP search in several OUs during ADBCC authentication is now available

Added the ability to sort by almost all columns on the ***Command online/Command history*** and ***Transaction online/Transaction history*** pages

Added the ability to show all available filters on the ***Command online/Command history*** and ***Transaction online/Transaction history*** pages

Improvements

Added the ability to edit the `JDBC URL` and the `Username` cluster connection settings after the cluster creation

Added a pop-up window with information about the application of the resource group settings

On the detailed transaction page in the table of commands belonging to the transaction, renamed the ***Workload*** column to ***Resource group***

Added the ability to specify the sorting direction on the ***Resource groups*** page

Fixed breadcrumbs that are shown when viewing resource groups. Instead of cluster and group IDs, resource group names are now displayed

Bug fixes

Sometimes the ***cpu skew*** metric obtained negative values

The statistics calculation for an interval of more than an hour

The calculation for the ***Total queue duration*** column on the ***Resource groups*** page

Removed the ability to enter negative values for the ***Batch size*** and ***Expire duration*** settings on the ***Configuration*** page for a job policy

The pagination error on the ***User*** page

The calculation for the number of commands in the `running` status on the ***Command online*** page

The search for a command in the text when there is a newline character in the query text

4.0.0

Date: 06.12.2022

New features

Added a new user interface for ADBCC

Added a new module for ADB binary backups management — Arenadata DB Backup Manager (ADBM)

Added a new module responsible for performing scheduled tasks — `adcc-scheduler`

Added a functionality for monitoring resource groups:

- The page with existing resource groups.
- The page with detailed information for the resource group including active commands of the group.
- The ability to configure the resource group.

Improvements

All datetime metrics are now stored in UTC

For clusters being removed from the ADBCC configuration, uncompleted commands and transactions are not updated

The `status`, `finishAt`, and system metrics (`readTotal`, `writeTotal`, and `cpuUsageTotal`) of the current active command are now updated immediately for a terminated transaction. The status of the command is set to `CANCELLED`

Bug fixes

The calculation of `Spill Skew` on the ***Command monitoring*** and ***Command history*** pages

The calculation of the node execution time for a command in ***PLAN & PROGRESS***

The calculation of system metrics with negative values

3.7.0

Date: 20.10.2022

New features

Added new performance metrics for commands and transactions: `Cpu usage total`, `Read bytes total`, `Write bytes total`

Added the ability to repeatedly change a resource group for a transaction

Improvements

Removed the ability to stop execution of a separate command by canceling or terminating it

Renamed the ***Workload*** column to ***Resource Group*** on the ***Monitoring*** and ***History data*** pages for commands and transactions, as well as for detailed pages

Removed the ability to change a resource group for a command on the ***Monitor Command*** page

For commands and transactions, rows are now sorted by the ***Status*** column value

Excluded the large bold font style for system metrics on detailed pages of commands and transactions

Bug fixes

Corrected the data sorting on the ***Audit*** page based on numbers, symbols, letters

It was impossible to enter data using the keyboard into the ***Start Time*** filter on the ***Monitor/Transaction*** pages

The error with appearance of multiple filters by cluster and by database, if a filter by database was being added from the filter list

Fixed the mapping of memory dimension from MiB to Mb

Fixed calculating the number of tuples affected by the request without updating the page

Misc/Internal

Included the Planchecker database objects into migration

3.6.0

Date: 30.08.2022

New features

Added a new column ***Planner***

Added the ability to filter commands by a planner

Added the ability to cancel a transaction on the ***Detailed transaction*** page

Added a button to reset and delete all previously applied filters on the ***Monitor*** and ***History command*** pages

Added the ability to reassign queries to another resource group

Added a configuration for a Planchecker address

Improvements

Changed the display of the `Spill Files (skew)`, `Row Skew` (in the ***Node*** panel), `Cpu Skew`, `Memory Skew`, `Read Skew`, `Write Skew` metrics to the uniform format: `<metric_name> Skew <metric_value>%`

Bug fixes

The ADB unavailability error in case of the `OutOfMemoryError` error in ADBCC

Fixed an actual tuples count for commands

Fixed the incorrect display of the command status on the ***Detailed transaction*** page if the transaction was cancelled

Misc/Internal

Added a Planchecker image to the ADBCC service (docker-compose) in the ADB bundle

3.5.1

Date: 21.07.2022

New features

Processing of NodeStatusMetric messages is supported

Added transaction pages: ***Transactions monitoring***, ***Transactions history***, page with detailed information about the transaction and a list of requests made within this transaction

Added the ability to filter data by schema and relation on the ***Query history*** page

Added a link to ***Transactions history*** in order to navigate to queries from the ***Audit*** page

Added a link to the parent transaction in the ***Command details*** tab

The SQL statement groups DDL, DML, DCL, and TCL are now displayed on the ***Monitor*** and ***History*** pages

Added ***Spill*** and ***Spill Skew*** columns to the customization table on the ***Monitor/Command*** and ***History/Command*** pages

Improvements

A non-blocking socket is now used to communicate with an agent

Bug fixes

Long values of the ***Query ID*** column overlapped the next column with data on the ***History/Command*** page

The incorrect sorting according to the schema with a heavy load on the ***Relation audit*** page

When a user moved from the statistics to the list of commands on the ***Relation audit*** page, the command was being duplicated

The missing tooltip for the ***Reset*** button

The `NullPointerException` exception occurred during attempts to get system metrics

Empty values were at the end of the list when the data was being sorted

The ***queuedTime*** field displayed different values on the ***Monitor*** page and the ***Command details*** page

The cache did not work properly, and the data was obtained directly from the database

After the ***Cluster name*** column was added, the cluster value was not displayed in the table on the ***Monitor/Command*** and ***History/Command*** pages

The `Duplicate key value violates unique constraint` error

The bug with sorting on the ***Monitor*** and ***History*** pages when all the records became hidden

3.4.0

Date: 11.04.2022

New features

Integrated the Avg cluster query metrics

Implemented compression for huge queries

Added a configuration for Registry Service to configure the addresses of backend servers

Improvements

The actual statistics from `EXPLAIN ANALYZE` for finished queries is processed now

The ***Plan&Progress*** and ***Text*** panels are hidden now if the content is empty

Bug fixes

ADBCC exceeded the limit of connections to ADB

Fixed the relation name display for a Dynamic Seq Scan node

The color-coding for plan node statuses was broken

The login error with cyrillic passwords

Misc/Internal

Migrated the `adcc-extension` to the *adbcc* repository

3.3.1

Date: 09.03.2022

Bug fixes

Filtering invalid metrics from `gpperfmon`

3.3.0

Date: 17.02.2022

New features

Data audit support

Background jobs history

Support for virtual process memory in system metrics of commands

Service load ratio support

Improvements

Changed the sort order for metrics with `NULLS LAST` on the ***Query Monitor***/***History*** pages

Reworked top menu

Misc/Internal

Bumped log4j2 dependencies

New system endpoints

3.2.5

Date: 20.12.2021

New features

HTTPS support in ADBCC SSL

The new `maxMessageKbSize` parameter in the `adcc-agent` configuration file

Bug fixes

The agent stopped processing query metrics due to `RejectedExecutionException`

3.2.4

Date: 18.11.2021

New features

Added the `maxMessageKbSize` setting to *agent.properties*

Improvements

Added a message size check

Bug fixes

Fixed re-sending of incorrect messages

Fixed getting system metrics from ClickHouse

3.2.2

Date: 12.11.2021

New features

Multi-clusters

New system metrics: `CPU`, `RAM`, `IO`

Adding columns dynamically to the ***History*** and ***Monitoring*** pages

Information about background processes on the system status information page

Improvements

Actualization for hanging queries

adcc-extension: database and username are now shown in a plan message

adcc-extension: lock polling is now disabled to avoid sending huge locks snapshots to agents

Misc/Internal

Using a ClickHouse database for saving draft system metrics

3.1.3

Date: 07.09.2021

New features

Added an agent build for Power8 LE platform (ppc64le arch)

3.1.0

Date: 17.08.2021

New features

Time-based Retention Policy

Fine-grained access to multiple databases for Advanced users

Extended information on errors

Column search filters and sorting on the ***Query Monitor*** and ***History*** pages

Improvements

Information on ADBCC/ADB versions and documentation

Scaling of the plan modal window

UI updates

3.0.3

Date: 03.08.2021

New features

New UI for Backend Server

Ability to cancel and terminate queries on the ***Query Details*** and ***Monitoring*** pages

Ability to view locks and blocks on the ***Query Details*** page

Support of displaying information for the ADBCC agent

LDAP authentication

Improvements

Truncating of query plan details

ADB Control known issues

Daria Barysheva

To Table of Contents

Issue	From	Fixed	Workaround
Clusters that are connected to an external ADB Control instance should be upgraded along with the main cluster to ADB 6.29.0.3 in order for ADBM and ADB Control to work properly	4.14.2	—	Upgrade clusters to ADB 6.29.0.3

In AltLinux 8, the Docker engine update may break the *docker-compose* package, which causes the ADB Control start to fail	—	AltLinux fix	To fix */usr/lib/python3/site-packages/compose/service.py*, replace the following:
			`container.image_config['ContainerConfig'].get('Volumes') or {}`
			by:
			`container.image_config['Config'].get('Volumes') or {}`

The ***Reconfigure*** service action may fail for ADB Control hosted on CentOS in case of using external Query DB, since the CentOS repository does not have the required PostgreSQL version	4.13.0	—	Follow the steps: 1. Uninstall completely PostgreSQL 9 from the host with ADB Control. 2. Install PostgreSQL client of version 14 or higher. Do not install ADB Control on the ADB master host
After the ADB Control service is configured to use LDAP authentication with `FreeIPA` type, a list of groups is not loaded when you create a new mapping on the ***Users → LDAP group mapping*** page	4.0.0	—	Use basic authentication or LDAP with `MSAD` type (Microsoft Active Directory)
When a cluster is upgraded from 6.27.1.56 to 6.27.1.58, an error occurs if external ADB Control is used	4.10.3	4.10.3	Upgrade a cluster with ADB Control before upgrading clusters that use this ADB Control as external (in ***External ADB Control server parameters***)
After the transaction cancellation, plan nodes in the Plan & progress section will have the `executing` status and display the increasing execution time	4.8.8	4.9.1	Keep in mind that cancelled queries do not have executing processes
ADB Control 4.8.8 implements changes in API that affect communication between the main and external ADB Control services of different versions. When an ADB cluster with the main ADB Control is upgraded to 6.27.1.56.b1, clusters of older versions that use this ADB Control as external will not be able to perform ***Reconfigure***/***Start*** actions with the ADB Control service	4.8.8	4.8.8	In order to make all ADB Control service actions to work, upgrade clusters with the external ADB Control and versions older than 6.27.1.56.b1 to 6.27.1.56.b1
After ADB Control upgrade to 4.3.2, the `Export job` settings (that are defined on the ***Configuration → Job policy → Metrics offload*** page) are reset to defaults	4.3.2	4.3.2	Fill in and save the required settings. The problem does not affect future upgrades
Standby activation after switching from master fails on clusters with ADB Control/ADBM installed	4.3.1	4.3.2	Use external ADB Control/ADBM
In ADB Control 4.2.1, you cannot register new external ADB clusters with versions earlier than 6.23.3	4.2.1	4.2.1	Update ADB cluster to 6.23.3

Switching off sending metrics of inner queries to ADB Control may cause hangs of ADB queries	4.1.0	4.10.3	Turn on the GUC that enables sending metrics of inner queries in ADB Control or in ADB (as follows):
			`$ gpconfig -c adcc.monitor_inner_queries -v on`
			`$ gpstop -u`

Commands appear in the `Running` status on the ***Monitoring → Commands → Online*** page but after 14-16 seconds they freeze in the `Unknown` status (with no metrics updated). After the command completion, all metrics become available on the ***Monitoring → Commands → History*** page, but the `Unknown` status persists	4.1.0	4.2.0	1. Using `psql` or other SQL IDE, set the `adcc` user password via the following command: `ALTER USER adcc WITH ENCRYPTED PASSWORD '<password>';` 2. Open the ***Configuration → Clusters*** page in the ADCC UI and go to editing the cluster connection. 3. In the ***Password*** field, set the `<password>` value from step 1. Click ***Save***.

There is a potential memory leak with specific queries affected by the `gpperfmon` engine. If you use queries against huge tables with UDF that generate inner per-tuple queries, you may experience the significant RAM consumption, and even get OOM errors. The example is the `information_schema.columns` view, which can cause memory leak when being selected against a large system catalog	4.0.0	4.1.0	The possible solution is to switch off monitoring metrics (this operation requires cluster restart):
			1. Remove `gpadcc` from *shared_preload_libraries*, e.g.:
			`$ gpconfig -c shared_preload_libraries -v ""`
			2. Switch off the following parameters:
			`$ gpconfig -c gp_enable_gpperfmon -v off`
			`$ gpconfig -c adcc.explain_log_verbose -v off`
			`$ gpconfig -c adcc.explain_log_analyze -v off`
			3. Restart your cluster:
			`$ gpstop -a -M fast`
			`$ gpstart a`
			Remember that this workaround does not disable the ADCC UI and Arenadata DB Backup Manager

Backend ADCC server cannot connect to ADCC UI server. Data update may not always work on the details page	4.0.0	4.1.0	1. Connect to the ADCC server via SSH.
			2. Go to the */opt/adcc* directory.
			3. Edit the *docker-compose.yaml* file. For the *adcc_ui_server* and *adcc_backend_server* services, change the value of the `ADCC_EUREKA_CLIENT_PREFER_IP_ADDRESS` environment variable to `true`.
			4. Run the command and wait until all Docker containers stop:
			`$ docker-compose down`
			5. Run the following command:
			`$ docker-compose up -d`

It is impossible to configure the `Data cleanup` job on the ***Configuration*** page. In general, ADCC can work without data cleanup, but you need to monitor the metrics database size	4.0.0	4.1.0	1. Connect to the ADCC server via SSH.
			2. Go to the */opt/adcc* directory.
			3. Edit the *docker-compose.yaml* file. For the *adcc_scheduler* service, add the following environment variable: `ADCC_SCHEDULER_GRPC_PORT: 6572`.
			4. Run the command and wait until all Docker containers stop:
			`$ docker-compose down`
			5. Run the following command:
			`$ docker-compose up -d`

ADBM releases

Daria Barysheva

Contents

To Table of Contents

2.8.1
2.8.0
2.6.6
2.6.5
2.5.7
2.5.5
2.5.0
2.4.0
2.3.2
2.2.3
2.1.2
2.0.4
1.7.3
1.6.3
1.5.2
1.4.0
1.3.3
1.3.2
1.3.0
1.2.1
1.1.0
1.0.0

2.8.1

Date: 23.12.2025

Bug fixes

- Fixed an issue where table restore would fail on tables with a high number of partitions

2.8.0

Date: 28.11.2025

New features

- ADBM now supports ADB 7
- `pgbackrest` updated to version 2.54

Bug fixes

- Fixed an issue where the status was changed to `failed` when an error occurred
- A backup that failed after the ***Resume*** action now shows 100% progress
- Added timeline ID comparison before and after the backup

2.6.6

Date: 08.10.2025

Bug fixes

- Fixed a bug where the ***Cleanup*** action would delete valid backups

2.6.5

Date: 17.09.2025

New features

- Added audit of operations with clusters and storage configurations
- Added the Storages tab and modified backup configurations accordingly
- Added representation of storage space utilization and availability

Improvements

- Implemented compatibility verification for database and agent versions
- Added processing of a new `XLogRecord` type during the ***Partial restore*** action

Bug fixes

- Fixed: WAL archives for backup were not verified during the ***Verify*** action
- Fixed: agents were not displayed on the ***Backup manager*** page when ADB was stopped
- Fixed: backup progress was sometimes incorrectly updated

Misc

- Changed the name of the stream entity to data flow and modified naming in UI accordingly

2.5.7

Date: 12.08.2025

Improvements

- Changed errors in jobs to more informative ones

Bug fixes

- Storage access was checked only from the master, which could cause incorrect backup status acquisition
- After launching a stream of the `Copy` type with the parallel mirror restore option, the *pg_hba.conf* file was missing on target segments if the source segments had no mirrors

2.5.5

Date: 05.08.2025

Improvements

- Added a check of the corresponding WAL chain during the ***Verify*** action
- Added a check of the PostgreSQL backend availability during the table restore

Bug fixes

- Fixed filtering WAL for headers that were divided across two pages
- Fixed handling backup blocks during filtering WAL in `pgbackrest` to avoid `gpdb` assertion triggering during the ***Partial restore*** action
- In `pgbackrest`, fixed handling a long record at the end of a previous WAL file during the ***Partial restore*** action
- Fixed errors that occurred during the ***Partial restore*** task execution in temporary PostgreSQL instances when running the `pgackrest archive-get` command
- Fixed the search for neighboring WAL files in other directories during filtering WAL
- In filtering WAL, replaced the error of missing the next WAL file with a warning
- Fixed the overwrite `contrecord` (continuation record) handling at the beginning of the next file when filtering the WAL
- Fixed the `record is too big` error when filtering WAL
- Fixed filtering WAL when a checksum is in the next segment file
- Fixed filtering WAL when there is no unfinished record at the end of the segment file
- Fixed handling WAL records larger than a WAL segment
- Fixed: a segment did not select timeline of a restore point during ***Partial restore*** after switching to a mirror
- Fixed reading an unfinished record at the end of the file when the next file is partial

2.5.0

Date: 23.06.2025

New features

- Added display of the Common restore action progress
- Removed the ***Skip validation*** option from the data restore form if a cluster is down. Now it is detected automatically and works for data restore, copy and streaming, giving the ability to recover a stopped cluster. ADBM agents should be alive on the stopped cluster hosts for this option to work
- Reworked the Verify action: backup files are now verified along with metadata. Added the corresponding option ***Verify*** to the Common restore, Partial restore, stream launch actions

Improvements

- Changed the `archive_mode` setting value to `on` to avoid the `Duplicates found in archive for WAL segment` error that occurred when primary and mirror segments tried to archive the same file. Note: ADB cluster will be restarted after ADBM upgrade because of the `archive_mode` change
- Implemented storing the `ic_proxy_port_delta` parameter on a cluster level for proper handling of the `ic_proxy` configuration during data restore
- Created the new page Backup manager → Configuration → Schedule to manage scheduling for a cluster. Also, reworked a schema of scheduling data
- Reworked a schema of cluster-related data and changed cluster addressing from a name to GUID
- In the ADBM agent, improved processing of the mirrors absence

Bug fixes

- Fixed handling of auto terminate actions for timed out backups
- Fixed the `A conflicting conditional operation is currently in progress against this resource` error that occurred when checking S3 writability

2.4.0

Date: 07.03.2025

New features

- Estimated progress is now shown for backups
- Added server-side encryption options for S3: `kms-key-id` and `sse-customer-key` (see ***Server-side encryption options*** in Manage configurations in ADBM → Add a new configuration → Repository)
- Added a choice of the restore point streaming error handling mode on stream creation (see ***On restore points streaming failure*** in Disaster recovery). Added the `Paused` status for streams and Resume/Cancel options for this status

Improvements

- Added storing of the `gp_interconnect_type` value in the cluster topology to support force restore on clusters with configured `ic_proxy`
- Changed the ***Apply backup configuration*** action logic to restart a cluster only when the `archive_mode` value is different from `always` and has to be set
- Added caching of the ***Recover segments*** action on the agent side for high availability (HA) purposes
- Adjusted the `await-segment-ready` parameters in the ADBM agent properties to optimize recovery time and avoid streaming failures by timeout

Bug fixes

- Data restore with the ***Force*** and ***Skip validation*** flags set could fail if there was switch to standby
- In ADBM CLI, some parameters were not returned in the command output and parameter compatibility was not checked

2.3.2

Date: 27.12.2024

New features

- Removed the etcd component from the ADBM architecture and assigned its functions to PostgreSQL
- To avoid illogical assignment of the `Invalid` status to backups, modified the ***Verify storage*** action to check repository accessibility from all agents instead of the master host only

Improvements

- A list of cluster actions now contains only currently allowed actions to eliminate errors that occurred when the ***Verify*** action was not applied after cluster upgrade
- For the backup report command of ADBM CLI, added the ability to select the output format in the `--output` option
- Added actualization of agent roles before data restore with the ***Force*** option — to support force restore on a target cluster with a lost dispatcher role
- Added filters, sorting, and pagination to the table on the restore details page

Bug fixes

- Fixed handling of `gp_interconnect_proxy_addresses` during data restore in an ADB cluster
- Fixed: the Partial restore action launched on a source cluster could terminate streaming
- Fixed: a filtering configuration for Partial restore was not used
- Fixed: a target cluster was shown in the ADB Control UI with the `Up` status while streaming

2.2.3

Date: 14.11.2024

New features

- Introduced an option of partial recovery from binary backups

Improvements

- Added parallel Cleanup execution on agents to avoid termination of this action by timeout on large installations
- Updated the backup configuration version display
- In the ***Subactions*** section on the page with action details, added line breaks to the display of long action names
- In the process of cluster reinstallation, added restart after the ADBM client installation to set the `PGBACKREST_CONFIG` environment variable

Bug fixes

- The `isQuiet`, `isVerbose`, `backupDir` options did not work for the `backup` and `backup config` commands in ADBM CLI
- Some elements were not aligned in the ADBM action details
- ADBM restarted ADB upon backup configuration changes even when `archive_command` was not changed
- The shortened version of the `database` option for commands did not work in ADBM CLI
- Backups in the `Invalid` status were not deleted during the ***Cleanup*** action
- The `archiver` process failed after the host recovery

2.1.2

Date: 29.08.2024

New features

- Implemented new parameters for S3 in backup configurations: ***Upload chunk size*** and ***Certificate verification*** (see the ***Repository*** section in Manage configurations in ADBM)
- Added integration with Vault

Improvements

- The Resume action functionality is now available only for full backups
- Restricted the number of backups in the `Stopped` status to `1` and added a history of backup status changes
- Added a limit for a total number of backups
- Added validation for a compression level in a backup configuration

Bug fixes

- Fixed the ***Cleanup*** action for backups in the `Unknown` status
- Fixed: backups were not transferred to the `Invalid` status
- Fixed: `gprecoverseg` failed on an agent during the ***Restore*** action due to running data backup or restore in ADB
- Fixed: no information on a backup after changing its status from `Unknown` to `Done`

2.0.4

Date: 27.06.2024

New features

- Implemented CLI for logical backups
- Enabled the feature of block-level incremental backups: the corresponding flag is added to the backup configuration page

Improvements

- Added support for expanding an ADB cluster for ADBM
- Added the ***Verify backup*** action to a stream flow

Bug fixes

- Removed the ability to delete a configuration for an active stream
- Fixed: the restore process failed for a restore point created after one of the segments went down
- Question marks were displayed in the ***Exclude path*** field if its value was added in Russian
- Fixed the incorrect `Warning` status of the ***Terminate*** action applied to a stream after several clicks on the ***Run*** button in the ***Terminate stream*** modal window
- Fixed: the only cluster backup configuration was deleted after several streams and cleanups were launched
- A backup with the `Applied` configuration failed if the `Draft` configuration existed
- Backups failed after upgrading an ADB cluster from 6.25.1 to 6.26.2
- ADB master stopped during the restart of *adbm-agent*

1.7.3

Date: 11.04.2024

New features

- Added a new option for Disaster Recovery: standby cluster can now be restored from WAL changes of a source cluster. Restore from a source backup is required only on the initial step, not on every iteration

Improvements

- Improved handling of an S3 repository in ADBM actions considering essential configuration parameters
- Cleanup logic was updated to delete all incremental and differential backups that might be left by policy
- Added the `liquibase.releaseLocks` parameter for ADBM migration service

Bug fixes

- Fixed: copy restore failed after stopping one of the source cluster segments
- Fixed a stream status type on applying impossible streaming configuration
- Fixed: the `Unknown` status for backups was absent

1.6.3

Date: 28.12.2023

New features

- Added new statuses for management of backup configurations
- Implemented changes in the ***Cleanup*** action to support modifications of a repository type or path, and to protect backups during Disaster Recovery (DR) copy stream

Improvements

- Added handling of repository unavailability to the process of backup status determination

Bug fixes

- The ***Backup*** action flow was being stopped when subactions were in the `Warning` status
- Some details of the ***Restore*** action were not visible in the light theme

1.5.2

Date: 02.11.2023

New features

- Implemented the ability to restore mirrors from binary backups in parallel with primary segments — via the Restore mirrors field. Note that it is an experimental feature

Improvements

- Verifying the repository accessibility was added for a copy stream command flow
- Added the Warning status for ADBM actions
- Date filter defaults were changed to the last day on the ***Backups*** and ***Restores*** pages
- Added the forced ***Skip validation*** flag display to the Restore modal window for restoring data in stopped ADB clusters

Bug fixes

- Fixed the error of running incremental backups connected to zero-sized files
- Fixed the verification error that occurred when an S3 repository with self-signed certificates was used
- Fixed the error of creating ADBM configuration for S3 storages with the `path` URI type
- Fixed downgrade in the ADBM UI performance when one of clusters was unavailable
- Fixed the error of restoring from a backup after a successful copy stream on a DR cluster

1.4.0

Date: 31.08.2023

New features

- CIFS repository support was added
- Added the option of automated backup copying to a target cluster when a new backup on a source cluster is completed

Improvements

- Action id was added into action details
- Auto detection of an active master host was added to support a backup process after switching to standby

Bug fixes

- It was impossible to restore backups in clusters with non-unique names within one ADBM
- Self-restore (via the ***Restore*** action) failed after changing roles on mirrors
- S3 certificate was deleted from segments after the host reboot

1.3.3

Date: 10.08.2023

New features

- Added the restore with mirrors option for Disaster Recovery (DR) copy process
- The repository validation step was added before saving a backup configuration

Improvements

- Backup topology data is now sorted by ***Segment***

Bug fixes

- ***Restores*** list was empty for clusters with spaces in names
- DR copy action failed when source and target clusters had different standby configurations

1.3.2

Date: 20.07.2023

New features

- Restore for standby master was implemented
- Automatic checking of ADB versions compatibility on source and target clusters was added to the Disaster Recovery (DR) copy process
- Availability check for a backup configuration on a target cluster was added to the DR copy process

Bug fixes

- Copy stream failed on a cluster with standby
- Could not restore a backup on DR cluster with a number of datanodes different from a source cluster
- Choosing particular databases from the database list caused the Disaster Recovery (DR) copy process to fail
- DR cluster could not be started after restore

1.3.0

Date: 30.06.2023

New features

- Implemented Disaster Recovery with Cold Standby — ability to recover a cluster from a backup located on another cluster. Beta version! Please pay attention to the ADBM known issues section and backup a target cluster before starting a stream
- Added `pgbackrest` options (`-delta`, `--force`, `--max-process`) for the restore action
- Added an option to save a backup configuration without schedule for running backups on request

Improvements

- ADBM action tree view was enhanced
- Enabled resuming failed backups considering segments that have already been backed up

Bug fixes

- Fixed: the ***Apply backup config*** action failed when S3 was selected

1.2.1

Date: 27.04.2023

New features

- Added support for PowerPC-based computers
- Added the ability to restore data from backups without mirror segments
- Added the ability to filter restore points for the ***Restore retry*** mechanism
- Added the Restores tab to view ***Restore*** actions

Bug fixes

- Creating an additional configuration when reinstalling without creating a backup
- The error with using an S3 repository to store backups

Misc/Internal

- Added the `openapi-yaml` generator to generate OpenAPI files
- Added a timeout to read an OpenAPI file

1.1.0

Date: 15.02.2023

New features

- The list of available cluster actions is supported now
- Added the ability to delete the last backup from the stanza. It expands the cleaning functionality logic
- Added the ability to restore specific databases from backups

Improvements

- Added the ability to use the ***Force*** flag when a cluster is being restored from a backup. It means bypassing all checks
- Added the validation for the `cron` expressions that are used when ADBM configuration is being created or edited
- Added a check for `archive_command` when saving the ADBM configuration. It allows avoiding unnecessary restart of the ADB cluster

1.0.0

Date: 06.12.2022

New features

- Implemented the first ADBM version. For more details, see Arenadata DB Backup Manager overview

ADBM known issues

Daria Barysheva

To Table of Contents

Issue	From	Fixed	Workaround
ADBM agents with a version lower than 2.8.0 (compatible with ADB 6) cannot connect to the ADBM server 2.8.0	2.8.0	—	Use separate ADBM deployments for ADB 6 and ADB 7 clusters

If an NFS or CIFS storage used for backup becomes unavailable, the `pgbackrest` process can hang, making ADBM operations unresponsive. For example, a backup operation can remain in a hung state even after the ***Protocol timeout*** period (defined in the backup configuration) expires	—	—	

1. Identify the `pgbackrest` process and check its state:

```
$ pgrep -a pgbackrest
$ ps -o pid,state,command -p <PID>
```

   If the `state` parameter has the value of `D`, the process has entered an `Uninterruptible Sleep` state and cannot be terminated via a SIGKILL signal or `pgbackrest` timeouts until connectivity with the storage is restored.

2. Mount your NFS or CIFS storage with the `soft` option, which allows system calls to return errors instead of waiting indefinitely. For example, add the following to the */etc/fstab* file:

   - For NFS:

```
server:/share /mnt/share nfs
soft,timeo=300,retrans=3 0 0
```

   - For CIFS:

```
//server/share /mnt/share cifs
soft,timeo=300,retrans=3,credentials=/path/to/crede
ntials 0 0
```

The ***Audit*** page does not show operations	2.6.5	—	In ADB Control → UI component parameters, set ***Listening port*** to `8890` for the main and external clusters. Run the ***Reconfigure & Restart*** action to apply changes
Clusters that are connected to external ADB Control should be upgraded along with the main cluster to ADB 6.29.0.3 in order for ADBM and ADB Control to work properly	2.6.5	—	Upgrade clusters to ADB 6.29.0.3
The standby’s WAL directory is not recycled properly if `wal_sender_archiving_status_interval` is greater than `0` and `archive_mode` is `always`, which causes data restore in ADBM to crash in some cases	2.0.4	—	Set `wal_sender_archiving_status_interval` to `0`
The backup repository configuration creation fails in ADBM CLI when the S3 plugin is used with default `uploadChunkSize` and `uploadConcurrency` settings	2.0.4	—	Configure `uploadChunkSize` and `uploadConcurrency` based on your system resources before creating a backup configuration. The S3 plugin will try to allocate the memory size equal to `segment_per_host` * `uploadChunkSize` * `uploadConcurrency`
The ***Information*** → ***Backup manager*** page does not display information for segment hosts in the `DOWN` status	1.0.0	—	This is the expected behavior that may be improved in future releases
Termination fails for streams with the ***Restore mirrors*** option set to `In parallel` (during the stream creation)	2.2.0	2.4.0	When creating streams with the `Streaming` restore type, set the `After primary` value for the ***Restore mirrors*** option
After upgrade to ADB 6.26.2, backups with the configuration created before upgrade fail	1.7.3	2.3.2	After upgrade, run the ***Verify*** action for clusters that have backup configurations

Restore fails if one of the cluster segment hosts is down for restore points created after the host was stopped	1.7.0	2.0.4	

1. Edit the *ssh_config* file under `root` on the master host:

```
$ vi /etc/ssh/ssh_config
```

2. Add the following line to the end of the file:

```
ConnectTimeout 20
```

3. Restart the SSH daemon (`sshd`):

```
$ systemctl restart sshd
```

The ***Create Stream*** modal window has the following restriction: the `Streaming` restore type is only a prototype, do not use it	1.3.1	1.7.3	Use only the ***Copy*** option for a stream
Mirrors are not restored automatically on a target cluster. If a target cluster already has mirrors, the restore job will fail, and if it doesn’t then the restore job will finish without errors but the restored cluster will not have mirrors even if the source cluster had	1.3.1	1.3.3	DR cluster shouldn’t have mirror segments
The ***Create Stream*** and ***Restore*** modal windows have the following restriction: the ***Restore mirrors*** option should not be used	1.3.1	1.3.3	Reset the ***Restore mirrors*** checkbox when setting up a stream or restore
The Disaster recovery copy action fails when: - A source cluster doesn’t have a standby and a target cluster has it. - A source cluster has a standby and a target cluster has not.	1.3.1	1.3.3	Standby should be present or absent on both source and target clusters
Restores list is empty for clusters with spaces in names	1.1.0	1.3.3	Cluster names shouldn’t have spaces
Standby restore for a target cluster is not available	1.3.1	1.3.2	DR cluster shouldn’t have standby master
It is impossible to restore data from a backup on the DR cluster when the source cluster number of data nodes differs from the DR cluster	1.3.1	1.3.2	DR cluster and source cluster should have the same topology
Cluster is marked `Down` in ADBM when master is on the same host as segment	1.3.1	1.3.2	Edit */opt/adbm-agent/service-configuration* and leave only one record for this host that describes ADBM agent roles: `ADBM_AGENT_ROLES=EXECUTOR,DISPATCHER`, then restart agent with the `systemctl restart adbm-agent` command
DR cluster cannot be used for writing	1.3.1	1.3.2	Do not use DDL or DML commands on DR cluster

DR cluster cannot be started after restoring	1.3.1	1.3.2	

1. Backup *pg_hba.conf* of `-1` and `0` segments on DR cluster before running a restore stream. Save it to persistent storage (do not use */tmp*). Use the following commands (under the `gpadmin` user) on master host:

```
$ cp /data1/master/gpseg-1/pg_hba.conf /home/gpadmin
$ gpssh -v -f
/home/gpadmin/arenadata_configs/arenadata_segment_host
s.hosts -e 'cp /data1/primary/gpseg*/pg_hba.conf
/home/gpadmin/'
```

2. Restore *pg_hba.conf* after completing ***Disaster recovery copy stream*** action via the following commands (under the `gpadmin` user) on master host:

```
$ cp /home/gpadmin/pg_hba.conf /data1/master/gpseg-1/
$ gpssh -v -f
/home/gpadmin/arenadata_configs/arenadata_segment_host
s.hosts -e 'find /data1/primary/gpseg* -maxdepth 0 -
exec cp /home/gpadmin/pg_hba.conf {} \;'
```

3. Change the `synchronous_standby_names` parameter on all primary segments from `'*'` to `''` in */<data_directory>/primary/gpseg*/postgresql.auto.conf*.

The ***Create Stream*** and ***Restore*** modal windows have the following restriction: choosing particular databases from the ***Database*** list will cause the job to fail, do not use it	1.3.1	1.3.2	Select all databases when setting up a stream or restore

It is impossible to add external ADB clusters to the general-purpose ADBM from the ADBM UI	1.0.0	1.3.1	

1. Ensure the external ADB cluster with the installed ADBM service exists in ADCM.

2. Connect to the Linux console. Generate the Basic Authorization hash with base64 for a user with access to Arenadata DB Backup Manager via the following command: `echo -n <adbm_user>:<adbm_user_password> | base64`. For example:

```
$ echo -n adbm_user:Password | base64
```

   Result:

```
YWRibV91c2VyOlBhc3N3b3Jk
```

3. Send the following POST request to the general-purpose ADBM host:

   - Header parameters:
     - `Accept`.
     - `Accept-Charset`.
     - `Authorization`. It should have the value started with `Basic ` and then the base64 value from step 2.
     - `Content-Type`.
     - `X-ADBM_USER`. It should have the `adbm_user` value from step 2.
   - Body:

```
{
"clusterName": "<name of external ADB cluster in
ADCM>",
"clusterType": "<purpose of ADB cluster>",
"clusterVersion": "<version of ADB cluster>"
}
```

   The following is an example of the POST request with the Linux `curl` command:

```
$ curl --request POST \
--url http://adbm-
main:8890/backend/adbm/api/v1/clusters \
--header 'Accept: application/json' \
--header 'Accept-Charset: ISO-8859-1' \
--header 'Authorization: Basic
YWRibV91c2VyOlBhc3N3b3Jk' \
--header 'Content-Type: application/json' \
--header 'X-ADBM-USER: adbm_user' \
--data '{
"clusterName": "adb-demo",
"clusterType": "ADB demo stand",
"clusterVersion": "6.22"}'
```

   A successful response contains a JSON with the generated identifier of an external cluster in ADBM. The cluster becomes visible in ***Arenadata DB Backup Manager*** → ***Clusters*** with the `Down` status.

4. In ADCM, go to the external ADB cluster and select ***Services*** → ***ADBM*** → ***Configuration***.

5. Set the ***Show advanced*** flag and edit the ADBM agent service configuration parameter `ADBM_EUREKA_CLIENT_SERV_URL_DEF_ZONE` located in the bottom area. Replace its value with URL of the service registry for a general-purpose ADBM host, e.g. `http://adbm-main:8761/eureka`.

6. Apply the ***Reconfigure & Restart*** action to ADBM. Upon success, the cluster status in ***Arenadata DB Backup Manager*** → ***Clusters*** is changed to `Up`.

FQDN names in the `hostname` and `address` columns in the `gp_segment_configuration` table are not supported. Essentially, it is a blocker for ADBM utilizing in clusters where FQDNs are used	1.0.0	1.1.0	Modify names in the `gp_segment_configuration` table to short ones. Take into account that it is an extremely dangerous operation, which can harm a cluster
Segment servers have an additional role that they don’t need to have. ADBM will work, but with extra queries on segment servers. Users can discover it only from a log	1.0.0	1.1.0	Change a configuration file on the ***ADCM*** → ***ADBM*** → ***Configuration*** page: 1. Set the ***Show advanced*** flag. 2. Find the ***ADBM agent service config parameters*** field. 3. Replace `master` by `adb.master` and `segment` by `adb.segment`. 4. Click ***Save*** and then run the ***Reconfigure & Restart*** action.