AD.CM

Arenadata Cluster Manager

Arenadata Cluster Manager (ADCM) provides an efficient way to deploy and manage all data services of a company regardless of the infrastructure used — in the cloud, on-premises or as PaaS-services. By dividing the data platform into two logically untied layers — the infrastructure layer and the applications layer — the operation of the entire platform becomes predictable, flexible, and repeatable.

Version **CURRENT** Language: **EN**

Contents

1. Concepts
   - ADCM overview
   - ADCM architecture
     - Architecture overview
     - Cloud providers and services
     - Proxying
     - Integration with the bundle
   - ADCM objects
     - Bundle
     - Hostprovider
     - Host
     - Cluster
     - Service
     - Component
   - Maintenance mode
   - Concern
2. Planning guide
   - Prerequisites
3. Get started
   - Installation
   - Use ADCM Installer
   - Connect to ADCM
4. Access management
   - ADCM role model
     - Create roles
     - Create policies
     - Policies migration
     - Configure custom roles
     - Configure user permissions
   - Users management
   - Enable HTTPS
   - Enable LDAP
5. Maintenance
   - Work with an external database
   - Start and stop
   - Backup and restore
   - Upgrade
   - Configure
   - Uninstall ADCM
   - Logging
     - Configure logging
     - Log rotation
   - Audit
     - Audit operations
     - View audit logs
6. Web user interface
   - Bundles
     - Upload a bundle to ADCM
   - Hostproviders
   - Hosts
   - Clusters
     - Import/export of services
     - Set up configuration groups
     - Set up host groups for actions
   - Jobs
   - Configuration parameters
7. Development tools
   - Ansible ecosystem
     - Ansible modules and plugins
     - Ansible inventory groups
     - Ansible inventory structure
   - Prototype definition guide
   - Create a simple bundle
   - ADCM API
     - ADCM API v2 usage examples
   - ADCM AIO Client
8. Release notes
   - ADCM
   - ADCM Installer

ADCM overview

Mikhail Serov

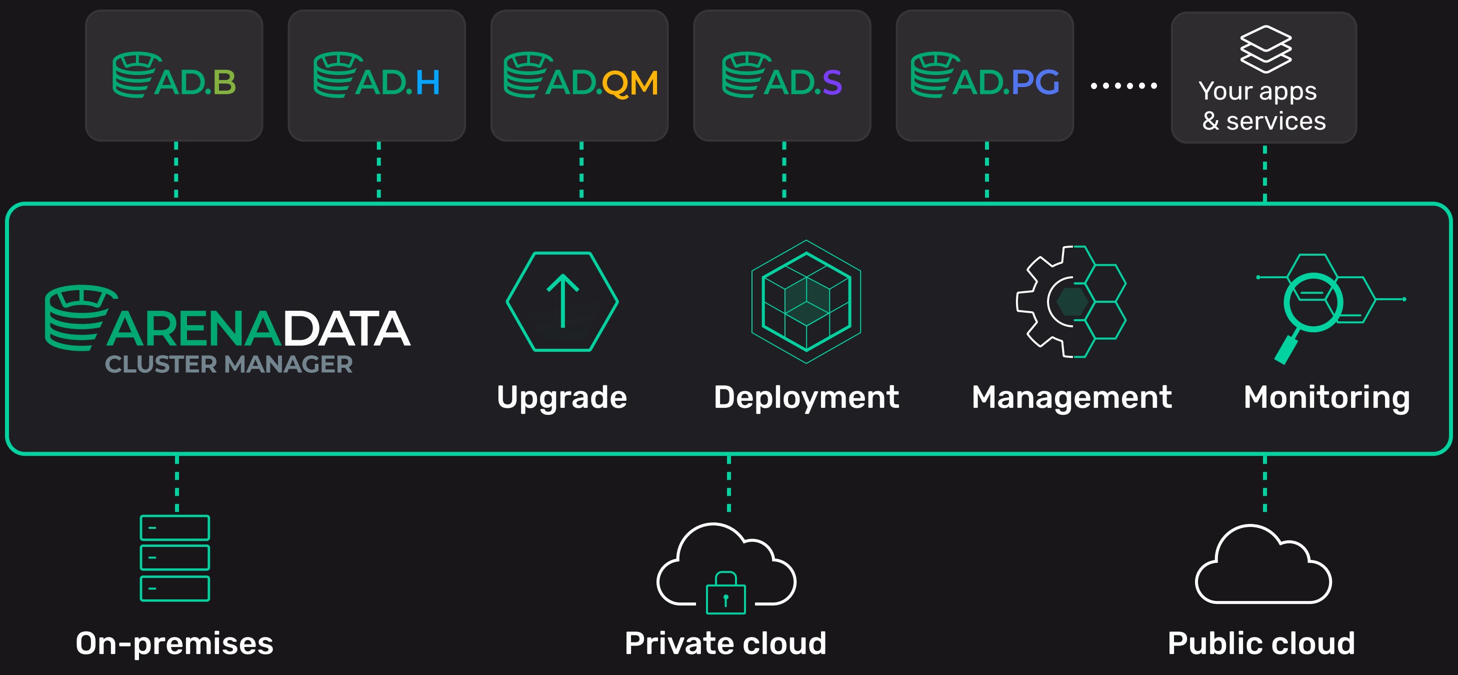
Contents

To Table of Contents

ADCM possibilities

Arenadata Cluster Manager (ADCM) is a data platform as a service solution (DPaaS) that simplifies deployment and management of data services on-premises, in private or public clouds. By splitting data landscape into two independent management layers — infrastructure layer and application layer — your platform becomes flexible, predictable and repeatable.

ADCM allows installing, configuring, and upgrading clusters via pressing a button in the graphical interface or via API request. All setup for an operating system, services, network, as well as disks mounting, is made automatically. ADCM assets manifest themselves to the fullest extent while working with heterogeneous infrastructure: you have the ability to place data services at various infrastructure objects — clouds, on-premises, or PaaS services.



ADCM interaction with heterogeneous infrastructure

ADCM possibilities

Service possibilities:

- Installing and configuring.
- Upgrading.
- Managing.
- Monitoring.
- Configuring access rights.
- Integration with other services.
- Containerization.

Infrastructure possibilities:

- Creating and removing virtual machines.
- Configuring an operating system.
- Monitoring.
- User management.
- Configuring access rights.

Additional possibilities:

- Configuring security rules in ADCM.
- Open ADCM API.
- Open format for creating new bundles — add your own services and infrastructure.

Architecture overview

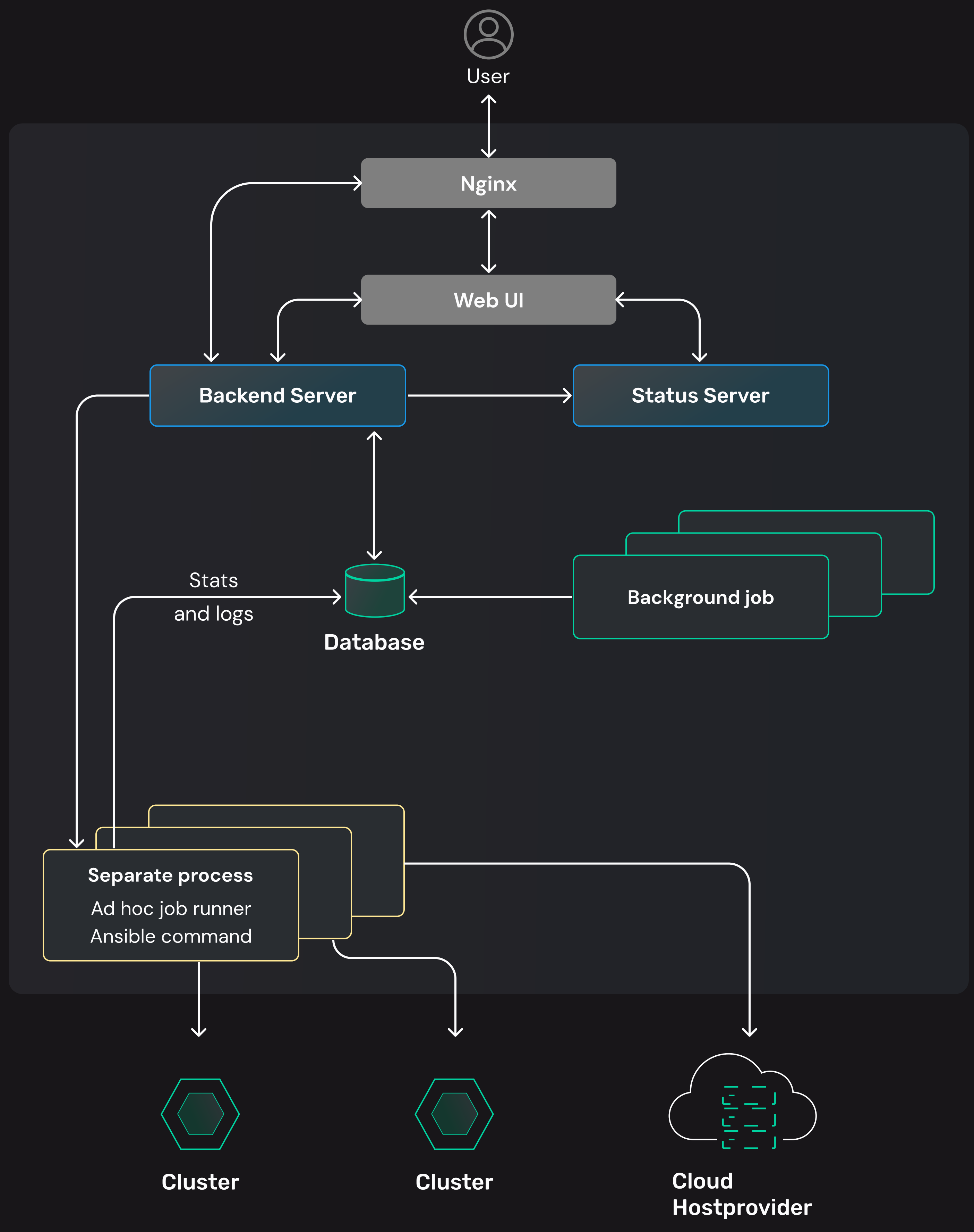
Mikhail Serov

To Table of Contents

ADCM main idea is to provide a simple user interface that produces enough data to run Ansible scripts on a number of hosts.

ADCM architecture is based on the following components:

- **Nginx**. Ensures ADCM availability (for example, in case of many users present) by distributing the load in front of WebUI. Nginx starts on the `8000` port. Nginx's main purpose for ADCM is to act as a proxy that redirects user queries to WebUI.
- **WebUI**. Receives user queries redirected by Nginx.
- **Status Server**. Receives heartbeats from hosts, services, and their components, as well as WebUI events. Sends web sockets to WebUI. Status Server is implemented in Go programming language.
- **Backend Server**. Contacts Status Server via the REST API and receives host, service, and component heartbeats. Backend Server is able to read and write data to the database and to invoke job runners to run Ansible commands.
- **Database**. Stores various data encrypted via Ansible Vault in the form of Ansible secrets.
- **Background job**. Works on various background tasks (for example, removing some data from the database according to the time period specified in ADCM). Background jobs use a database to receive information on how often they should run.
- **Ad hoc job runner**. Prepares environment to run Ansible via creating an inventory file, then runs Ansible command. For cluster and hostprovider, a job runner starts the set of some commands at host to affect the whole cluster performance (for example, starting a service, configuring files, or installing packages). Job runner works asynchronously: the browser running ADCM does not get locked, and after the process is completed, the result gets written into the database to be accessed by the Backend Server. Job runner prepares the environment from an inventory file every time an Ansible command is run, which is implied by the Ad hoc prefix (meaning "created for a particular purpose"). Each job runner is not a thread, but a separate process (in UNIX terms).



ADCM architecture

Cloud providers and services

Mikhail Serov

Contents

To Table of Contents

Supported services

Supported cloud providers

Supported services

ADCM supports deployment and management of product services.

Currently supported services are listed below according to their products.

ADB

- Greenplum

ADH

- Airflow
- Flink
- HBase
- HDFS
- Hive
- Phoenix
- Solr
- Spark
- Sqoop
- Tez
- YARN
- Zeppelin
- ZooKeeper

ADPG

- PostgreSQL

ADPS

- Ranger
- Solr

ADQM

- Clickhouse

ADS

- KSQL
- Kafka
- Kafka REST Proxy
- Kafka-Manager
- MiNiFi
- Nifi
- Nifi-Registry
- Schema-Registry
- ZooKeeper

Supported cloud providers

ADCM supports working with virtual machines for certain cloud providers. Supported virtual machines actions are creation, deletion, switching on and off, and changing settings.

Currently supported cloud providers are:

- Yandex
- VK Cloud
- VCloud
- Cloud.ru Advanced
- CROC

Proxying

Mikhail Serov

To Table of Contents

A reverse proxy server is a type of proxy server that sits behind the firewall in a private network and directs client requests to the appropriate backend server. A reverse proxy provides additional level of abstraction and control to ensure smooth flow of network traffic between clients and servers.

You may want to use reverse proxy for TLS termination, caching and compressing content, or some other purpose.

Here is an example of Nginx configuration file:

```
server {
  listen 443 ssl;
  server_name <your_adcm_domain>;
  ssl_certificate      example.crt;
  ssl_certificate_key  example.key;
  ssl_protocols        TLSv1 TLSv1.1 TLSv1.2;
  ssl_ciphers          HIGH:!aNULL:!MD5;

  access_log /var/log/nginx/adcm.access.log;
  client_max_body_size 60M;  # used to increase size of an uploaded file. Size by default is 1MB

  location / {
    proxy_pass http://<ip_adcm>:<port_adcm>;
    proxy_read_timeout 90;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto https;
  }

  location /ws {  # used for proxying websocket API
    proxy_pass http://<ip_adcm>:<port_adcm>;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
  }
}
```

Integration with the bundle

Mikhail Serov

Contents

To Table of Contents

Load the bundle
Start an action

When a new bundle is loaded into ADCM, information about the prototypes is added based on the description specified in the bundle when it was developed. The following sections describe the process of integrating ADCM and the new bundle.

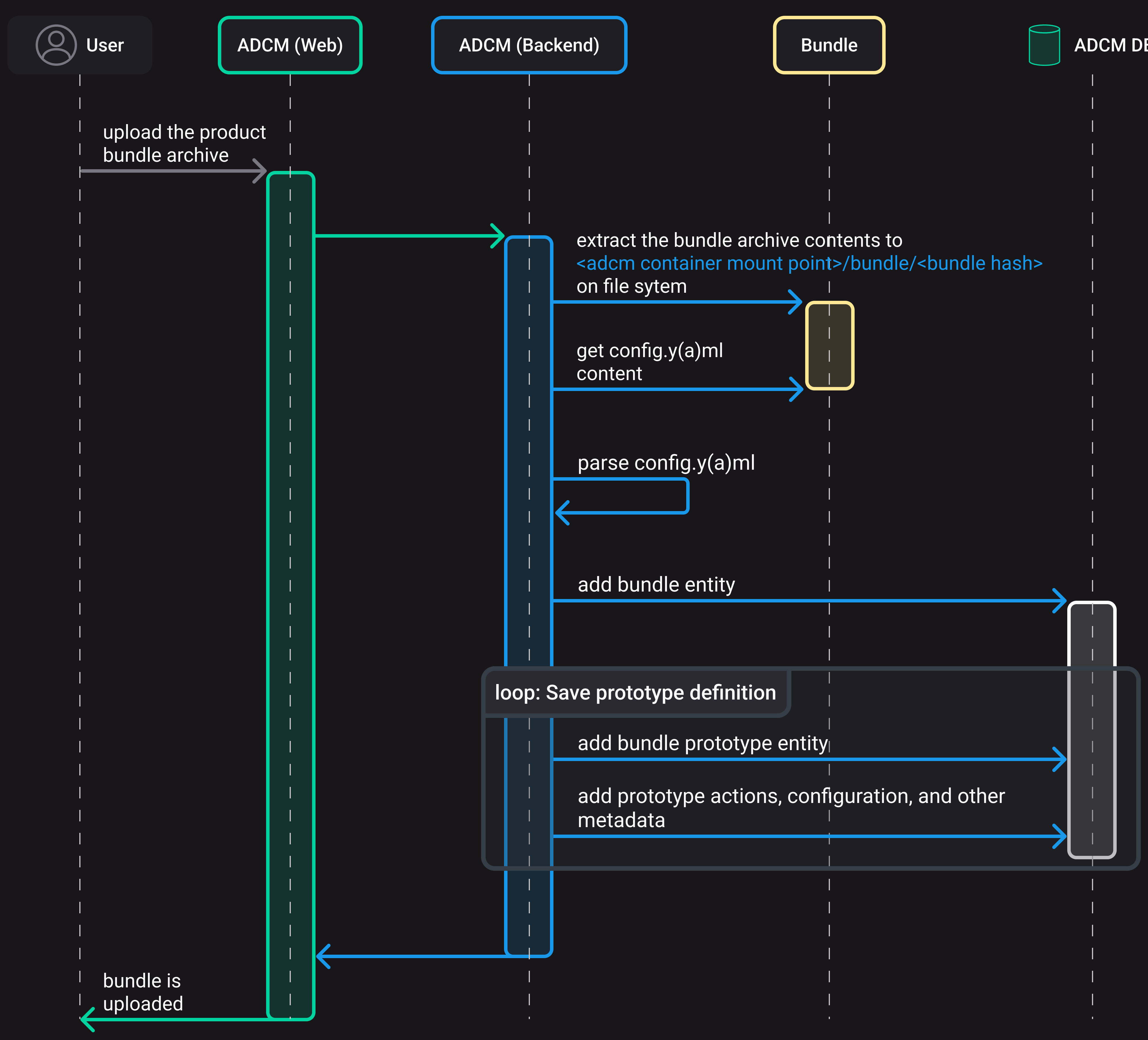
Load the bundle

When the bundle is loaded into ADCM, the backend receives a request that triggers the process described below.

All bundle files are unpacked and put into the file system (*<adcm container mount point>/bundle/<bundle hash>*). Each bundle is unpacked to its corresponding directory and has a unique hash. This allows playbooks and roles of different products not to be mixed up — each playbook is stored inside its corresponding bundle.

After unpacking the bundle, ADCM parses a special file called *config.yaml* or *config.yml*. This file can be located in any directory of the bundle. There may be multiple *config.yaml* files — in which case ADCM finds all files with that name the bundle contains.

After searching for the *config.yaml* files, each of these files undergoes processing. This processing involves creating prototype entities in ADCM based on their descriptions in the *config.yaml* files. This is how metainformation about ADCM objects is created. The backend interacts with the database: all the described prototypes, their actions, configurations, and other related information are written into the database.

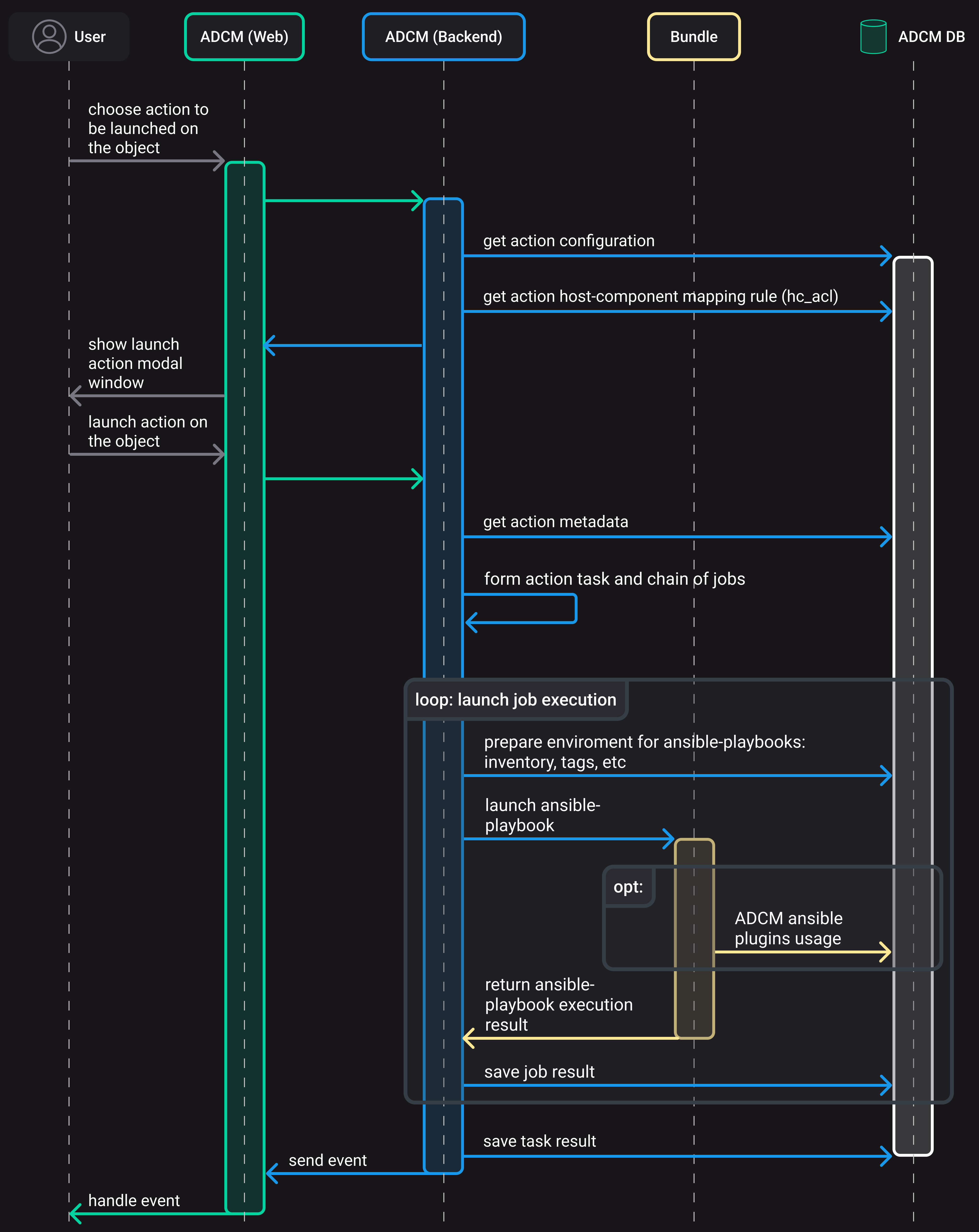

User
ADCM (Web)
ADCM (Backend)
Bundle
ADCM DB
upload the product bundle archive
extract the bundle archive contents to <adcm container mount point>/bundle/<bundle hash> on file sytem
get config.y(a)ml content
parse config.y(a)ml
add bundle entity
loop: Save prototype definition
add bundle prototype entity
add prototype actions, configuration, and other metadata
bundle is uploaded

Sequence diagram for bundle load

Start an action

When starting an action on an ADCM object, metainformation about that action is requested from the database (for example, whether it contains the `config` or `config_jinja` configuration that should be requested from the user before running the action, or whether the components need to be placed on the `hc_acl` host). The backend returns this metainformation, and the UI uses it to generate appropriate forms to interact with the user.

The metainformation from the backend is also used to generate a task that is sent to the task runner for processing, which generates a chain of jobs, each of which is then run by the job runner. In essence, the job runner starts the Ansible playbooks that are specified in the `script` section of the action or subaction. When running, these Ansible playbooks can call Ansible modules and plugins added during the bundle development.


User
ADCM (Web)
ADCM (Backend)
Bundle
ADCM DB
choose action to be launched on the object
get action configuration
get action host-component mapping rule (hc_acl)
show launch action modal window
launch action on the object
get action metadata
form action task and chain of jobs
loop: launch job execution
prepare enviroment for ansible-playbooks: inventory, tags, etc
launch ansible-playbook
opt:
ADCM ansible plugins usage
return ansible-playbook execution result
save job result
save task result
send event
handle event

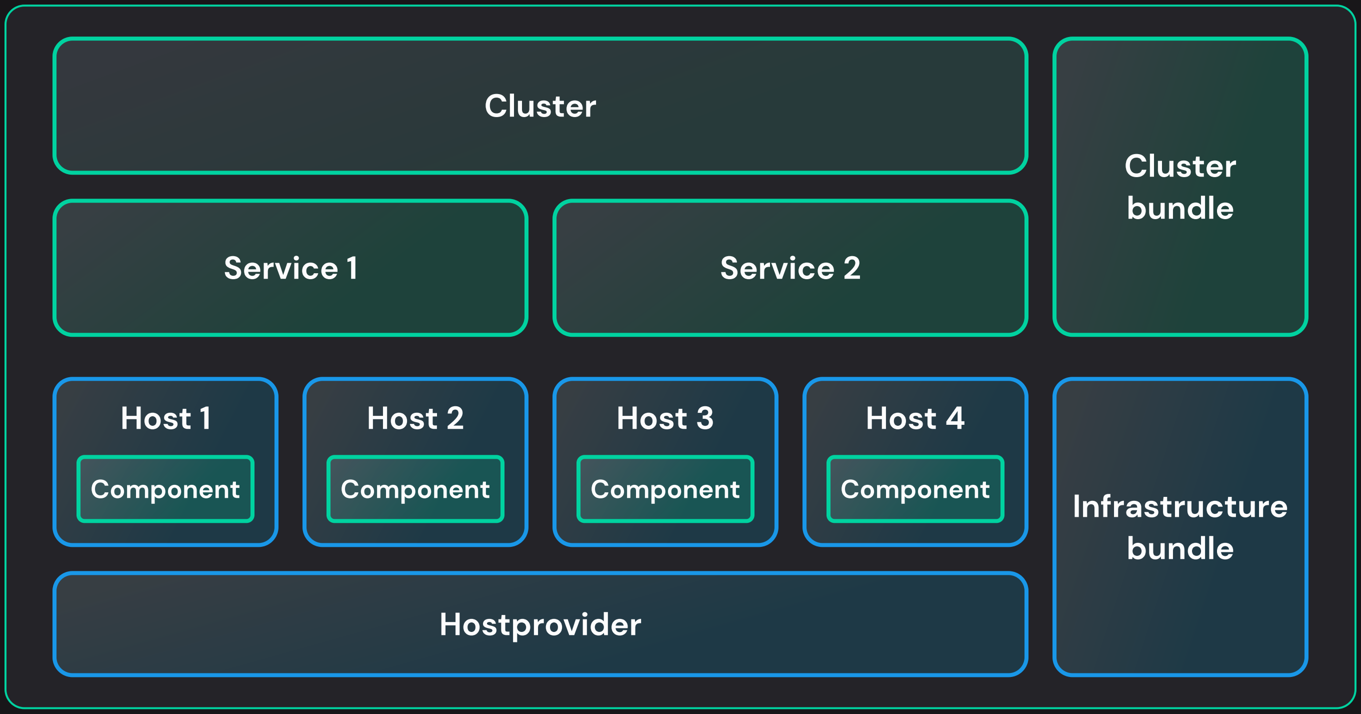
Sequence diagram for action launch

ADCM objects

Mikhail Serov

To Table of Contents

The hierarchy model for internal ADCM objects is presented in the schema below.


Cluster
Cluster bundle
Service 1
Service 2
Host 1
Component
Host 2
Component
Host 3
Component
Host 4
Component
Infrastructure bundle
Hostprovider

ADCM object model

You can find more information on the ADCM objects on the following pages:

- Bundle
- Hostprovider
- Host
- Cluster
- Service
- Component

Bundle

Mikhail Serov

To Table of Contents

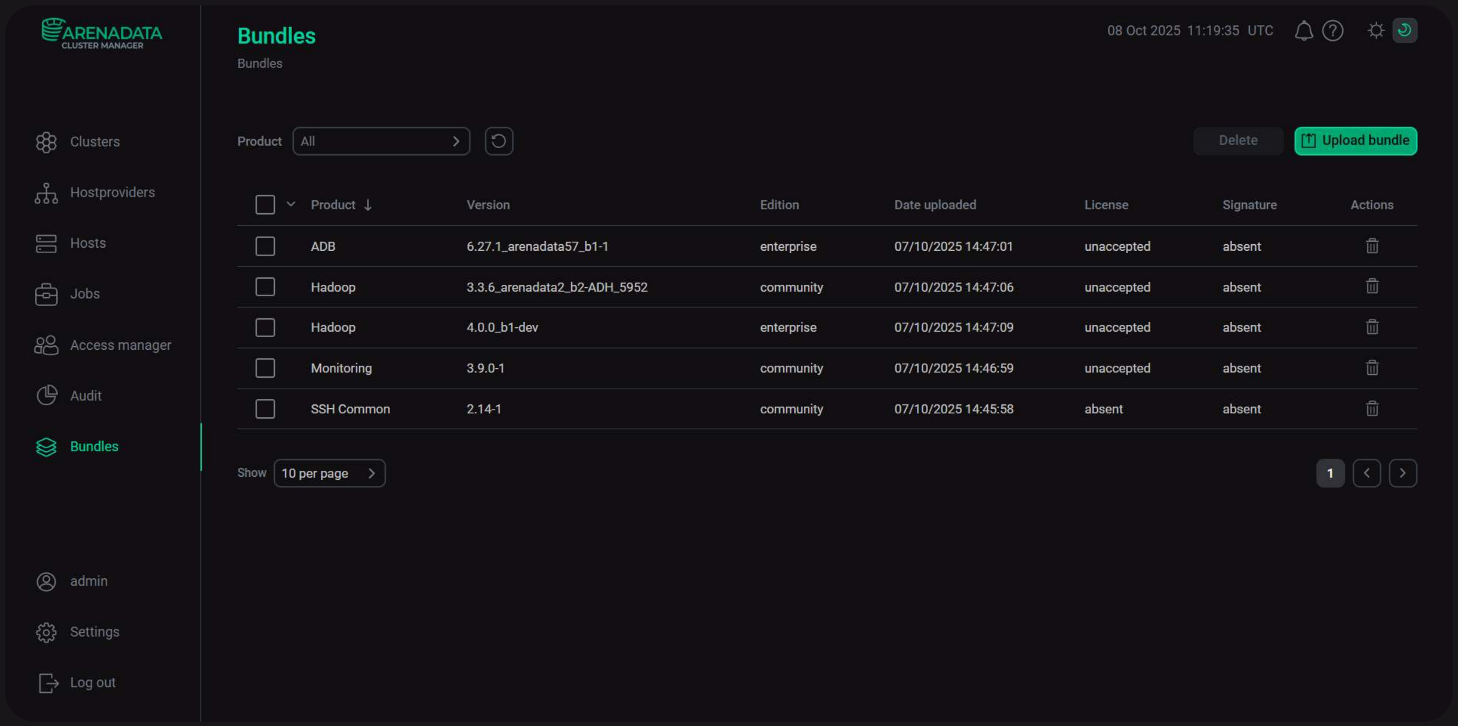
Bundle is a delivery unit in ADCM. New extensions and parts come in bundles. Essentially, a bundle is a tarball that contains information on cluster content, services and components, configuration files, and actions that can be performed on bundle objects.

Bundles come in two types:

- cluster bundle;
- infrastructure bundle.

The cluster bundle allows you to create and manage a cluster, while infrastructure bundle allows you to add support of various hostproviders (for example, Google Cloud, Yandex Cloud, or a SSH hostprovider).

You can see the list of bundles on the ***Bundles*** page.


ARENADATA
CLUSTER MANAGER
Bundles
Bundles
08 Oct 2025 11:19:35 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Product
All
Delete
Upload bundle
Show
10 per page
1

☐	Product ↓	Version	Edition	Date uploaded	License	Signature	Actions
☐	ADB	6.27.1_arenadata57_b1-1	enterprise	07/10/2025 14:47:01	unaccepted	absent	🗑
☐	Hadoop	3.3.6_arenadata2_b2-ADH_5952	community	07/10/2025 14:47:06	unaccepted	absent	🗑
☐	Hadoop	4.0.0_b1-dev	enterprise	07/10/2025 14:47:09	unaccepted	absent	🗑
☐	Monitoring	3.9.0-1	community	07/10/2025 14:46:59	unaccepted	absent	🗑
☐	SSH Common	2.14-1	community	07/10/2025 14:45:58	absent	absent	🗑

The Bundles page

For detailed information about work with bundles in the ADCM UI, see Overview of the Bundles tab in the ADCM UI.

Hostprovider

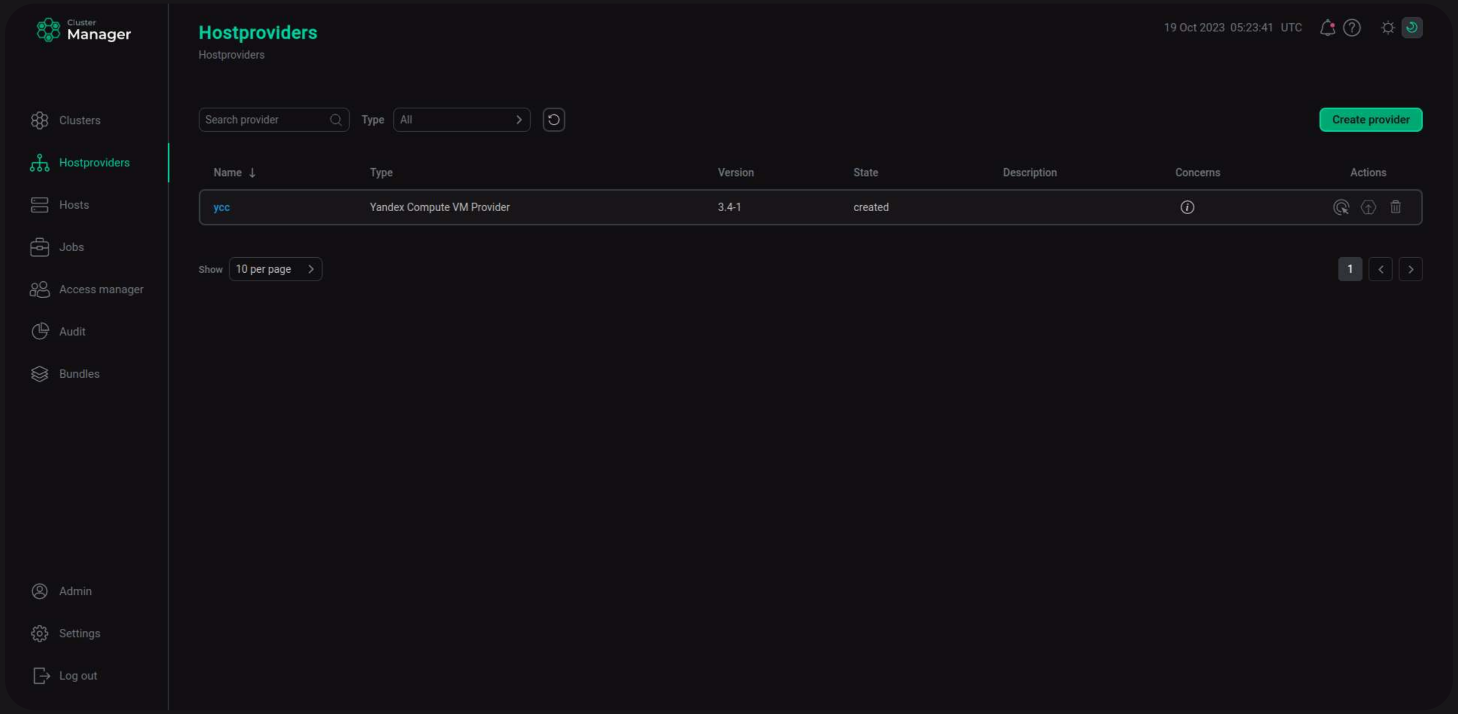
Mikhail Serov

To Table of Contents

Hostprovider is anything that is able to produce a host.

For example, you can create a GCE hostprovider using a Google Cloud account. After you place credentials to the created GCE hostprovider, it allows you to create a host in GCE. You also can create several hosts having the same hostprovider and, consequently, the same credentials.

You can see the list of hostproviders on the ***Hostproviders*** page.


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Admin
Settings
Log out
Hostproviders
Hostproviders
19 Oct 2023 05:23:41 UTC
Search provider
Type
All
Create provider
Show
10 per page
1

Name ↓	Type	Version	State	Description	Concerns	Actions
ycc	Yandex Compute VM Provider	3.4-1	created		ⓘ	

The Hostproviders page

For detailed information about work with hostproviders in the ADCM UI, see Overview of the Hostproviders tab in the ADCM UI.

Host

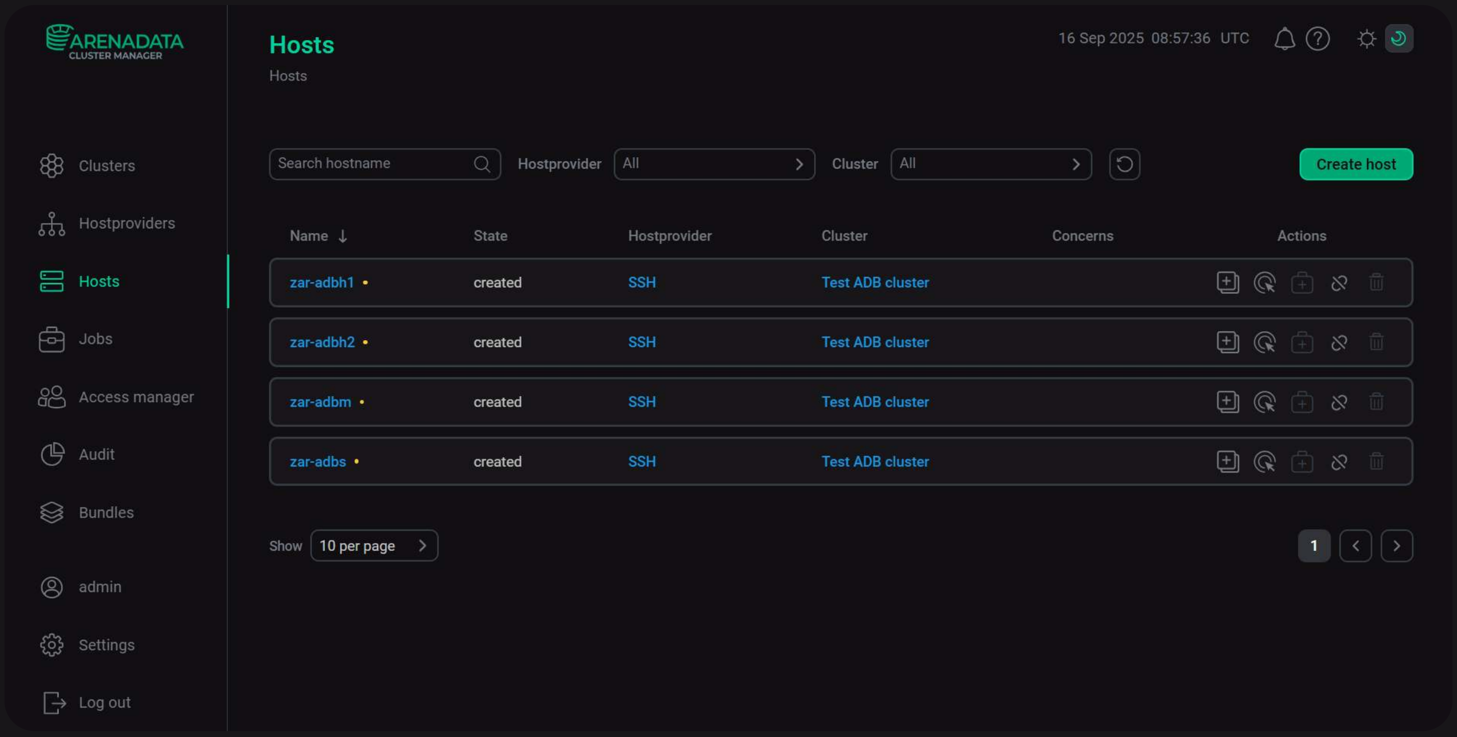
Mikhail Serov, Adelya Zvyagintseva

To Table of Contents

Host is a representation of an actual hardware or virtual host. It does not matter whether the host is a hardware or virtual (cloud) one, yet the host should be accessible via SSH.

Each host can be added only to a cluster that is already created. The way the host is added into ADCM depends on the type of the selected hostprovider.

You can see the list of hosts on the ***Hosts*** page.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Hosts
Hosts
16 Sep 2025 08:57:36 UTC
Search hostname
Hostprovider
All
Cluster
All
Create host
Show
10 per page
1

Name	State	Hostprovider	Cluster	Concerns	Actions
zar-adbh1 •	created	SSH	Test ADB cluster		
zar-adbh2 •	created	SSH	Test ADB cluster		
zar-adbm •	created	SSH	Test ADB cluster		
zar-adbs •	created	SSH	Test ADB cluster		

The Hosts page

For detailed information about work with hosts in the ADCM UI, see Overview of the Hosts tab in the ADCM UI.

Starting with ADCM 2.8.0, it is possible to enable host sharing between multiple clusters that you have created. This can be useful when installing components on the same hosts, if the components are created using different bundles. This operation is performed by creating a **subhost** — a representation of the host in the cluster.

There is a limitation that you need to consider before enabling host sharing. It is forbidden to run actions available to manage the hostprovider.

Cluster

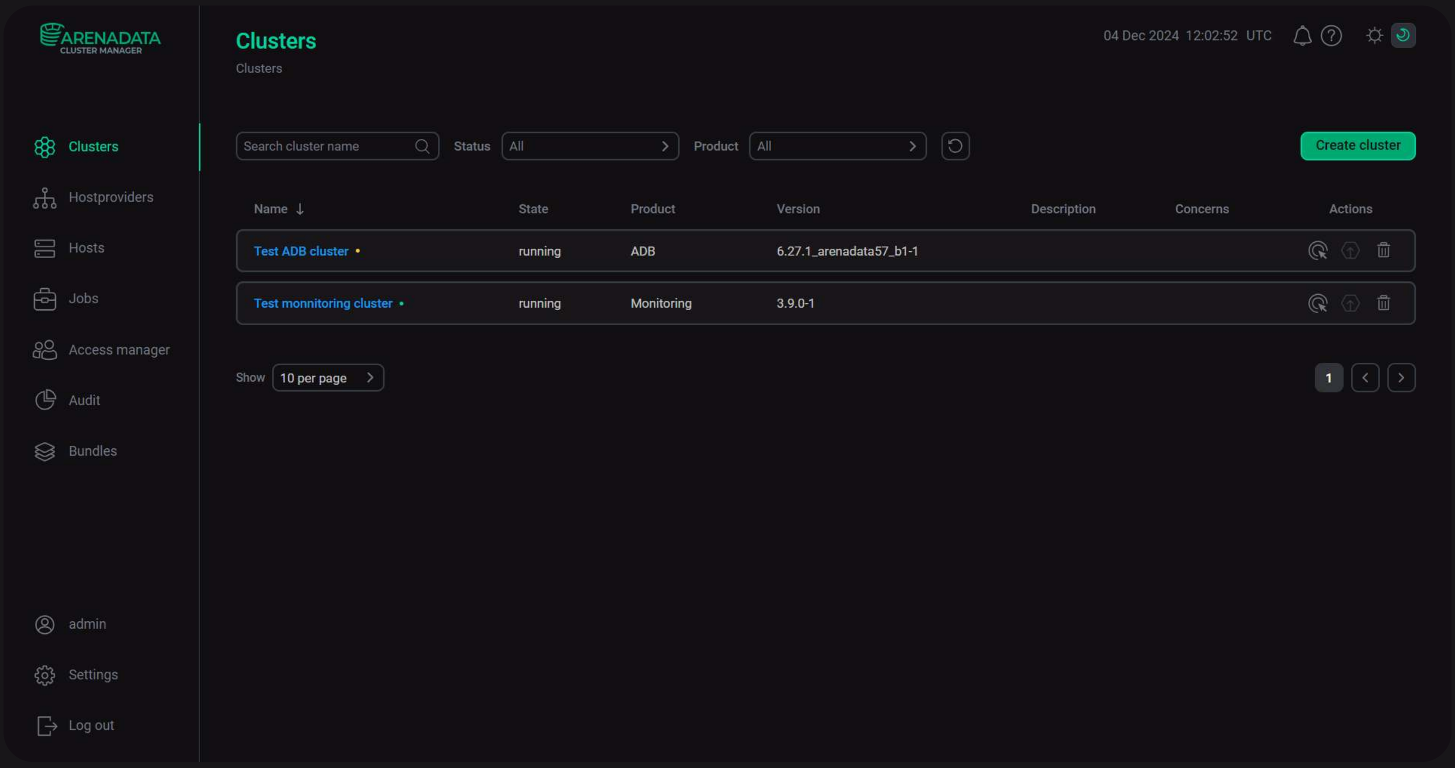
Mikhail Serov

To Table of Contents

Cluster is a set of services running on the set of hosts. ADCM allows you to create a lot of clusters with different configurations and different service distributions.

A single cluster allows only a certain set of services. For example, if you create a Hadoop cluster using a Hadoop bundle, there will be no way to install services from an ADB cluster created using an ADB bundle.

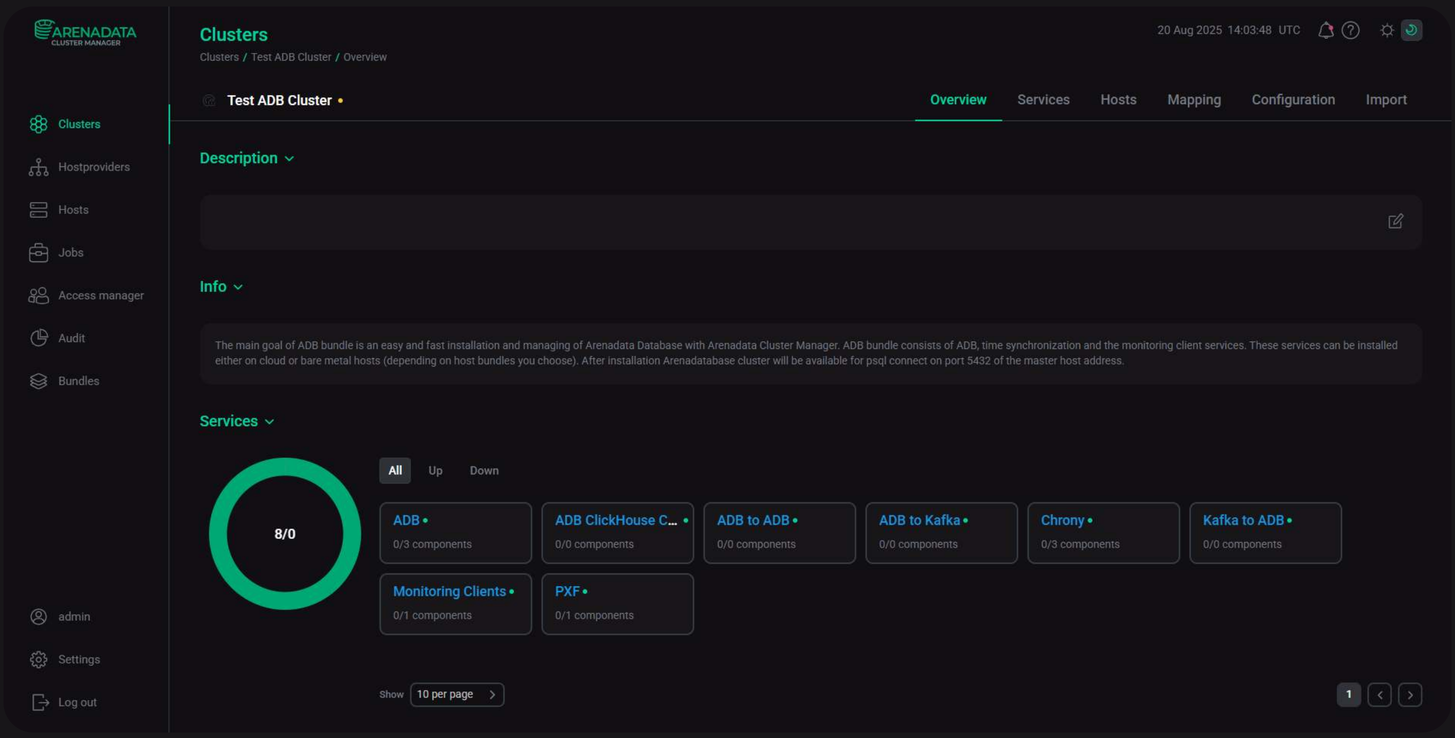
You can see the list of clusters on the ***Clusters*** page.


ARENADATA CLUSTER MANAGER
Clusters
Clusters
04 Dec 2024 12:02:52 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	running	ADB	6.27.1_arenadata57_b1-1			
Test monnitoring cluster	running	Monitoring	3.9.0-1			

The Clusters page

You can see the detailed information about a cluster on the ***Overview*** tab of the cluster page.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADB Cluster / Overview
20 Aug 2025 14:03:48 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Test ADB Cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Description
Info
Services
All
Up
Down
8/0
ADB
0/3 components
ADB ClickHouse C...
0/0 components
ADB to ADB
0/0 components
ADB to Kafka
0/0 components
Chrony
0/3 components
Kafka to ADB
0/0 components
Monitoring Clients
0/1 components
PXF
0/1 components
Show
10 per page
1

The main goal of ADB bundle is an easy and fast installation and managing of Arenadata Database with Arenadata Cluster Manager. ADB bundle consists of ADB, time synchronization and the monitoring client services. These services can be installed either on cloud or bare metal hosts (depending on host bundles you choose). After installation Arenadatabase cluster will be available for psql connect on port 5432 of the master host address.

The Overview tab

For detailed information about work with clusters in the ADCM UI, see Overview of the Clusters tab in the ADCM UI.

Service

Mikhail Serov

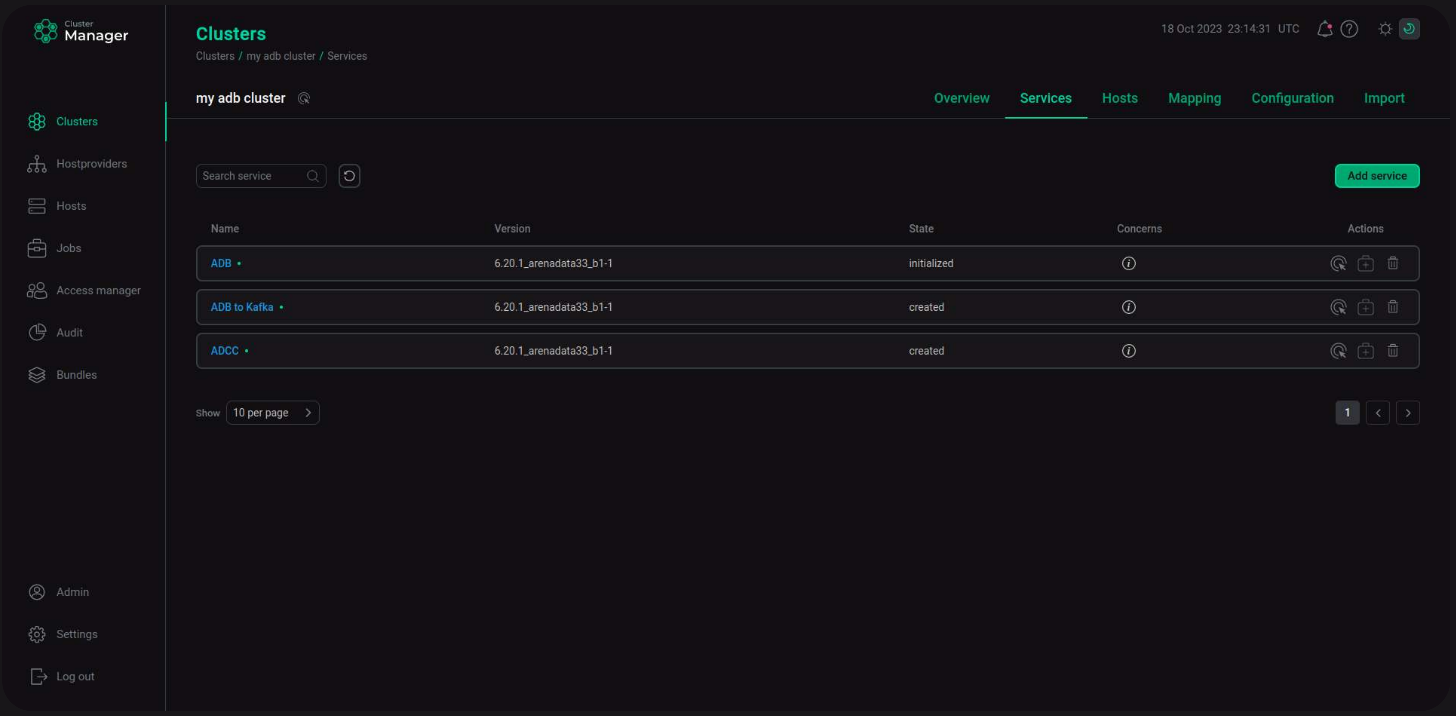
To Table of Contents

Service is a software that functions on hosts. Service can be a single entity or a logical entity that consists of several components (such as HDFS).

Examples of services:

- HDFS
- Hive
- Postgres

The set of services depends on the bundle that is used to deploy a cluster. After the cluster is created, you can choose services on the ***Services*** tab of the cluster page.


Cluster
Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Admin
Settings
Log out
Clusters
Clusters / my adb cluster / Services
18 Oct 2023 23:14:31 UTC
my adb cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Search service
Add service
Show
10 per page
1

Name	Version	State	Concerns	Actions
ADB	6.20.1_arenadata33_b1-1	initialized		
ADB to Kafka	6.20.1_arenadata33_b1-1	created		
ADCC	6.20.1_arenadata33_b1-1	created		

The Services tab

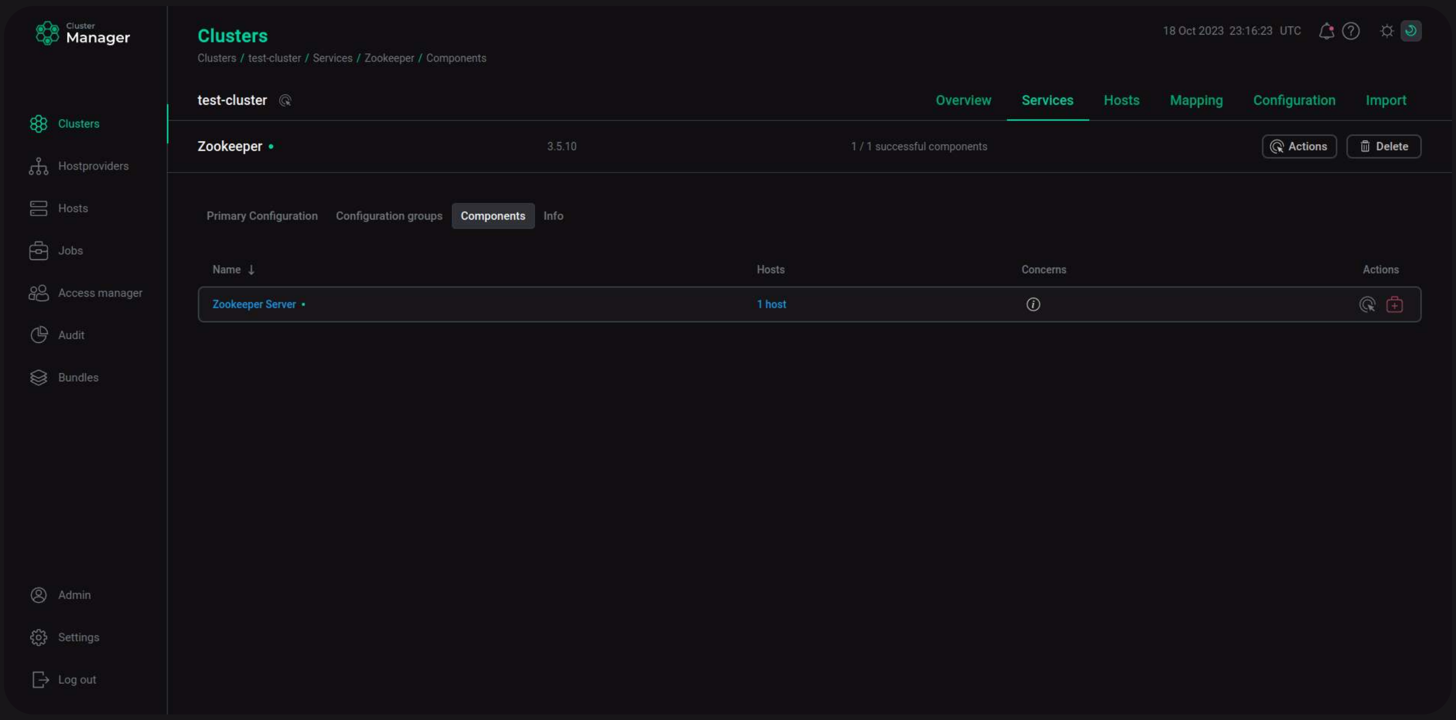
For detailed information about work with services in the ADCM UI, see Overview of the Clusters tab in the ADCM UI.

Component

Mikhail Serov

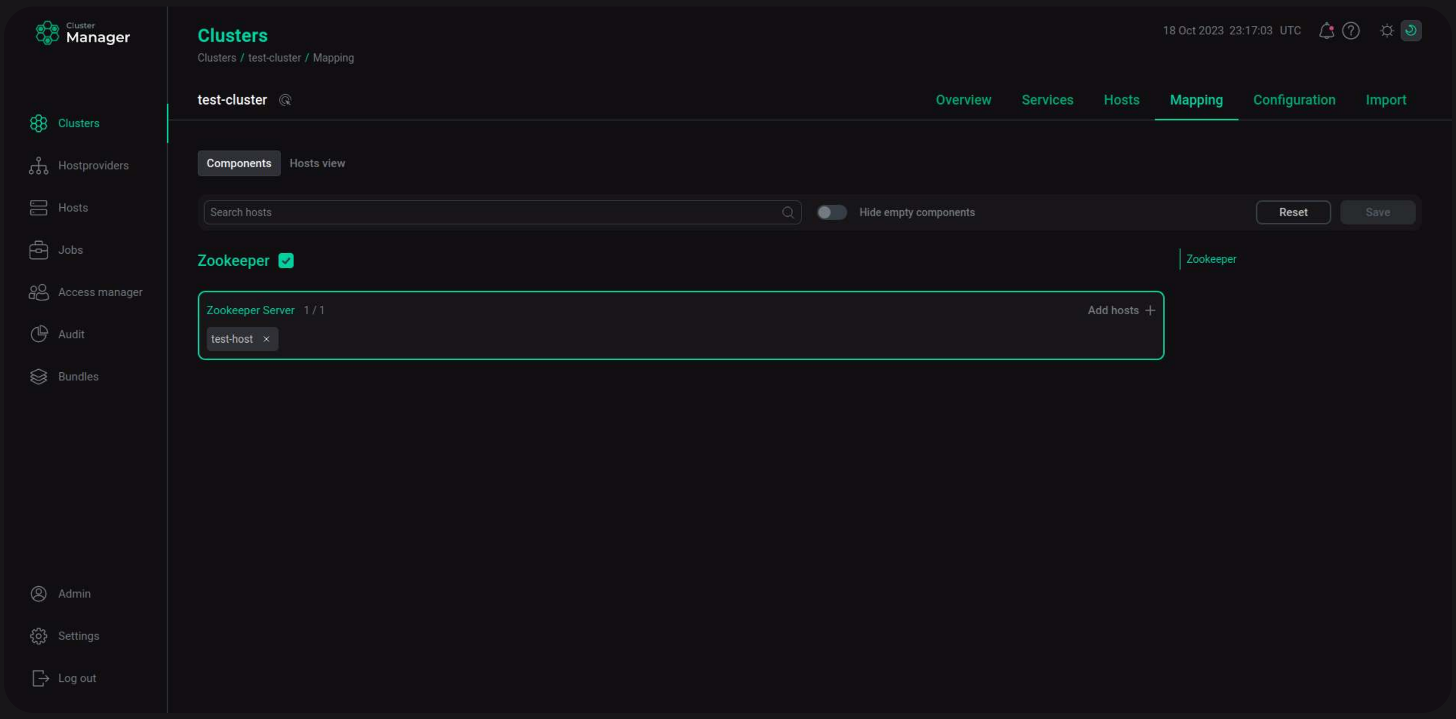
To Table of Contents

Component is a part of a service. For example, the HDFS service consists of the following components: Client, DataNode, JournalNode, HttpFS server, NameNode, and ZKFC. Breaking a service down into components makes it easier to work with the complicated cluster software. You can see the components of a service on the ***Components*** tab of the ***<Cluster name> → Services*** page.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Admin
Settings
Log out
Clusters
Clusters / test-cluster / Services / Zookeeper / Components
18 Oct 2023 23:16:23 UTC
test-cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Zookeeper
3.5.10
1 / 1 successful components
Actions
Delete
Primary Configuration
Configuration groups
Components
Info
Name
Hosts
Concerns
Actions
Zookeeper Server
1 host

The Components tab

The set of components depends on the bundle that is used to deploy a cluster. After the cluster is created and necessary services are added, you can choose components on the ***Mapping*** tab of the cluster page.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Admin
Settings
Log out
Clusters
Clusters / test-cluster / Mapping
18 Oct 2023 23:17:03 UTC
test-cluster
Overview
Services
Hosts
Mapping
Configuration
Import
Components
Hosts view
Search hosts
Hide empty components
Reset
Save
Zookeeper
Zookeeper
Zookeeper Server 1 / 1
Add hosts
test-host

The Mapping tab

For detailed information about work with components in the ADCM UI, see Overview of the Clusters tab in the ADCM UI.

Maintenance mode

Mikhail Serov

To Table of Contents

A **maintenance mode** for services, hosts, and components is implemented in ADCM.

> **IMPORTANT**
>
> Maintenance mode functionality is available only when maintenance mode is supported by the product in use.

In case of hosts, maintenance mode stops the implementation of any cluster- or service-related operations and restricts addition of any components. Maintenance mode is useful if the host is not ready to participate in ADCM actions for some reason (for example, the host is turned off or has malfunctioned).

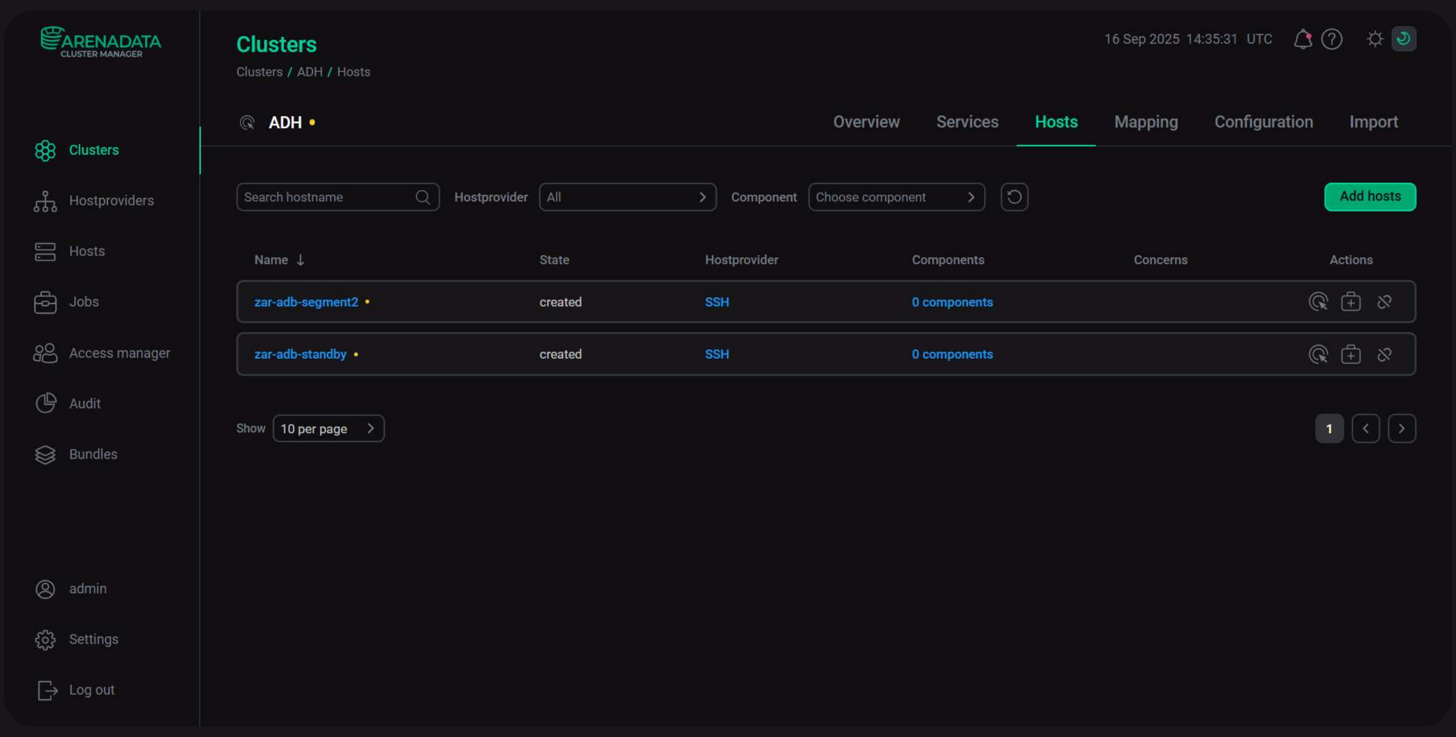
In case of services and components, maintenance mode prevents their participation in composite actions for clusters and services. If a service is in maintenance mode, then it does not participate in cluster actions — the steps that require this service are omitted. If a component is in maintenance mode, then this component and any linked actions are excluded from the service operations.

Consider using maintenance mode when you perform hardware or software maintenance, change configuration settings, perform troubleshooting, decommissioning, or removing cluster nodes.

> **IMPORTANT**
>
> Not every cluster, service, or component action is possible if one of the cluster hosts is in maintenance mode. This depends on whether the action affects all cluster hosts, as well as on action’s logic. At the same time, depending on the product implementation, the event of switching an object to maintenance mode may cause application of additional actions to its host. The result of those actions can be seen on the ***Jobs*** page.

To switch a host to maintenance mode, perform the following steps:

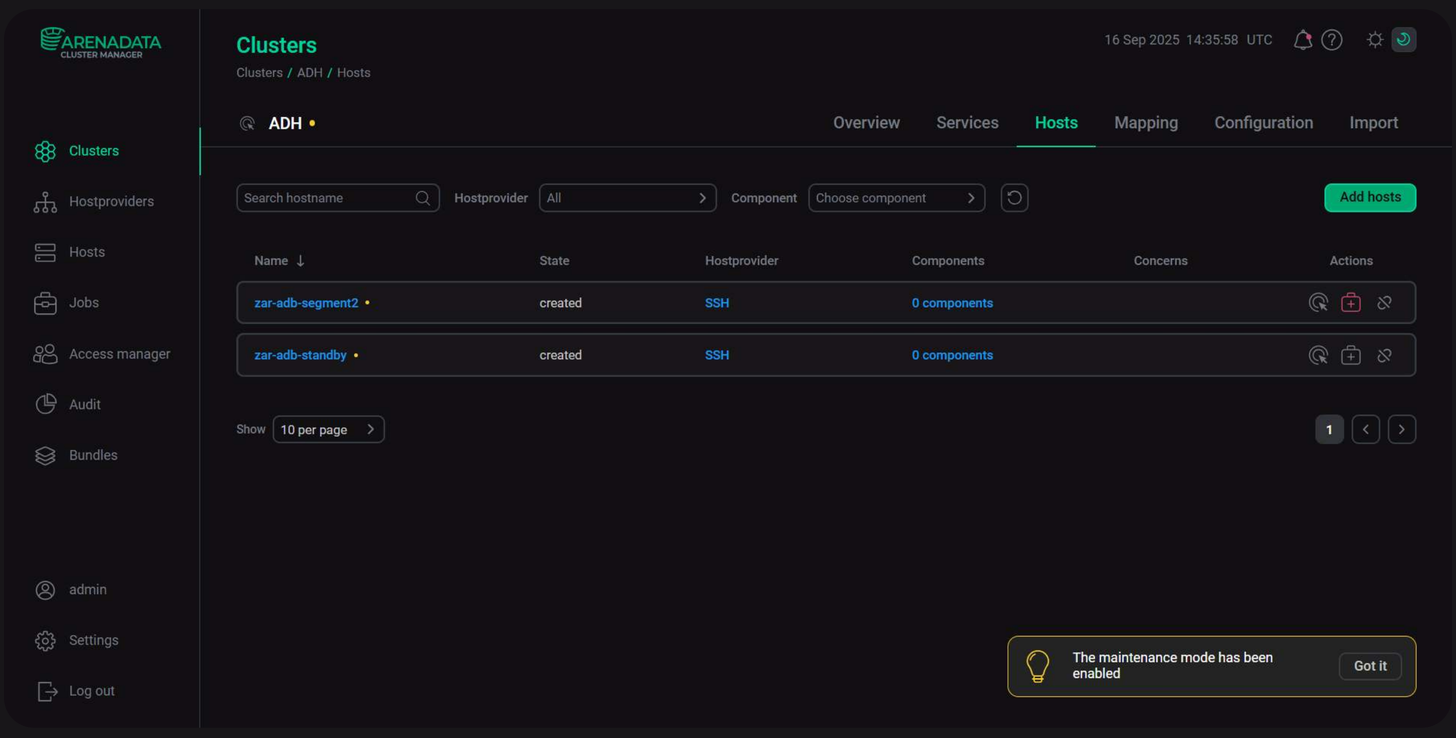
1. Locate the host you wish to switch to maintenance mode in the ADCM interface. You can use the ***Hosts*** tab of the ***Clusters*** page or the ***Hosts*** page.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / ADH / Hosts
16 Sep 2025 14:35:31 UTC
ADH
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
zar-adb-segment2 •	created	SSH	0 components		
zar-adb-standby •	created	SSH	0 components		

Choosing a host

2. Click the icon. The color of the icon will be changed to red — . The host is now in maintenance mode.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Clusters
Clusters / ADH / Hosts
16 Sep 2025 14:35:58 UTC
ADH
Overview
Services
Hosts
Mapping
Configuration
Import
Search hostname
Hostprovider
All
Component
Choose component
Add hosts
Show
10 per page
1
Got it

Name ↓	State	Hostprovider	Components	Concerns	Actions
zar-adb-segment2 •	created	SSH	0 components		
zar-adb-standby •	created	SSH	0 components		

The maintenance mode has been enabled

Switch to the maintenance mode

3. To switch off the maintenance mode, click the icon again.

Components and services are put into maintenance mode in the same way.

> **IMPORTANT**
>
> An object cannot be switched to maintenance mode if a running action is present at that object. For such object, the maintenance mode becomes available after the action is completed.

While you cannot perform any operations with components on a host that is in maintenance mode, you can still remove a component from this host on the logical level. Generally speaking, the maintenance mode prevents bulk operations being performed on a host, yet you may explicitly start and stop a component on a host while in maintenance mode (for example, by connecting via SSH). The statuses of objects in maintenance mode do not affect the statuses of the higher level objects associated with them. For example, if a component is put into maintenance mode and component status is `Failed`, then it does not affect the status of the service that is associated with that component.

Concern

Mikhail Serov, Adelya Zvyagintseva

Contents

To Table of Contents

Types of concerns
  - Flag
  - Lock
  - Issue
Concern owners
Concern prioritization and inheritance
Run actions using the non-blocking mode
Remove non-blocking concerns

In ADCM, some of the user notifications are implemented in the form of **concerns**. Such concern notifications occur, for example, in cases of incorrect configuration, when mandatory parameter fields are not filled in configurations of services, components, or other objects. Another example where ADCM creates a concern is when, after modifying the configuration of an object, the user forgets to apply it to the cluster hosts and continues to work with the outdated configuration.

The causes for concerns, as well as their types and properties, are described in detail below.

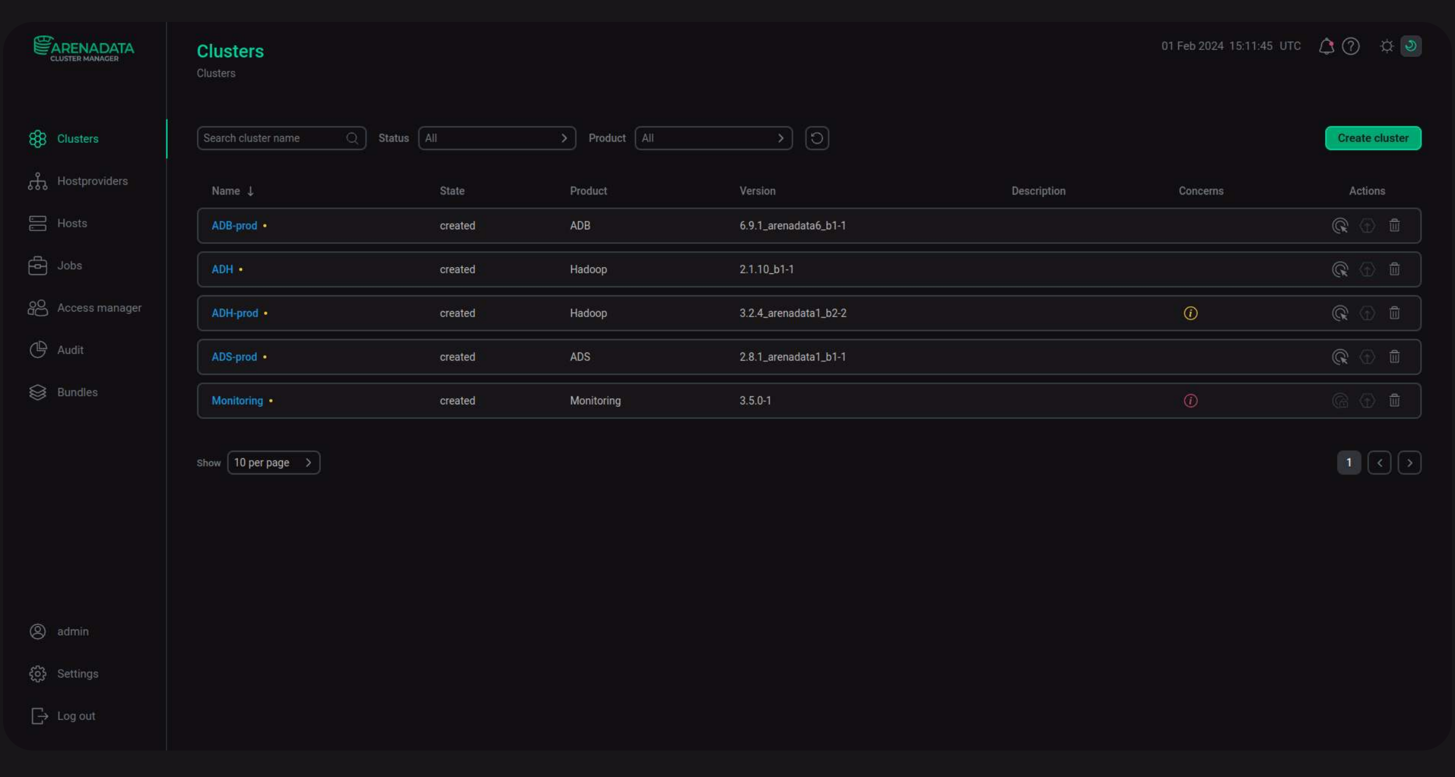
Types of concerns

ADCM has three types of concerns:

- **Flag**
- **Lock**
- **Issue**

Concerns of all types are displayed in the UI.

Concerns can be either **blocking** or **non-blocking**. If a concern is blocking, any actions and updates are unavailable for the associated ADCM object. In the UI, blocking concerns are marked in red, while non-blocking concerns are marked in yellow.



Name	State	Product	Version	Description	Concerns	Actions
ADB-prod	created	ADB	6.9.1_arenadata6_b1-1			
ADH	created	Hadoop	2.1.10_b1-1			
ADH-prod	created	Hadoop	3.2.4_arenadata1_b2-2		ⓘ	
ADS-prod	created	ADS	2.8.1_arenadata1_b1-1			
Monitoring	created	Monitoring	3.5.0-1		ⓘ	

Blocking and non-blocking concerns in the ADCM interface

Let's consider each type separately.

Flag

IMPORTANT

Flag concern functionality is available only when it is supported by the product in use.

Flag is a non-blocking concern that indicates some event.

There are two types of flags:

- embedded
- custom

Custom flags are raised via the `adcm_change_flag` Ansible plugin. Embedded flags are raised every time when the corresponding events occur.

One of the events that might raise a flag is outdated configuration on hosts after changing object parameters in ADCM without subsequent application. To resolve such a problem, it is necessary to apply changes on the cluster hosts by performing the corresponding actions.

The cause of a flag concern is displayed in the UI. In case of the event mentioned above, the following message is displayed: `<Object> has a flag: outdated configuration`.

The purpose of raising a flag concern for the event mentioned is to exclude interaction with objects whose configuration is outdated on the hosts and to provide the flexibility to set the order of actions for interconnected services. For example, changing the configuration of an already installed service may require its restart. In this case, ADCM creates a flag concern, reminding the user of the need for a restart.

Lock

Lock is a blocking concern that occurs each time an action is started on an object. At this point, UI and API elements related to the object become inaccessible.

The cause of a lock concern is displayed in the UI as the following message: `Object was locked by running job <job>`.

The purpose of lock concerns is to prevent simultaneous execution of multiple actions by the user.

Issue

Issue is a blocking concern that can be caused by several reasons:

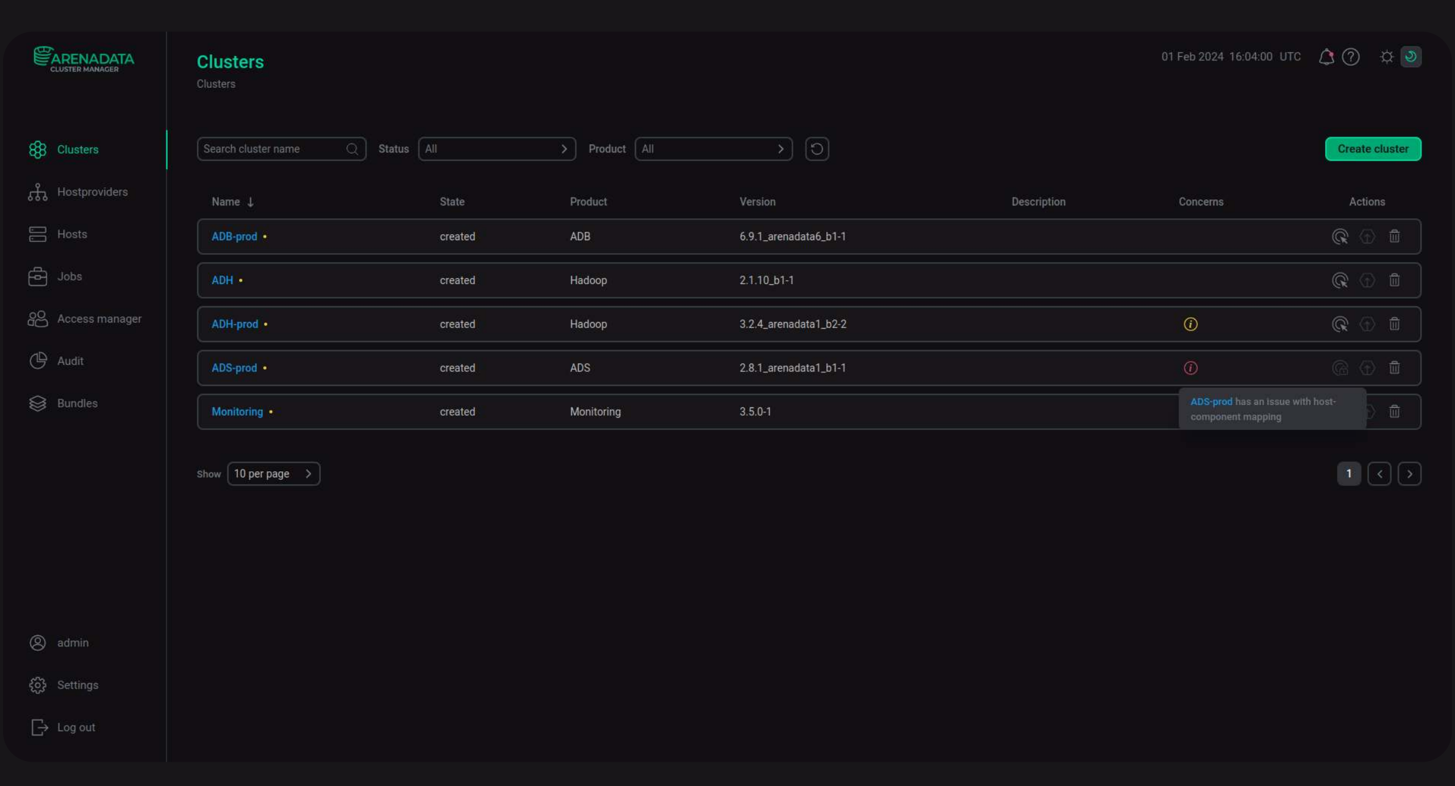
- The object has unfilled or invalid fields in its configuration. This cause is displayed in the UI as the following message: `<Concern owner> has an issue with its config`.
- The object contains a service with mandatory import. This cause is displayed in the UI as the following message: `<Concern owner> has an issue with required import`.
- A required service needs to be installed for the cluster to work. This cause is displayed in the UI as the following message: `<Concern owner> has an issue with required service`.
- The object contains a service that requires another service to be installed for it to work. This cause is displayed in the UI as the following message: `<Concern owner> has an issue with requirement. Need to be installed: <target service>`.
- There is no mapping between hosts and components in the object. This cause is displayed in the UI as the following message: `<Concern owner> has an issue with host-component mapping`.

The purpose of issues is to prevent incorrect interaction and operation of objects.

Concern owners

Concerns have an **owner**. For flag and issue concerns, the owner is an ADCM object affected by this problem.

In the UI, the concern owner is demonstrated in a pop-up message about the cause next to the concern icon.



Name	State	Product	Version	Description	Concerns	Actions
ADB-prod	created	ADB	6.9.1_arenadata6_b1-1			
ADH	created	Hadoop	2.1.10_b1-1			
ADH-prod	created	Hadoop	3.2.4_arenadata1_b2-2		ⓘ	
ADS-prod	created	ADS	2.8.1_arenadata1_b1-1		ⓘ	
Monitoring	created	Monitoring	3.5.0-1			

ADS-prod has an issue with host-component mapping

Concern owner (ADS-prod cluster) in the ADCM interface

For lock concerns, the owner is the corresponding job.

Concern prioritization and inheritance

If an ADCM object has several concerns, one of which is blocking, the blocking concern takes precedence. The icon of the concern is always colored with the color of the higher-priority concern.

Concerns can also be inherited by objects from other objects. For example, if a hostprovider has a concern, it can be inherited by a host created using that hostprovider.

Concerns are inherited in both directions between parent and child objects. Objects are in parent-child relationships if there is an explicit hierarchical link between them. If a service has a concern, that concern is inherited both by the cluster containing that service and by the component contained within that service. However, a service concern is not inherited by hosts if the service's components are not added to the host. Similarly, if a concern arises on a cluster containing a host without components, that cluster concern will not be inherited by the host until at least one component is added to that host. If a concern arises on a host without components, it will not be passed to the cluster containing that host.

Flag concerns for a service do not spread to other services in the cluster. Lock and issue concerns spread from a host to components located on that host, even if the concern owner is another parent object: cluster, service, component, or hostprovider.

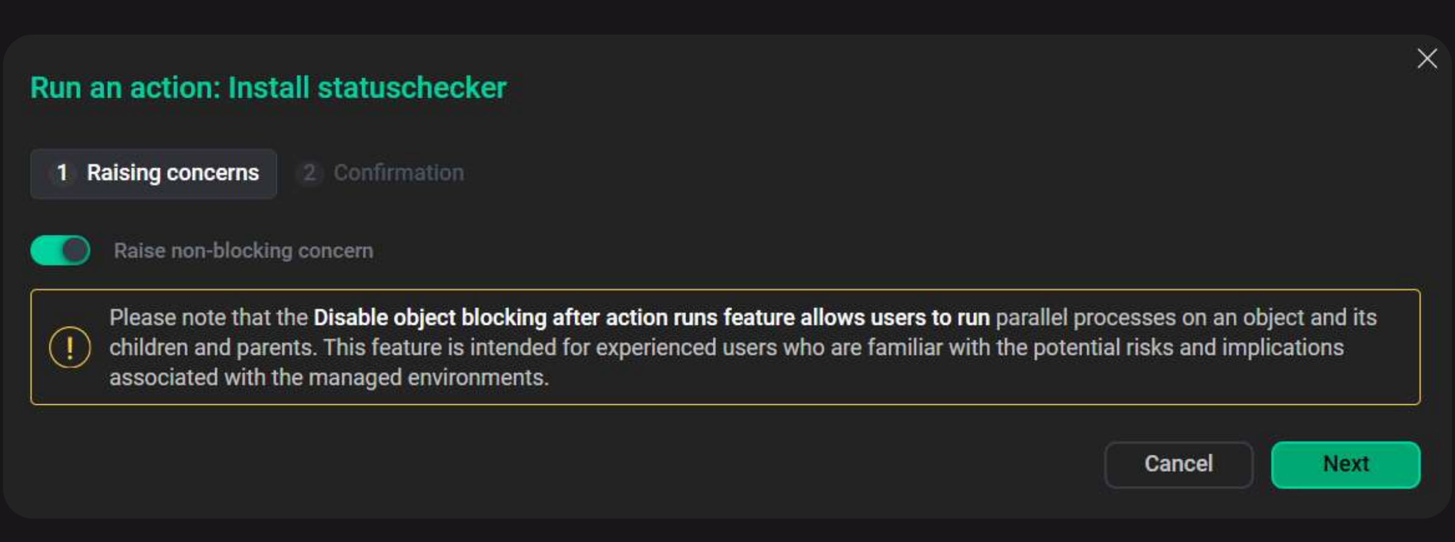
If the concern owner object is deleted, the concerns of all related objects (both parent and child) will also be deleted. For example, if services with concerns are removed from a cluster, those concerns will also disappear from the cluster. However, concerns whose owner is the cluster itself will not disappear.

Run actions using the non-blocking mode

Starting with ADCM 2.5.0, there is a possibility to run actions for ADCM objects without blocking objects. As a result of launching an action, non-blocking concerns might be raised which do not prevent parallel running other actions for this object as well as parent and child objects.

Consider running an action in the non-blocking mode using the example of the ***Install statuschecker*** action:

1. On the ***Hosts*** tab, find the host in the table and click the icon.
2. Select the ***Install statuschecker*** action from the drop-down list.
3. In the dialog box, activate the ***Raise non-blocking concern*** switch.

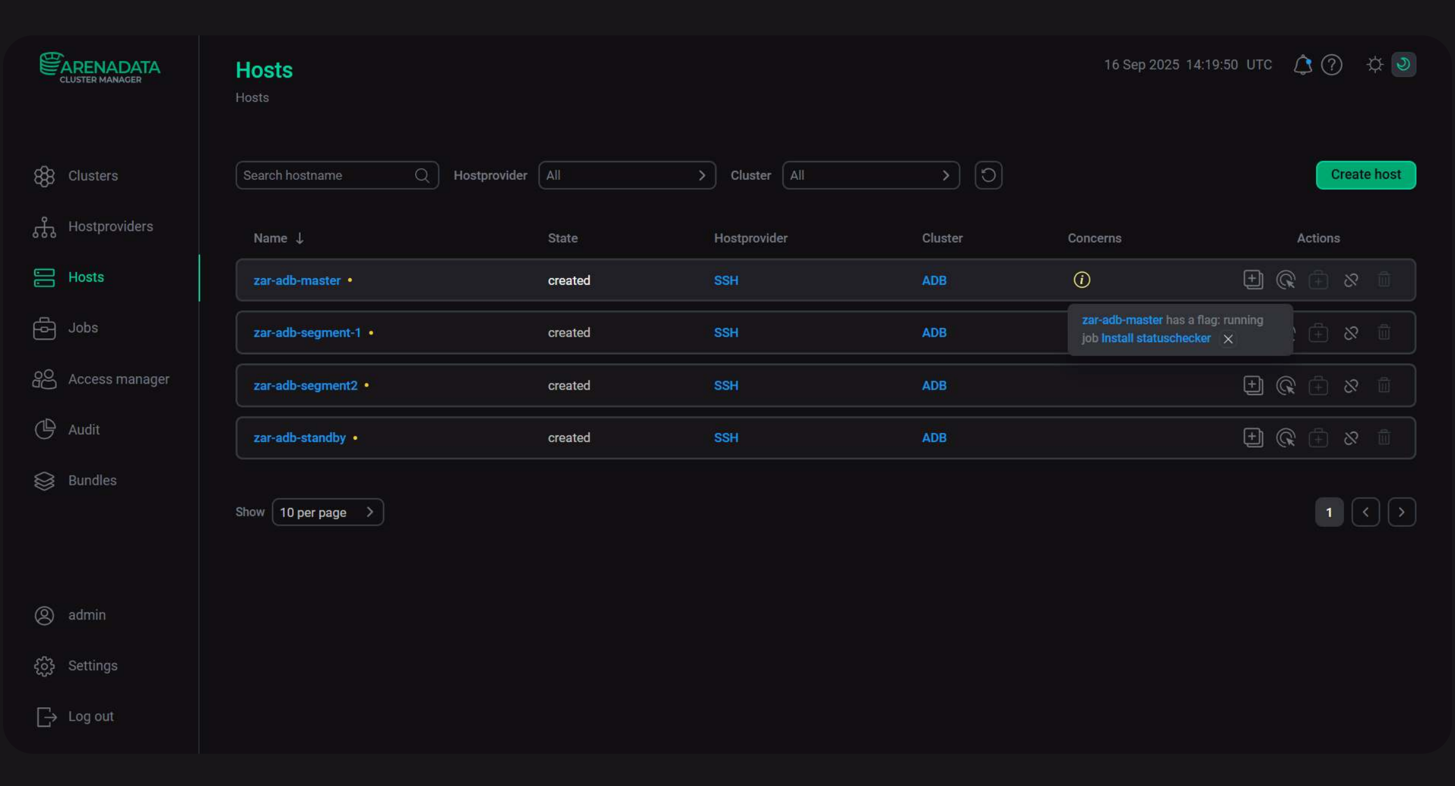


Raise non-blocking concern

Please note that the **Disable object blocking after action runs feature allows users to run** parallel processes on an object and its children and parents. This feature is intended for experienced users who are familiar with the potential risks and implications associated with the managed environments.

Run an action in the non-blocking mode

4. Carefully read the alert message text and click ***Next***.
5. Check the ***Verbose*** checkbox to see additional execution details on the ***Jobs*** page and click ***Run***.
   As an indicator of successfully executed actions, a non-blocking concern such as flag is raised. In the ***Concerns*** column, the ⓘ icon is displayed. A pop-up window with the following error message appears when hovering over the icon: `<Object> has a flag: running job <job>`, where `<Object>` — object name, `<job>` — action name.



Name	State	Hostprovider	Cluster	Concerns	Actions
zar-adb-master	created	SSH	ADB	ⓘ	
zar-adb-segment-1	created	SSH	ADB		
zar-adb-segment2	created	SSH	ADB		
zar-adb-standby	created	SSH	ADB		

zar-adb-master has a flag: running job Install statuschecker

The Hosts page when running an action in the non-blocking mode

Remove non-blocking concerns

ADCM allows users with the `Cluster Administrator`, `Service Administrator`, `Provider Administrator`, and `ADCM Administrator` roles to remove non-blocking concerns on the following tabs:

- ***Clusters***
- ***Services***
- ***Components***
- ***Hostproviders***
- ***Hosts***

The description of these tabs is given in the following articles:

- Overview of the Clusters tab in the ADCM UI
- Overview of the Hostproviders tab in the ADCM UI
- Overview of the Hosts tab in the ADCM UI

To remove an existing non-blocking concern, follow these steps:

1. Select the desired tab.
2. Hover over the ⓘ icon in the list next to the object for which you want to remove the non-blocking concern.
3. In the pop-up window with the error message, click the ✕ icon.

As a result, the ⓘ icon is no longer displayed next to the corresponding object and the following message appears: `The concern has been deleted`.

Implementation specifics

When removing a non-blocking concern, pay attention to the following:

- If a parent object has a non-blocking concern, the same concern will be removed from a child object. This rule also works in the opposite direction.
- If a component has a non-blocking concern, the same concern will be removed from hosts to which the corresponding component has been added.

Prerequisites

Daria Barysheva, Adelya Zvyagintseva

Contents

To Table of Contents

Software and hardware requirements

Install Docker

Disable SELinux

Stop firewalld

Allow access to the Docker CLI without root privileges

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.

Installation

Daria Barysheva, Adelya Zvyagintseva

Contents

To Table of Contents

- Online installation via a Docker Image
- Offline installation (available in the Enterprise Edition)
- Start and stop
- Check installation
  - Check the Docker container status
  - Check the 8000 port availability
  - Check the ADCM web-interface
  - Set the ADCM URL

Online installation via a Docker Image

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

Offline installation (available in the Enterprise Edition)

If you use Enterprise Edition of the product, you can also try offline ADCM installation. It can be useful in situations without Internet access.

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>
```

Start and stop

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.

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
```

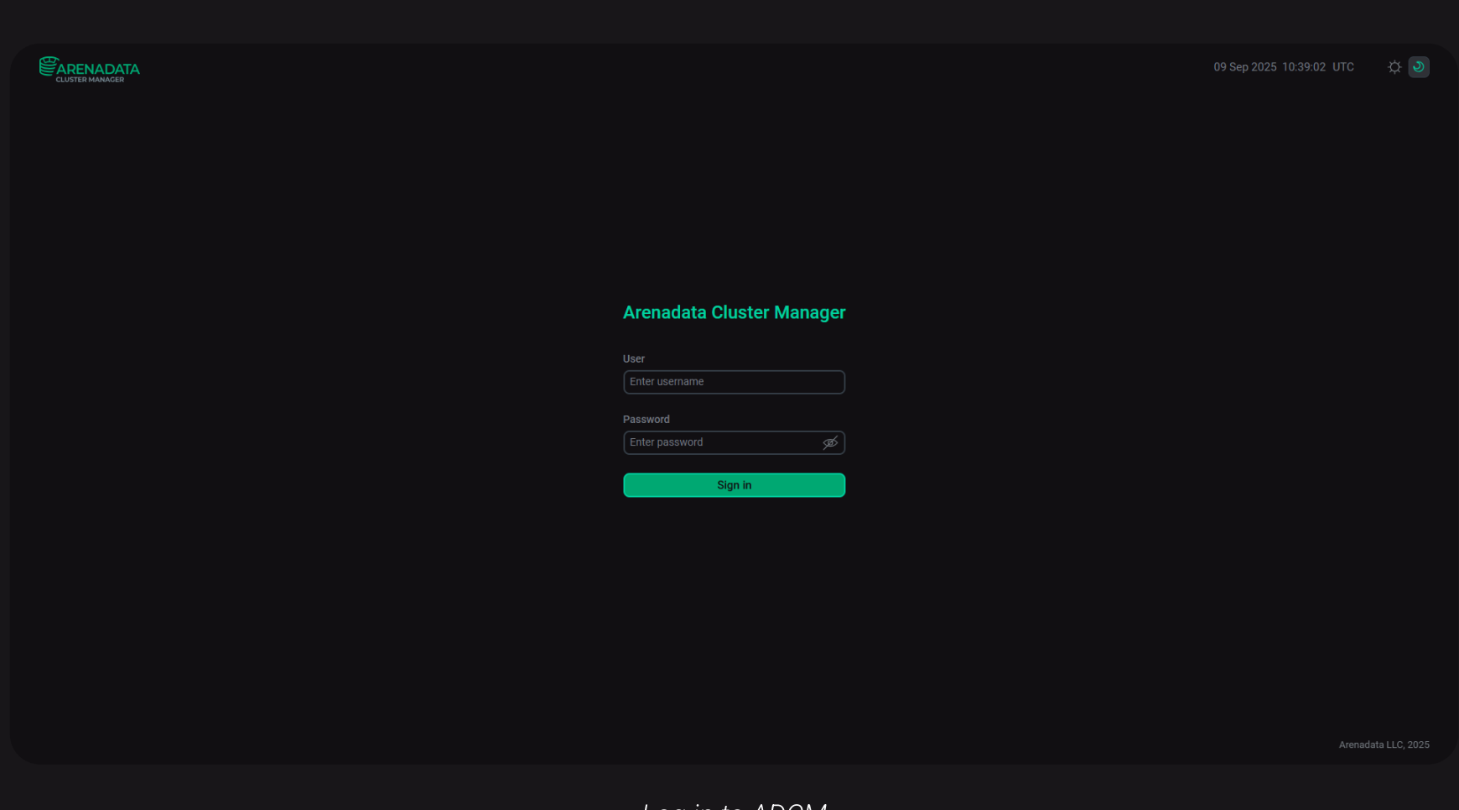
The example of the command execution result:

```
<!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

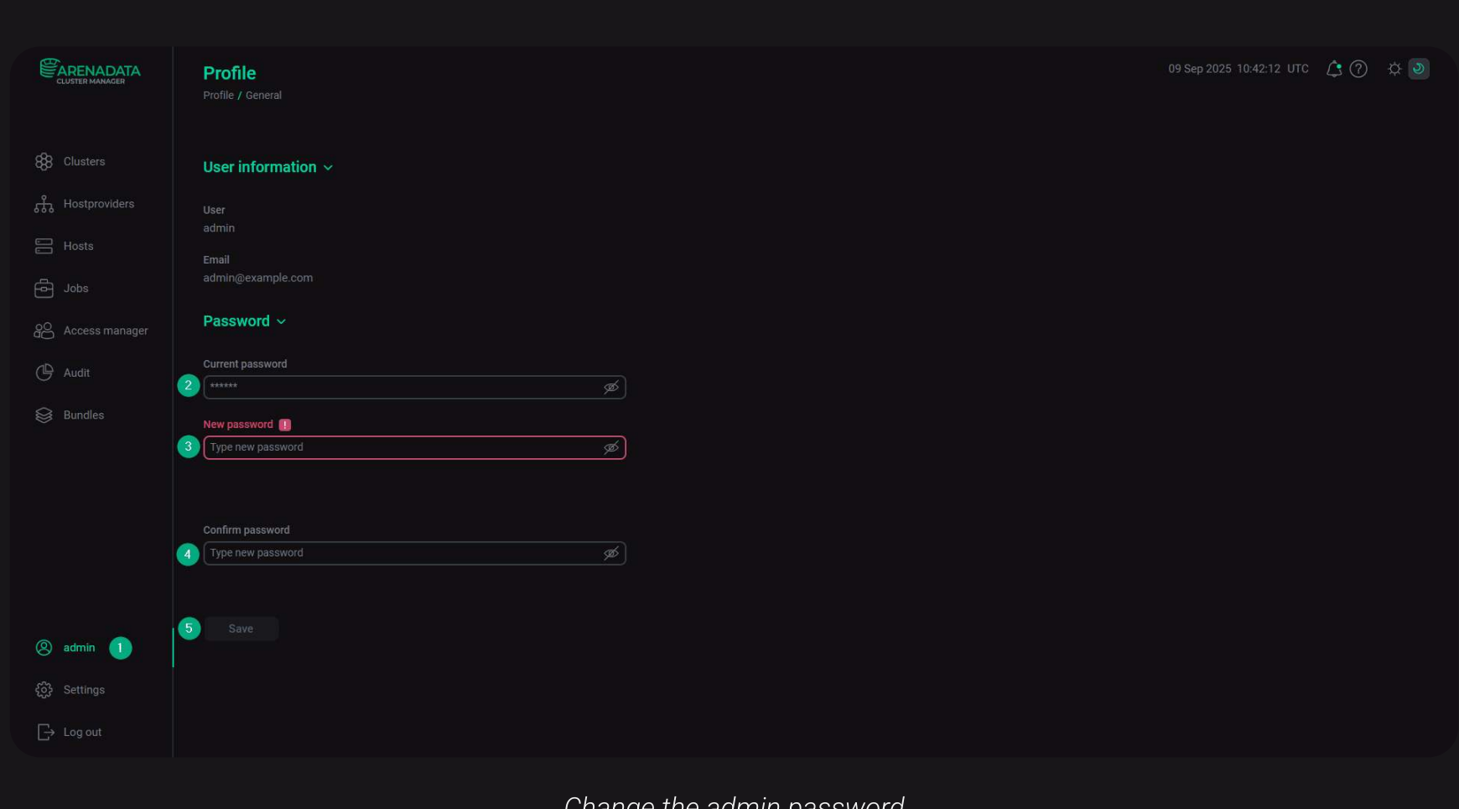


ARENADATA CLUSTER MANAGER
09 Sep 2025 10:39:02 UTC
Arenadata Cluster Manager
User
Enter username
Password
Enter password
Sign in
Arenadata LLC, 2025

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***.



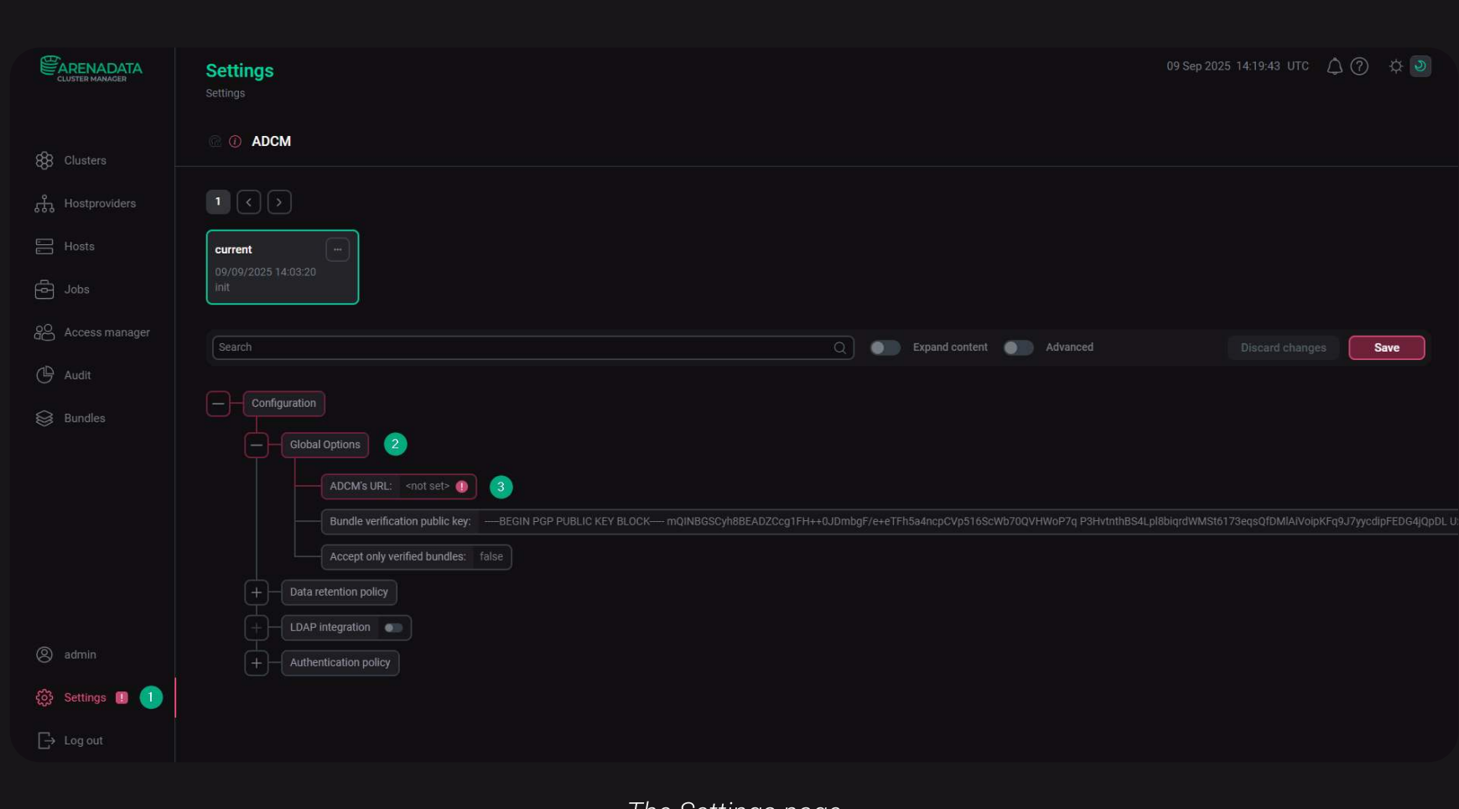
ARENADATA CLUSTER MANAGER
Profile
Profile / General
09 Sep 2025 10:42:12 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
User information
User
admin
Email
admin@example.com
Password
Current password
New password
Type new password
Confirm password
Type new password
Save
admin
Settings
Log out
1
2
3
4
5

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:

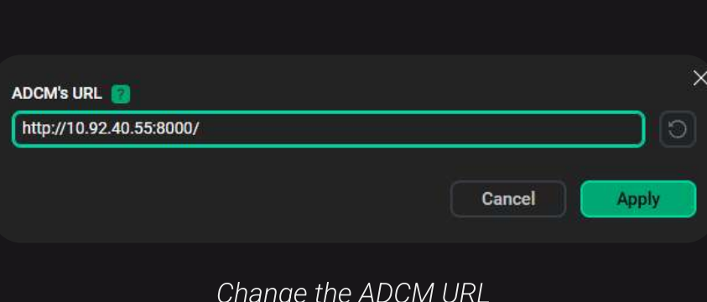
1. Select the ***Settings*** item in the left navigation menu.
2. Expand the ***Global Options*** node in the tree of configuration settings.
3. Click the ***ADCM's URL*** field value.



ARENADATA CLUSTER MANAGER
Settings
09 Sep 2025 14:19:43 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
ADCM
current
09/09/2025 14:03:20
init
Search
Expand content
Advanced
Discard changes
Save
Configuration
Global Options
ADCM's URL: <not set>
Bundle verification public key:
Accept only verified bundles: false
Data retention policy
LDAP integration
Authentication policy
admin
Settings
Log out
1
2
3

The Settings page

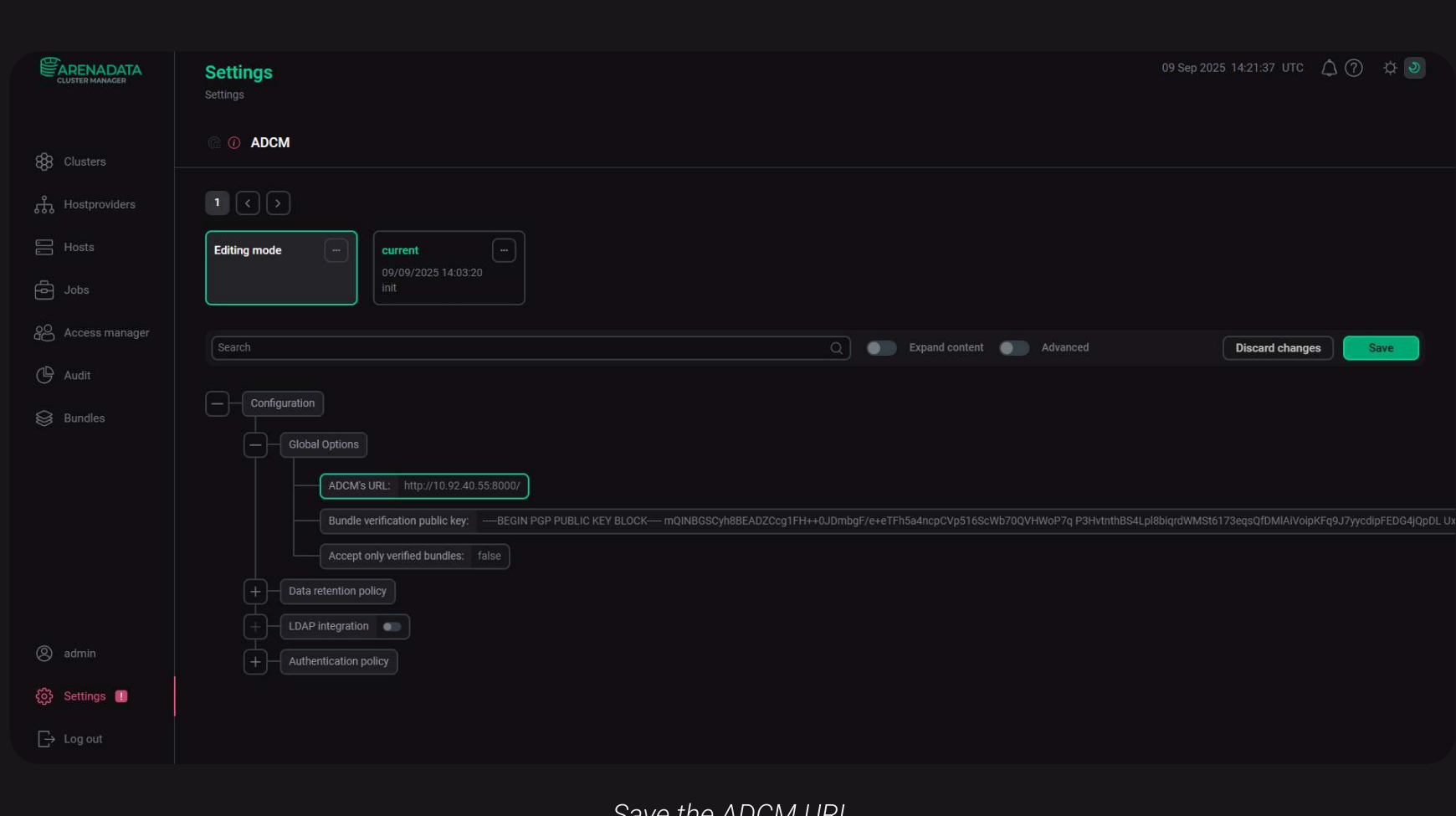
4. In the window that opens, edit the ***ADCM's URL*** field value and click ***Apply***.



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

Change the ADCM URL

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



ARENADATA CLUSTER MANAGER
Settings
09 Sep 2025 14:21:37 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
ADCM
Editing mode
current
09/09/2025 14:03:20
init
Search
Expand content
Advanced
Discard changes
Save
Configuration
Global Options
ADCM's URL: http://10.92.40.55:8000/
Bundle verification public key:
Accept only verified bundles: false
Data retention policy
LDAP integration
Authentication policy
admin
Settings
Log out

Save the ADCM URL

Use ADCM Installer

Adelya Zvyagintseva

Contents

To Table of Contents

- Quick start with ADCM Installer
- Use cases
  - Initialize a configuration file
  - Deploy the ADCM installation
  - Create scripts with auto-complete
  - List running ADCM installations
  - List available ADCM versions
  - Manage secrets
  - Stop and remove running ADCM installations

ADCM Installer is a tool written in Go to help simplify the processes of installation, upgrade, and configuration of the ADCM installations. It also allows you to extend the Docker Compose functionality by secret storage.

ADCM Installer consists of the following packages:

- Docker Compose — controls the life cycle of Docker containers.
- age-encryption — encodes and decodes sensitive data (such as username and password to connect to the database).

For details of possible ADCM Installer use cases, see Use cases.

Quick start with ADCM Installer

NOTE

If you use CentOS, the operating system version must be at least version 8.

Follow the instructions below to install ADCM Installer:

1. Install Docker. For more information, refer to the Docker documentation.
2. Download the *adi* file to the */usr/local/bin* directory and set permissions of the *adi* file as follows:

```
$ sudo wget -O /usr/local/bin/adi https://github.com/arenadata/adcm-installer/releases/latest/download/adi
$ sudo chmod +x /usr/local/bin/adi
```

Wait for the file download to finish.

IMPORTANT

For offline installation, you need to use the URL of the local Container Registry:

```
$ sudo wget -O /usr/local/bin/adi <local_container_registry_url>/arenadata/adcm-
installer/releases/download/<version>/adi
```

To get a brief background on the ADCM Installer commands, use the following command:

```
$ adi --help
```

The expected command execution result:

```
Command line tool for installing Arenadata products

Usage:
  adcm-installer [flags]
  adcm-installer [command]

Available Commands:
  adcm-versions List versions of Arenadata products
  apply         Apply a configuration by file name
  completion    Generate the autocompletion script for the specified shell
  components    Manage installed components
  delete        Delete resources by file name or installation name
  help          Help about any command
  init          Initialize a new configuration
  list          List running ADCM installation
  secrets       Manage secrets

Flags:
  -h, --help      help for adcm-installer
      --version   Print the version and exit
```

Use cases

ADCM Installer supports the following commands:

- `init`
- `apply`
- `completion`
- `list`
- `adcm-versions`
- `secrets`
- `delete`

The command descriptions and examples of their usage are given below.

Initialize a configuration file

The `init` command generates the configuration file described below, which is necessary for the ADCM installation deployment.

For example, to initialize the configuration file of the ADCM installation using PostgreSQL as a database, use the following command with the `--adpg` flag:

```
$ adi init <installation_name> --adpg
```

where `<installation_name>` is the name of the ADCM installation such as `my-adcm`, that includes the Docker containers: `my-adcm-adpg` and `my-adcm-adcm`.

To specify the database credentials and configure the database connection settings, use the interactive mode by specifying the `-i` or `--interactive` flag:

```
$ adi init <installation_name> --adpg -i
```

NOTE

Without the `--adpg` flag, settings for connection to an external PostgreSQL database will be requested.

Database connection parameters can also be specified in a file using the `--from-config` flag.

Parameters that can be specified in the file are presented in the table below. The table also describes a correlation between parameters that can be configured in the interactive mode and parameters that can be specified in the file.

Parameters

Parameter used in the file	Parameter displayed in interactive mode	Type	Description	Default value
adcm-count	Number of ADCM instances	uint8	Number of the ADCM instances. The parameter is available in ADCM Installer starting with version 1.1	1
adcm-image	ADCM image	string	ADCM container image repository (in the above example — `my-adcm-adcm`)	hub.arenadata.io/adcm/adcm
adcm-tag	ADCM image tag	string	Docker image tag used for the ADCM container (in the above example — `my-adcm-adcm`)	2.7.1
adcm-publish-port	ADCM publish port	uint16	Port of localhost on which the ADCM container runs (in the above example — `my-adcm-adcm`)	If you have one ADCM instance, the default value is `8000`. If you have several ADCM instances, the amount of the increment for each instance is `1`
adcm-db-host	ADCM database host	string	ADCM database server address. This parameter is ignored when specifying the `--adpg` flag in the interactive mode	—
adcm-db-port	ADCM database port	uint16	Port number for connecting to the ADCM database. This parameter is ignored when specifying the `--adpg` flag in the interactive mode	5432
adcm-db-name	ADCM database name	string	Name of the ADCM database	If you have one ADCM instance, the default value is `adcm`. If you have several ADCM instances, the default value is `adcm_<instance_serial_number>`
adcm-db-user	ADCM database user	string	Username for connecting to the ADCM database	If you have one ADCM instance, the default value is `adcm`. If you have several ADCM instances, the default value is `adcm_<instance_serial_number>`
adcm-db-pass	ADCM database password	string	Password for connecting to the ADCM database	—
adcm-db-ssl-mode	Postgres SSL mode	string	Flag used to connect to the ADCM database for establishing a TLS connection. This parameter is ignored when specifying the `--adpg` flag in the interactive mode. Possible values: - `disable` - `allow` - `prefer` - `require` - `verify ca` - `verify full` The description of each value is given in the Installation article	disable
adcm-db-ssl-ca-file	ADCM database SSL CA file path	string	Path to the certificate authority (CA) certificate file for establishing a TLS connection to the ADCM database	—
adcm-db-ssl-cert-file	ADCM database SSL certificate file path	string	Path to the certificate file in PEM format for establishing a TLS connection to the ADCM database	—
adcm-db-ssl-key-file	ADCM database SSL private key file path	string	Path to the private key file in PEM format for establishing a TLS connection to the ADCM database	—
adcm-url	ADCM url	uint16	URL of the host where the ADCM container is deployed. Use the following format: `http://<ip_address>:<port>/`	If you have one ADCM instance, the default value is `http://<ip_address>:8000/`. If you have several ADCM instances, port number for each instance increases by `1`. For example, for the first instance, the default value is `http://<ip_address>:8001/`
adcm-volume	ADCM volume name or path	string	Name of the ADCM container volume (in the above example — `my-adcm-adcm`)	<installation_name>-adcm-<instance_serial_number>
adcm-publish-ssl-port	ADCM publish SSL port	uint16	Localhost port for the ADCM container (in the above example — `my-adcm-adcm`) to establish TLS connection	8443
adcm-ssl-cert-file	ADCM SSL Certificate file path	string	Path to the certificate file in PEM format for establishing a TLS connection to ADCM	—
adcm-ssl-key-file	ADCM SSL Private Key file path	string	Path to the private key file in PEM format for establishing a TLS connection to ADCM	—
adpg-pass	ADPG superuser password	string	Password of the superuser role which is set during the PostgreSQL database initialization. The parameter is available when specifying the `--adpg` flag in the interactive mode	—
adpg-image	ADPG image	string	ADPG container image repository (in the above example — `my-adcm-adpg`). The parameter is available when specifying the `--adpg` flag in the interactive mode	hub.arenadata.io/adcm/postgres
adpg-tag	ADPG image tag	string	Docker image tag used for the ADPG container (in the above example — `my-adcm-adpg`). The parameter is available when specifying the `--adpg` flag in the interactive mode	v16.3.1
adpg-publish-port	ADPG publish port	uint16	Port of localhost on which the ADPG container runs (in the above example — `my-adcm-adpg`). The parameter is available when specifying the `--adpg` flag in the interactive mode	5432
adpg-volume	ADPG volume name or path	string	Name of the ADPG container volume (in the above example — `my-adcm-adpg`). The parameter is available when specifying the `--adpg` flag in the interactive mode	<installation_name>-adpg
consul-image	Consul image	string	Consul container image repository. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--consul` flag)	hub.arenadata.io/adcm/consul
consul-tag	Consul image tag	string	Docker image tag used for the Consul container. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--consul` flag)	v0.0.0
consul-publish-port	Consul publish port	uint16	Port of localhost on which the Consul container runs. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--consul` flag)	8500
vault-mode	Vault Deployment mode	boolean	Flag used to select the mode in which OpenBao will be launched. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag). Possible values: - `non-ha` — persistent mode; - `ha` — high availability (HA) mode; - `dev` — development mode. Description of the modes is given below	non-ha
vault-ui	Vault enable UI	boolean	Flag used to enable access to the OpenBao web interface (for example, to create or update secrets). The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag). Possible values: - `true` — enebles access to the web interface; - `false` — disables access to the web interface.	true
vault-image	Vault image	string	OpenBao container image repository. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	openbao/openbao
vault-tag	Vault image tag	string	Docker image tag used for the OpenBao container. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	2.2.0
vault-publish-port	Vault publish port	uint16	Port of localhost on which the OpenBao container runs. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	8200
vault-db-host	Vault database host	string	OpenBao database server address. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	1
vault-db-port	Vault database port	uint16	Port number for connecting to the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	5432
vault-db-name	Vault database name	string	Name of the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	vault
vault-db-user	Vault database user	string	Username for connecting to the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	vault
vault-db-pass	Vault database password	string	Password for connecting to the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	—
vault-db-ssl-mode	Postgres SSL mode	string	Flag used to connect to the OpenBao database for establishing a TLS connection. The parameter is available when specifying the `--vault` flag in the interactive mode. Possible values: - `disable` - `allow` - `prefer` - `require` - `verify ca` - `verify full`	disable
vault-db-ssl-ca-file	Vault database SSL CA file path	string	Path to the certificate authority (CA) certificate file for establishing a TLS connection to the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	—
vault-db-ssl-cert-file	Vault database SSL certificate file path	string	Path to the certificate file in PEM format for establishing a TLS connection to the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	—
vault-db-ssl-key-file	Vault database SSL private key file path	string	Path to the private key file in PEM format for establishing a TLS connection to the OpenBao database. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	—
vault-ssl-cert-file	Vault SSL Certificate file path	string	Path to the certificate file in PEM format for establishing a TLS connection to OpenBao. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	—
vault-ssl-key-file	Vault SSL Private Key file path	string	Path to the private key file in PEM format for establishing a TLS connection to OpenBao. The parameter is available in ADCM Installer starting with version 1.1 (in the interactive mode, you need to specify the `--vault` flag)	—

If parameter values are not specified, default values will be used for names and Docker container configuration parameters and passwords will be generated automatically.

After the command mentioned above is executed, the following files will be created:

- *adcm.yaml* (or *adcm.yml*) is the ADCM installation configuration file in the YAML format;
- *age.key* is the file with the private key in the `AGE-SECRET-KEY-1…` format to access sensitive data (secrets) stored in the *adcm.yaml* (or *adcm.yml*) file.

To take advantage of additional command options, use the following flags:

- `--force` — allows you to overwrite the existing configuration file;
- `-o` or `--output` — allows you to save the configuration file with the given name, for example: `--output /<file_path>`, where `<file_path>` is the absolute path to the configuration file in the YAML format including the file name;
- `--age-key-file` — determines the location of the file with the private key;
- `--age-key` — determines the private key, for example:

```
$ adi init <installation_name> --age-key AGE-SECRET-KEY-1...
```

NOTE

The private key can also be passed via the `AGE_KEY` variable, for example: `export AGE_KEY=AGE-SECRET-KEY-1…`.

Example of the configuration file for deploying ADCM with a PostgreSQL database

```
# Project name
name: my-adcm
# Configurations of the following Docker containers: my-adcm-adcm and my-adcm-adpg
services:
  adcm:
    # Add specified Linux capabilities to the Docker container. The list of supported capabilities is given in the Linux
documentation
    cap_add:
      - CAP_CHOWN
      - CAP_SETUID
      - CAP_SETGID
    # Remove specified Linux capabilities from the Docker container
    cap_drop:
      - ALL
    # Order in which the Docker containers are started. In this case, the my-adcm-adcm container will be started after the my-
adcm-adpg container startup
    depends_on:
      adpg:
        # Condition under which the my-adcm-adcm container is started. In this case -- wait until the my-adcm-adpg container is
healthy
        condition: service_healthy
        restart: true
        required: true
    # Custom hostname for the Docker container
    hostname: my-adcm-adcm
    # Image name
    image: hub.arenadata.io/adcm/adcm:2.5.0
    # Metadata to add to the Docker container
    labels:
      app.arenadata.io/type: adcm
    # Create a dedicated network for the Docker container on the host
    networks:
      default: null
    # Map ports between the Docker container and the host
    ports:
      - mode: ingress
        # Port of the Docker container
        target: 8000
        # Published port
        published: "8000"
        # Port protocol
        protocol: tcp
    # Pull the Docker image from the registry for every container startup
    pull_policy: always
    # Policy of the Docker container restart. In this case, the policy restarts the Docker container in all cases, except
forcible stop of the Docker container
    restart: unless-stopped
    # Provide the Docker container with access to sensitive data defined by the x-secrets top-level element
    secrets:
      - source: adcm-db-name
        target: /run/secrets/adcm-db-name
      - source: adcm-db-user
        target: /run/secrets/adcm-db-user
      - source: adcm-db-pass
        target: /run/secrets/adcm-db-pass
    # Named volumes that are accessible by the Docker container
    volumes:
      - type: volume
        source: my-adcm-adcm
        target: /adcm/data
        volume: {}
  adpg:
    cap_drop:
      - ALL
    hostname: my-adcm-adpg
    # Docker container health check using the pg_isready utility
    healthcheck:
      test:
        - CMD-SHELL
        - pg-entrypoint isready postgres
      timeout: 3s
      interval: 10s
      retries: 3
    image: mybackspace/adpg:16.4
    labels:
      app.arenadata.io/type: adpg
    networks:
      default: null
    pull_policy: always
    # Run the Docker container with a read-only internal filesystem
    read_only: true
    restart: unless-stopped
    # Prevent the Docker container processes from gaining additional privileges
    security_opt:
      - no-new-privileges
    # User used to run the Docker container
    user: 10001:10001
    volumes:
      - type: volume
        source: my-adcm-adpg
        target: /data
        volume: {}
# Named network that allows Docker containers to communicate with each other
networks:
  default:
    name: my-adcm_default
# Named volumes that will be created with the Docker container initial startup
volumes:
  my-adcm-adcm:
    name: my-adcm-adcm
  my-adcm-adpg:
    name: my-adcm-adpg
# Sensitive data that is granted to the Docker containers
x-secrets:
  age_recipient: age1htxwzyy75rlrhut6jry0uet3ks7cw44w80jl2gktukp9m800auyqspyfws
  data:
    adcm-db-name:
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSBoMWh2d2E5TU8xR0RkQVdCalB6ZUEvZUh5d2ZqN3hpbEJBK2taMHZCM3hvCkNzbVNudHlSVUIxY0hCT3lDaHJ
NVld0VHlHK1hRU2RjMzRnUGxjaUdBUDAKLS0tIEFIcmVEVnpQVE5iVFdwQ0NRcVpNQ0w5TWRRQTB3ZmZBemZVbFcySk5ZOEEK3Y7P8Lgl0+Gx9+UwJQOzRheRbVxox/
XaXEj6M1xMukn56AgJ
    adcm-db-pass:
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSBWRXR6eEduL3piZk1oODd5QnNENUZPTHhzMXhxczRMNTVhd0VrUWJoYlNVCnV5OHBadTlpMlpTWnc4SXFoMTh
qbHE4UVBheVloNnk1U0o1N2E2dHZaZFEKLS0tIFViUm96eXJFWWJJY3o1WHlNYmdneVRQUTVRL3ZKbFlhVFF2Z2l4cDZ0QUUKr7EYAMlS/h6TeoZV4bgBKqlVDUysOK
GgAU1qWjCOezUWA/2hARb1Kt928jB+Qn8i
    adcm-db-user:
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSAzYm95bDZDV1NrbGpuYWdXN21IaUZwSFVaSHlVNVZXd2tGR1RhWE1XcTFjCnY0RUFJQnZwcUxWWk40bjUxWi9
xdHM3TFpOUTZlYWk0elpSU21mUnlTbWcKLS0tIHJvYjZHNGk4UUNDMnFCd3NFVGsvV0tyYWprQUw2Wm1pMGNHMWRYYkIwTlEK4pYnqi8iDd2GefTUK2SLz8Q3BO5Re4
DHmV33+TKiUQcjTXV6
    adpg-password:
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IFgyNTUxOSBObFI3MUptcW1ZdSs0ZGc2Tlp3SXprRGptdWhld01RekxjakZNQ1NIWFZNCnZDbzYzUXVzNUtvS3dYbkxhNnN
NV2k2TXJCZWNQeVI2Y2c0aXlKdjk0bUUKLS0tIDZEUWFNRGRScFN0Y2s5WUQwMzUzQTJKNTAzRTBmVXhWNFQ0V2tXZUN6S00KTA1a/+YkjQ1bzr4/9I5GTHAnFHgCwb
lCW656C8O2qyw2NmbZtgeypIWf6n1yz2wg
```

OpenBao

Starting with ADCM Installer 1.1, it is possible to configure and deploy OpenBao for identity-based secrets and encryption management.

ADCM Installer allows you to run OpenBao in the following modes:

- Dev mode (by default) — enables the following options:
  - automatically initialize and unseal the OpenBao server;
  - store data (in an encrypted form) in memory without requiring access rights;
  - store an OpenBao root token that can be used to access the OpenBao command line interface (CLI).

IMPORTANT

It is not recommended to use the development mode in a production environment, as data will be lost after restarting the OpenBao server. Use this mode only for testing purposes.

- Persistent mode — the exact opposite of the dev mode which allows you to use the PostgreSQL storage backend to persist data after restarting the OpenBao server.
- High availability (HA) mode — protects from failures by running several OpenBao servers using the PostgreSQL storage backend.

NOTE

You can run OpenBao in the high availability mode by running ADCM Installer on the required hosts.

For more information on OpenBao features and functions, please refer to the OpenBao documentation.

For example, to create a configuration file for deploying the ADCM installation with a PostgreSQL database and installing OpenBao, perform the following steps:

1. Run the `init` command by specifying the `--vault` flag:

```
$ adi init <installation_name> --adpg --vault -i
```

2. Determine the values of parameters required to deploy the ADCM installation with a PostgreSQL database and to install OpenBao. The parameters are described in the table.

HashiCorp Consul

Starting with ADCM Installer 1.1, it is possible to configure and deploy HashiCorp Consul in the dev mode. HashiCorp Consul is a solution for network services that allows you to coordinate secure communication between services and check their availability. HashiCorp Consul can be used instead of the status-checker in all products (ADB Control, ADH, and so on).

NOTE

HashiCorp Consul provides the ability to register services, but it does not automatically do it. Products should register their services in HashiCorp Consul on their own.

To create a configuration file for deploying the ADCM installation with a PostgreSQL database and installing HashiCorp Consul, perform the following steps:

1. Run the `init` command by specifying the `--consul` flag:

```
$ adi init <installation_name> --adpg --consul -i
```

2. Determine the values of parameters required to deploy the ADCM installation with a PostgreSQL database and to install HashiCorp Consul. The parameters are described in the table.

Deploy the ADCM installation

The `apply` command allows you to deploy the ADCM installation with the created files *adcm.yaml* (or *adcm.yml*), and *age.key*.

```
$ sudo adi apply
```

When you run the command, the *docker-compose.yaml* file is generated which participates in the following actions:

1. Creating and running the `my-adcm-adpg` container.
2. Creating the PostgreSQL database.
3. Creating and running the `my-adcm-adcm` container.

To ensure that the ADCM installation is deployed successfully, run the commands described in the Installation article.

To deploy the ADCM installation in debug mode, specify the `--dry-run` flag. As a result, the *docker-compose.yaml* file is created without creating and running containers.

For more information on other flags that can be specified when running the `apply` command, use the following command:

```
$ adi apply --help
```

NOTE

With the `--help` flag, you can also get a brief description of other commands described below.

Create scripts with auto-complete

The `completion` command allows you to generate scripts with auto-complete for the ADCM Installer commands using the `Tab` key for the following shells:

- Bash
- fish
- PowerShell
- Zsh

NOTE

For Bash, first install the bash-completion package required for the script.

For more detailed instructions on how to use a generated script, run the following command:

```
$ adi completion <shell> --help
```

where `<shell>` is the shell name.

List running ADCM installations

The `list` command allows you to obtain the following information on the ADCM installations:

- `NAMES` — the ADCM installation name;
- `STATUS` — the status and the number of the Docker containers;
- `CONFIG FILES` — the path to the configuration file.

Command example:

```
$ sudo adi list
```

The example of the command execution result:

```
NAMES               STATUS              CONFIG FILES
my-adcm             running(3)          /home/adelya/adcm-installer/adcm.yaml
```

To display information on stopped ADCM installations, specify the `--all` flag.

List available ADCM versions

The `adcm-versions` command allows you to check for the new ADCM image versions available for download from Arenadata Docker Registry:

```
$ adi adcm-versions --all
```

The example of the command execution result:

```
2.5.1
2.5.0
2.4.0
2.3.0
2.2.0
2.1.2
2.1.1
2.1.0
2.0.0
```

To display the last five versions, run the `adcm-versions` command without specifying the `--all` flag.

Manage secrets

The `secrets` command allows you to manage secrets specified in the configuration file as follows:

- `secrets new-key` — creates a new private key which by default is saved in the *age.key* file;
- `secrets show` — shows secrets defined during initialization;
- `secrets set` — determines new secret values, for example:

```
$ adi secrets set adcm-db-pass 1234
```

Stop and remove running ADCM installations

The `delete` command stops running ADCM installations using the specified configuration file or running installation name.

If *adcm.yaml* (or *adcm.yml*) or *age.key* are located in the directory in which the command is run, the `delete` command will be executed automatically without specifying the configuration file or the ADCM installation name:

```
$ sudo adi delete
```

As a result, the ADCM installation will be stopped with the preservation of data specified during initialization of the configuration file.

To remove a running ADCM installation along with volumes related to the installation, specify the `--volumes` flag. Confirm the removal during the execution of the command.

IMPORTANT

Note that running this command will permanently delete all information.

Connect to ADCM

Mikhail Serov

Contents

To Table of Contents

Connect via browser UI

Connect via API

You can connect to ADCM in two ways:

- via browser UI;
- via API.

Both ways are described in detail below.

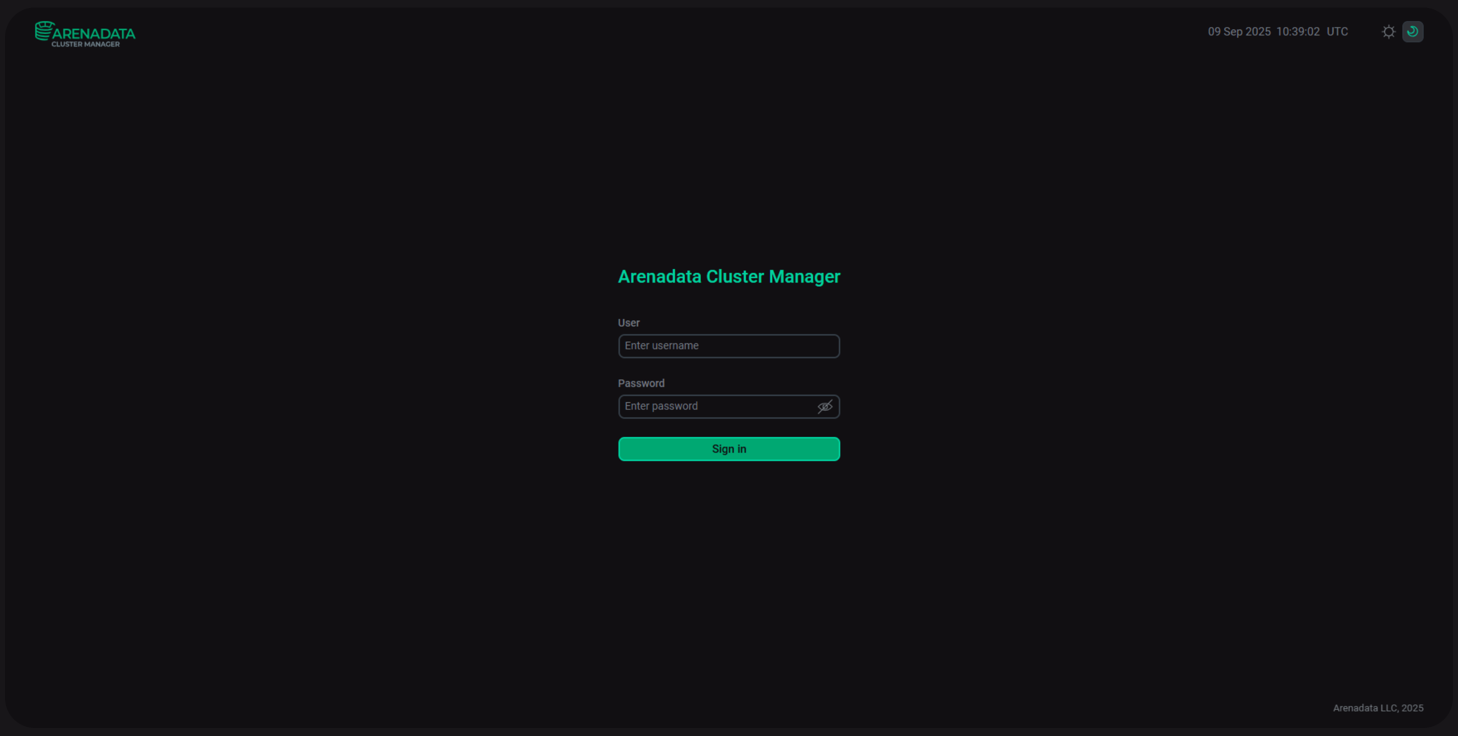
Connect via browser UI

To connect to ADCM via its browser UI, perform the following steps:

1. Enter the IP address of the machine you need to access and the port you used when adding the ADCM container in the address bar of your browser. The format should be the following: *<IP address>:<port>*.

> **NOTE**
>
> If you used the command from the Get started guide, the port should be `8000`.

2. Enter your credentials (***User*** and ***Password***) into the corresponding fields in the Login window. Click ***Sign in***.



ARENADATA
CLUSTER MANAGER
09 Sep 2025 10:39:02 UTC
Arenadata Cluster Manager
User
Enter username
Password
Enter password
Sign in
Arenadata LLC, 2025

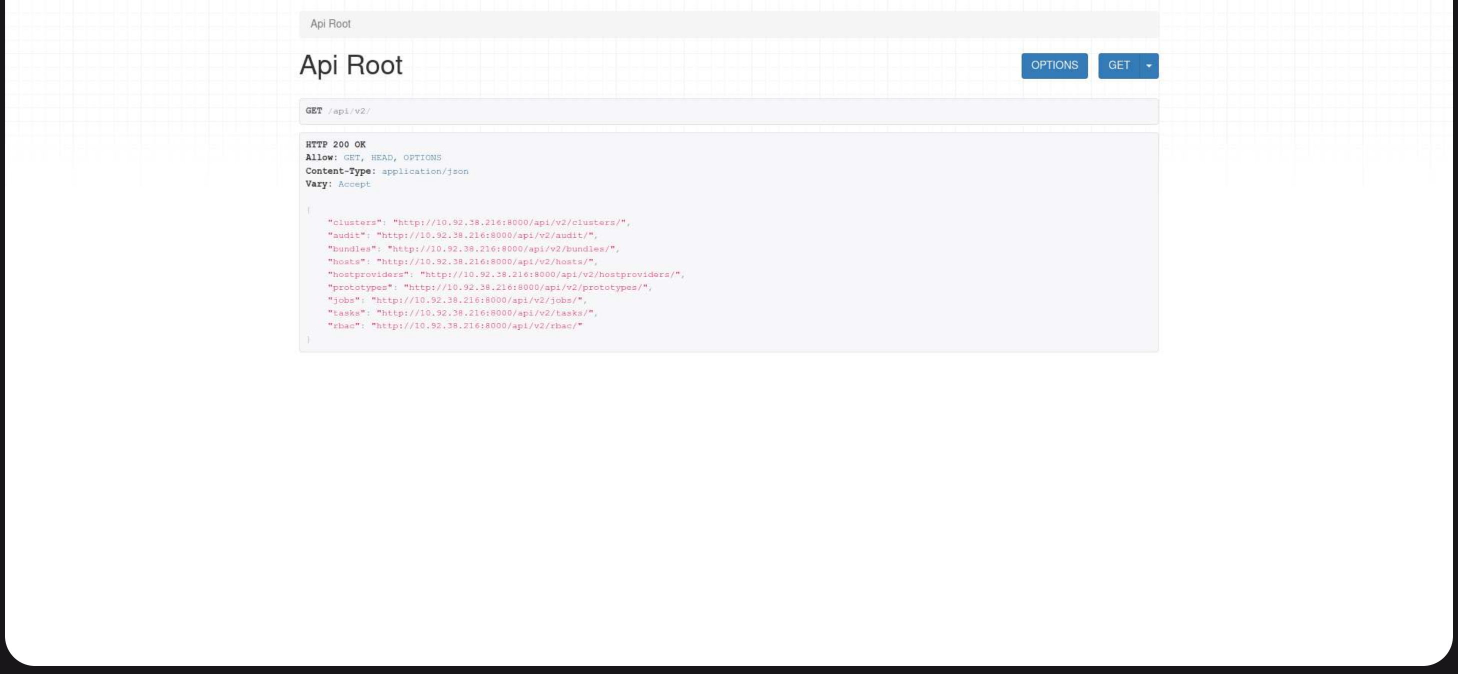
The login window

Connect via API

ADCM provides an API to access and work with ADCM objects. To connect to ADCM via its API, perform the following steps:

1. Enter the *<IP address>:<port>/api/v2/* in the address bar of your browser. *<IP address>* stands for the IP address of the machine where the ADCM is installed, *<port>* stands for the port where the ADCM container is running. API Root page opens.

2. Click ***Log in*** in the upper right part of the screen and enter your credentials (***Username*** and ***Password***) in the window that opens. Click ***Log in***.



Django REST framework
admin
Api Root
Api Root
OPTIONS
GET

```
GET /api/v2/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "clusters": "http://10.92.38.216:8000/api/v2/clusters/",
    "audit": "http://10.92.38.216:8000/api/v2/audit/",
    "bundles": "http://10.92.38.216:8000/api/v2/bundles/",
    "hosts": "http://10.92.38.216:8000/api/v2/hosts/",
    "hostproviders": "http://10.92.38.216:8000/api/v2/hostproviders/",
    "prototypes": "http://10.92.38.216:8000/api/v2/prototypes/",
    "jobs": "http://10.92.38.216:8000/api/v2/jobs/",
    "tasks": "http://10.92.38.216:8000/api/v2/tasks/",
    "rbac": "http://10.92.38.216:8000/api/v2/rbac/"
}
```

The API Root page

ADCM role model

Mikhail Serov

Contents

To Table of Contents

Built-in roles

Permissions and roles

A role model is implemented in ADCM. Specific sets of permissions for operations with various ADCM objects can be grouped into roles. The roles are used to create policies to grant roles's permissions to user groups.

Permission is a basic element of the role model. **Role** is a set of permissions. Permissions can not be assigned directly to the groups. Permissions can be only added to the roles.

Some permissions are necessary to work with any product via ADCM. Such permissions are always present in ADCM. Other permissions are necessary to work only with a particular product. Those permissions appear in ADCM when a bundle is uploaded.

The role is assigned to a user group via policy. Assignment of multiple roles is not supported. Each user inherits the roles of the group to which they belong.

> **NOTE**
>
> Only the ADCM administrator (= superuser) can assign roles to the users. In particular, only the superuser can grant a superuser role to another user.

Policy is a triad that consists of the following elements:

1. User group.
2. Role.
3. Object, to which the role's permissions are applied. Object can have one of the following types:
    - Cluster
    - Service
    - Host
    - Provider

While creating a policy, the user is prompted to select one or more objects of the type that corresponds to the chosen user role.

> **NOTE**
>
> A policy is applied to an object as well as to all of its child objects. For example, if a policy is given to a cluster, then that policy also affects all services, components, and hosts of that cluster.

Built-in roles

ADCM comes with a few built-in roles. Changing the permissions of built-in roles is prohibited.

Built-in ADCM roles

Role name	Description	Object type
ADCM User	View-only role that provides ability to view all objects information	None
ADCM Auditor	View-only role that provides ability to view audit results	None
Service Administrator	Role provides ability to configure and control the life cycle of the relevant service	Service
Cluster Administrator	Role provides ability to configure and control the relevant cluster, its hosts, and its services	Cluster
Provider Administrator	Role gives full control over the relevant hostprovider and its hosts	Provider
ADCM Administrator	Role gives full control over all aspects of ADCM	None

Permissions and roles

In addition to the built-in roles, ADCM administrator can create custom roles with permissions. Available permissions and their relations to the built-in roles are listed in the table below.

Permissions and roles

Permission name	Description	ADCM User	Service Administrator	Provider Administrator	Cluster Administrator	ADCM Administrator	ADCM Auditor
View any object configurations	Ability to view any object configurations	+				+	
View cluster configurations	The ability to view cluster configurations				+	+	
View service configurations	Ability to view service configurations		+		+	+	
View component configurations	Ability to view component configurations		+		+	+	
View provider configurations	Ability to view hostprovider configurations			+		+	
View host configurations	Ability to view host configurations		+	+	+	+	
Edit cluster configurations	Ability to edit cluster configurations				+	+	
Edit service configurations	Ability to edit service configurations		+		+	+	
Edit component configurations	Ability to edit component configurations		+		+	+	
Edit provider configurations	Ability to edit hostprovider configurations			+		+	
Edit host configurations	Ability to edit host configurations			+	+	+	
View any object imports	Ability to view imports of all objects	+				+	
View imports	Ability to view imports of a chosen object		+		+	+	
Manage Imports	Ability to manage imports		+		+	+	
View any cluster host-components	Ability to view any cluster hosts and components mappings	+				+	
View Host-Components	Ability to view hosts and components mapping		+		+	+	
Manage Host-Components	Ability to change hosts and components mapping				+	+	
Add service	The ability to add services to cluster				+	+	
Remove hosts	Ability to remove hosts (in the ***Hosts*** section)			+		+	
Add hosts to the cluster	Ability to add hosts to the cluster				+	+	
Remove hosts from the cluster	Ability to remove hosts from the cluster				+	+	
Upgrade cluster bundle	Ability to upgrade the cluster bundle				+	+	
Upgrade provider bundle	Ability to upgrade the hostprovider bundle			+		+	
Create hostprovider	Ability to create hostproviders					+	
Create host	Ability to create hosts			+	+	+	
Remove hostprovider	Ability to remove hostproviders					+	
Create cluster	Ability to create clusters					+	
Remove cluster	Ability to remove clusters					+	
Upload bundle	Ability to upload bundles			+	+	+	
Remove bundle	The ability to remove bundles			+	+	+	
View audit operations	Ability to view results of operation audit	+	+	+	+	+	+
View audit logins	Ability to view results of login audit	+	+	+	+	+	+
View ADCM settings	Ability to view settings					+	
Edit ADCM settings	Ability to edit settings					+	
View users	Ability to view users					+	+
Add new user	Ability to create users					+	
Delete user	Ability to remove users					+	
Update user	Ability to update users					+	
View roles	Ability to view roles					+	+
Add new role	Ability to create roles					+	
Delete role	Ability to remove roles (only for custom roles)					+	
Update role	Ability to update roles (only for custom roles)					+	
View groups	Ability to view groups					+	+
Add new group	Ability to create groups					+	
Delete group	Ability to remove groups					+	
Update group	Ability to update groups					+	
View policies	Ability to view policies					+	
Add new policy	Ability to create policies					+	
Delete policy	Ability to remove policies					+	
Update policy	Ability to update policies					+	
Cluster Action: <Action name>	Ability to perform the <Action name> action				+	+	
Host Action: <Action name>	Ability to perform the <Action name> action			+	+	+	
Service Action: <Action name>	Ability to perform the <Action name> action		+		+	+	
Component Action: <Action name>	Ability to perform the <Action name> action		+		+	+	
Provider Action: <Action name>	Ability to perform the <Action name> action			+		+	

Create roles

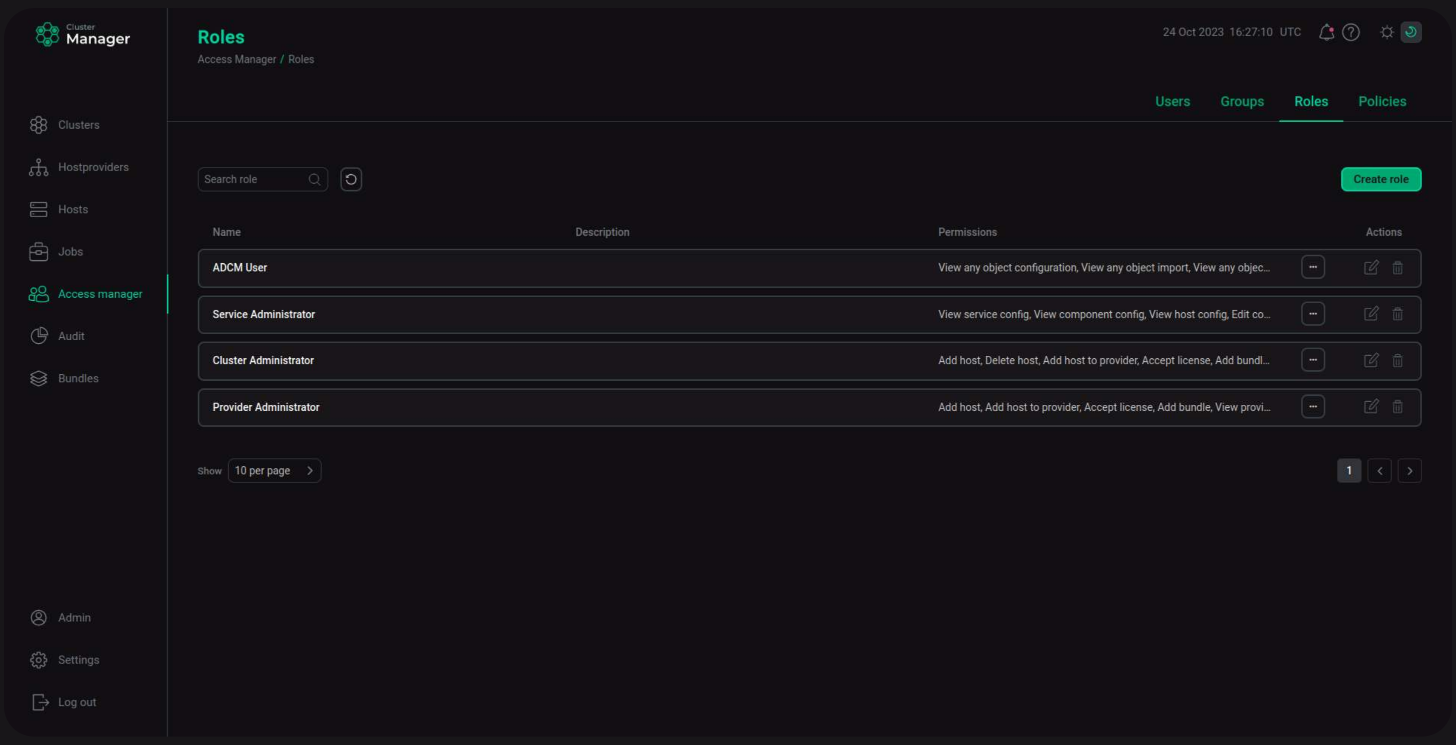
Mikhail Serov

To Table of Contents

This feature is available only to users with appropriate permissions.

To create a new role:

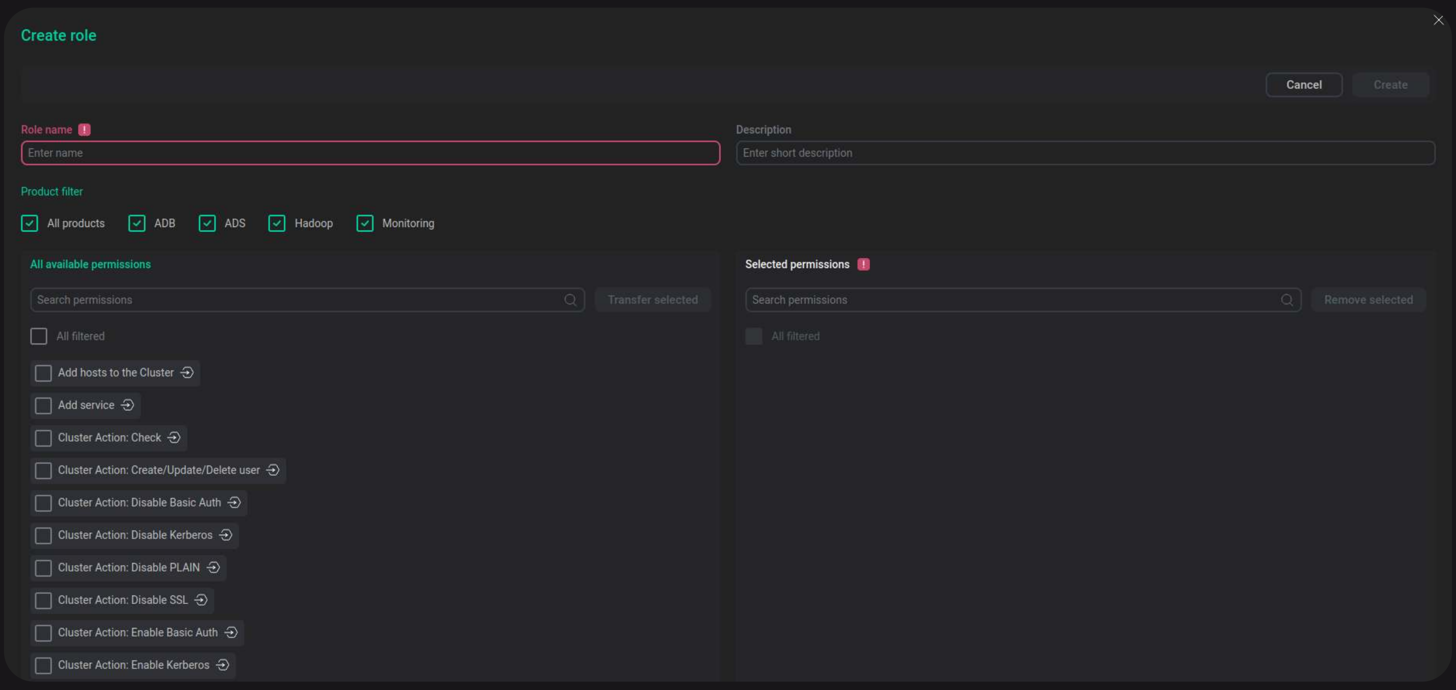
1. Log into ADCM console.
2. Navigate to the ***Roles*** tab of the ***Access manager*** page. Roles list opens.


Cluster Manager
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Admin
Settings
Log out
Roles
Access Manager / Roles
24 Oct 2023 16:27:10 UTC
Users
Groups
Roles
Policies
Search role
Create role
Show
10 per page
1

Name	Description	Permissions	Actions
ADCM User		View any object configuration, View any object import, View any objec...	
Service Administrator		View service config, View component config, View host config, Edit co...	
Cluster Administrator		Add host, Delete host, Add host to provider, Accept license, Add bundl...	
Provider Administrator		Add host, Add host to provider, Accept license, Add bundle, View provi...	

Roles subsection

3. Click ***Create role***. The ***Create role*** window opens.


Create role
Cancel
Create
Role name
Enter name
Description
Enter short description
Product filter
All products
ADB
ADS
Hadoop
Monitoring
All available permissions
Search permissions
Transfer selected
All filtered
Add hosts to the Cluster
Add service
Cluster Action: Check
Cluster Action: Create/Update/Delete user
Cluster Action: Disable Basic Auth
Cluster Action: Disable Kerberos
Cluster Action: Disable PLAIN
Cluster Action: Disable SSL
Cluster Action: Enable Basic Auth
Cluster Action: Enable Kerberos
Selected permissions
Search permissions
Remove selected
All filtered

The Create role window

4. Fill in the ***Role name*** field.
5. Add permissions. Use buttons with product names as filters to select permissions. You should add at least one permission.
6. Click ***Create***. The new role is created.

IMPORTANT

You cannot mix permissions related to the objects of different hierarchy.

Cluster hierarchy:

- Cluster
- Service
- Component
- Host

Hostprovider hierarchy:

- Hostprovider
- Host

Since a host object belongs to both hierarchies, it has all kinds of permissions, so you can launch actions related to both hierarchies when working with hosts.

Create policies

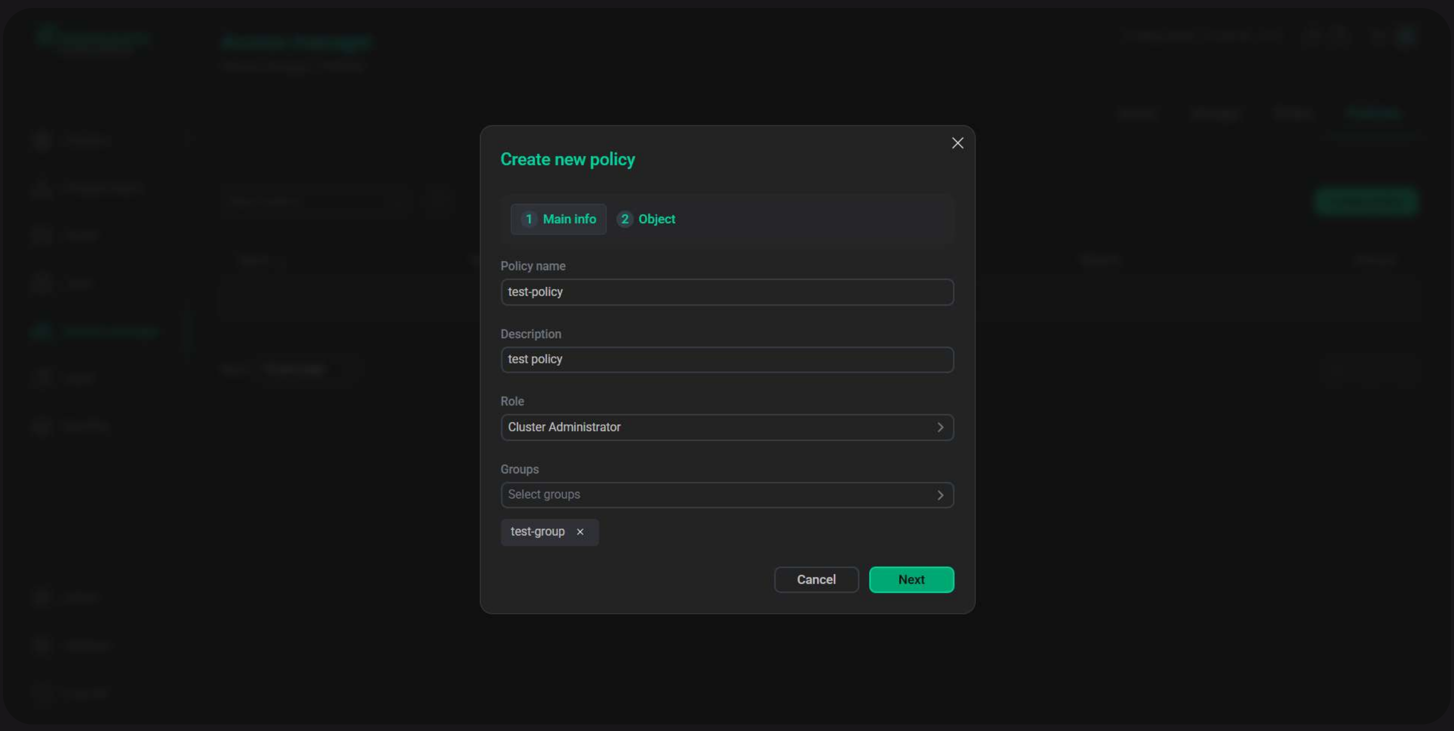
Mikhail Serov

To Table of Contents

This feature is available only to users with appropriate permissions.

To create a new policy:

1. Log into the ADCM web interface.
2. Navigate to the ***Policies*** tab of the ***Access manager*** page. Policy list opens.
3. Click ***Create Policy***. The ***Create new policy*** window opens.
4. Fill in the fields at the ***Main info*** step. ***Policy name***, ***Role***, and ***Groups*** are required.


Create new policy
1 Main info
2 Object
Cancel
Next

Policy name: test-policy

Description: test policy

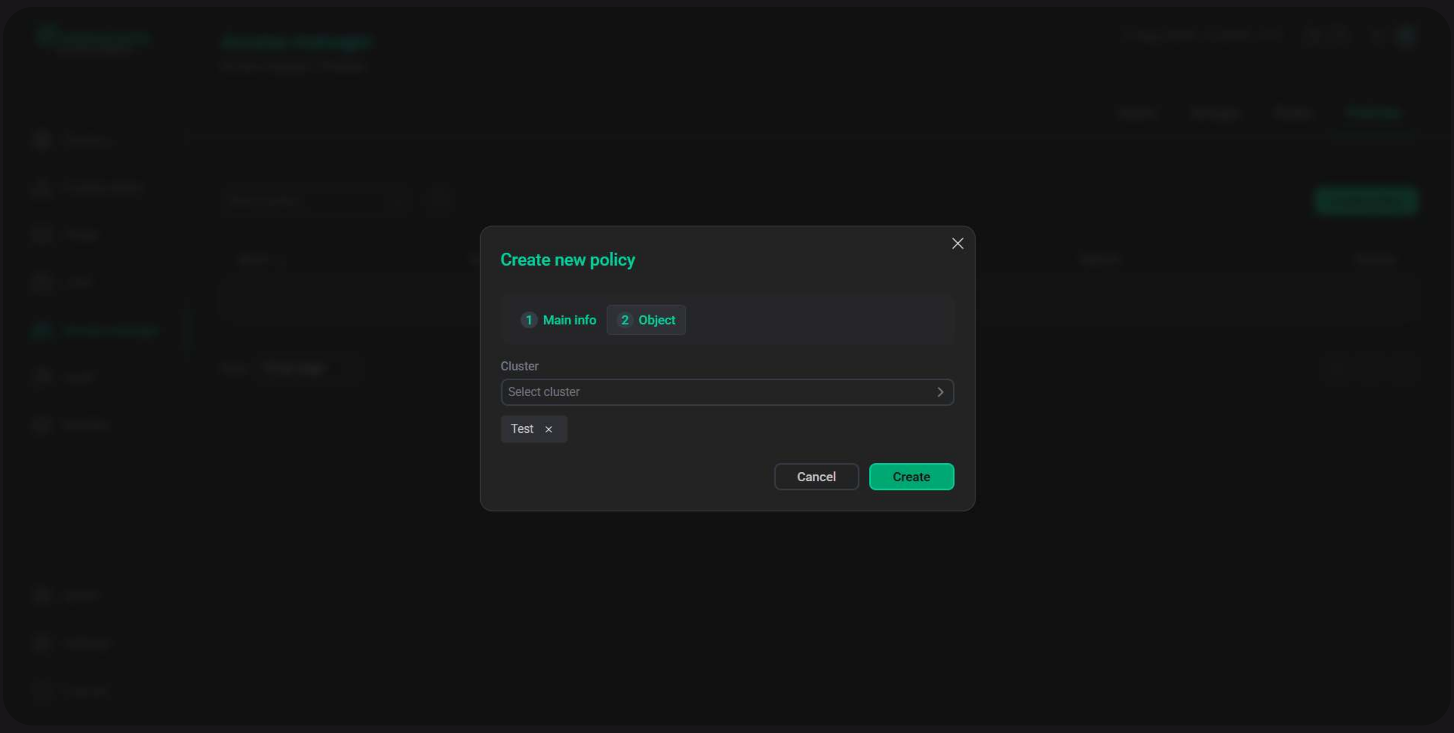
Role: Cluster Administrator

Groups: Select groups

test-group ×

The Create new policy window — Main info

5. Specify the objects at the ***Object*** step.

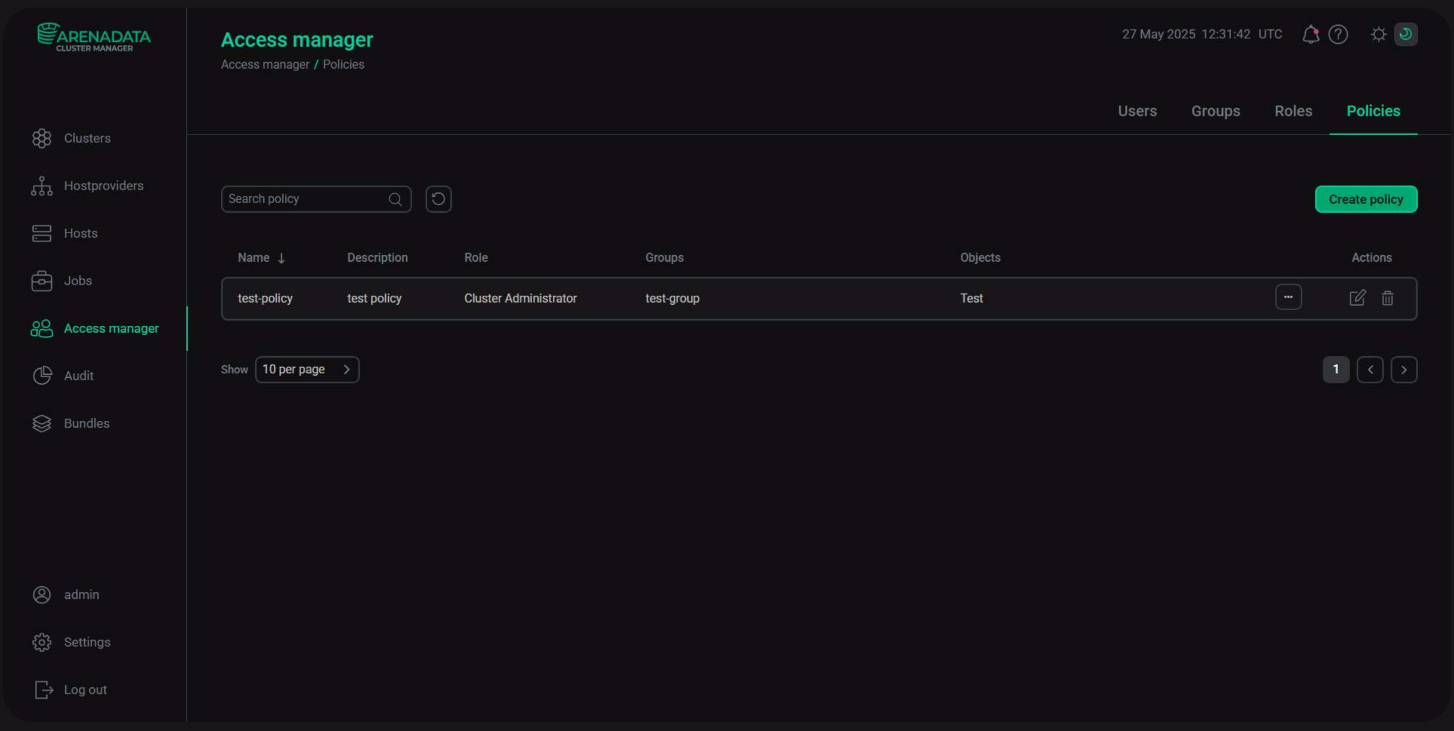

Create new policy
1 Main info
2 Object
Cancel
Create

Cluster: Select cluster

Test ×

The Create new policy window — Object

6. Click ***Create***. The new policy is added.


ARENADATA
CLUSTER MANAGER
Access manager
Access manager / Policies
27 May 2025 12:31:42 UTC
Users
Groups
Roles
Policies
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search policy
Create policy
Show
10 per page
1

Name ↓	Description	Role	Groups	Objects	Actions
test-policy	test policy	Cluster Administrator	test-group	Test	

New policy in the policy list

Policies migration

Mikhail Serov

Contents

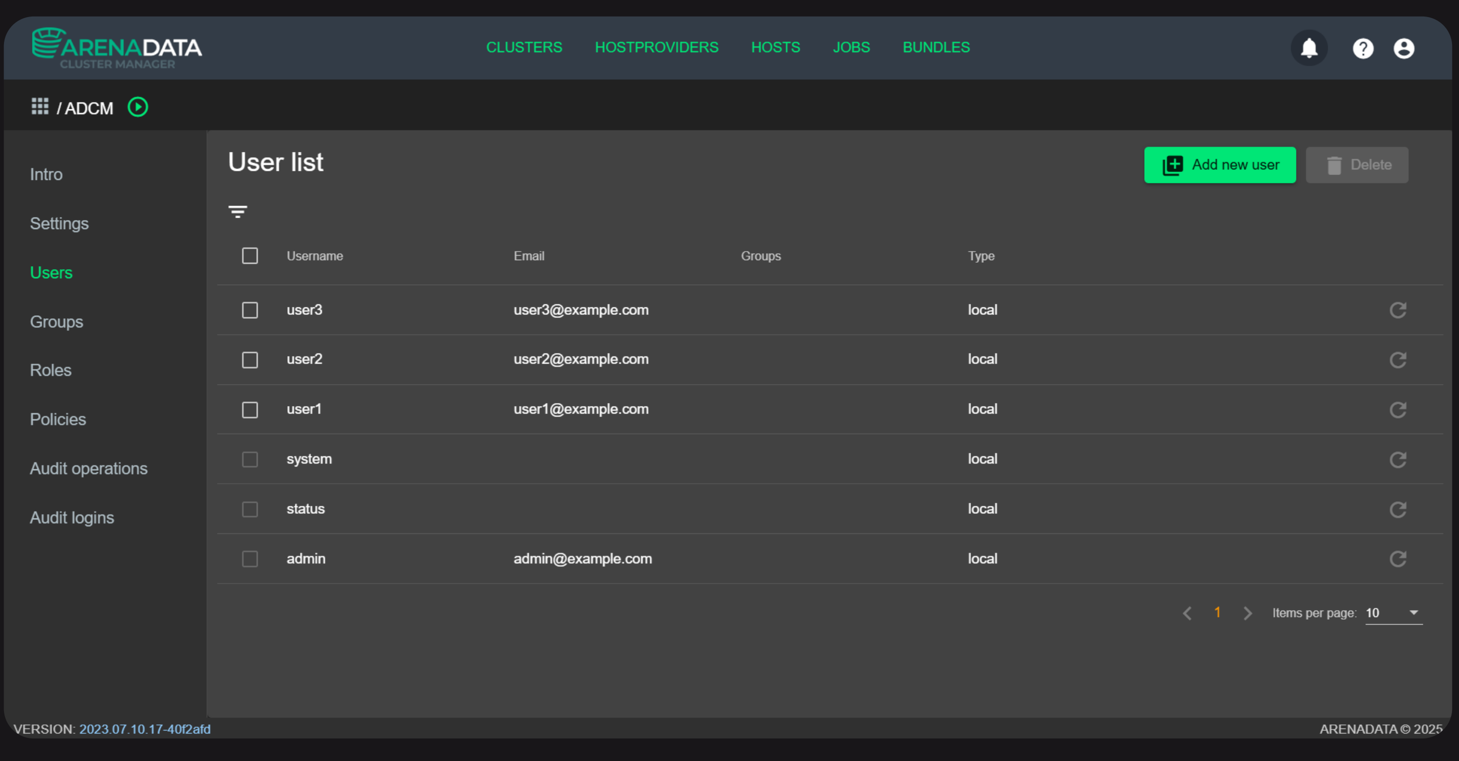
To Table of Contents

Example

ADCM performs migration of policies on upgrade. This will match the policies assigned via users with the policies assigned via user groups. If you had a role model configured for an ADCM version earlier than 2023.09.28.17, then upgrading to the current version will create user groups containing all users affected by the policies, and the policies will become reconfigured to those user groups.

Example

Consider three users: `user1`, `user2`, and `user3`.



User list

Username	Email	Groups	Type
user3	user3@example.com		local
user2	user2@example.com		local
user1	user1@example.com		local
system			local
status			local
admin	admin@example.com		local

VERSION: 2023.07.10.17-40f2afd

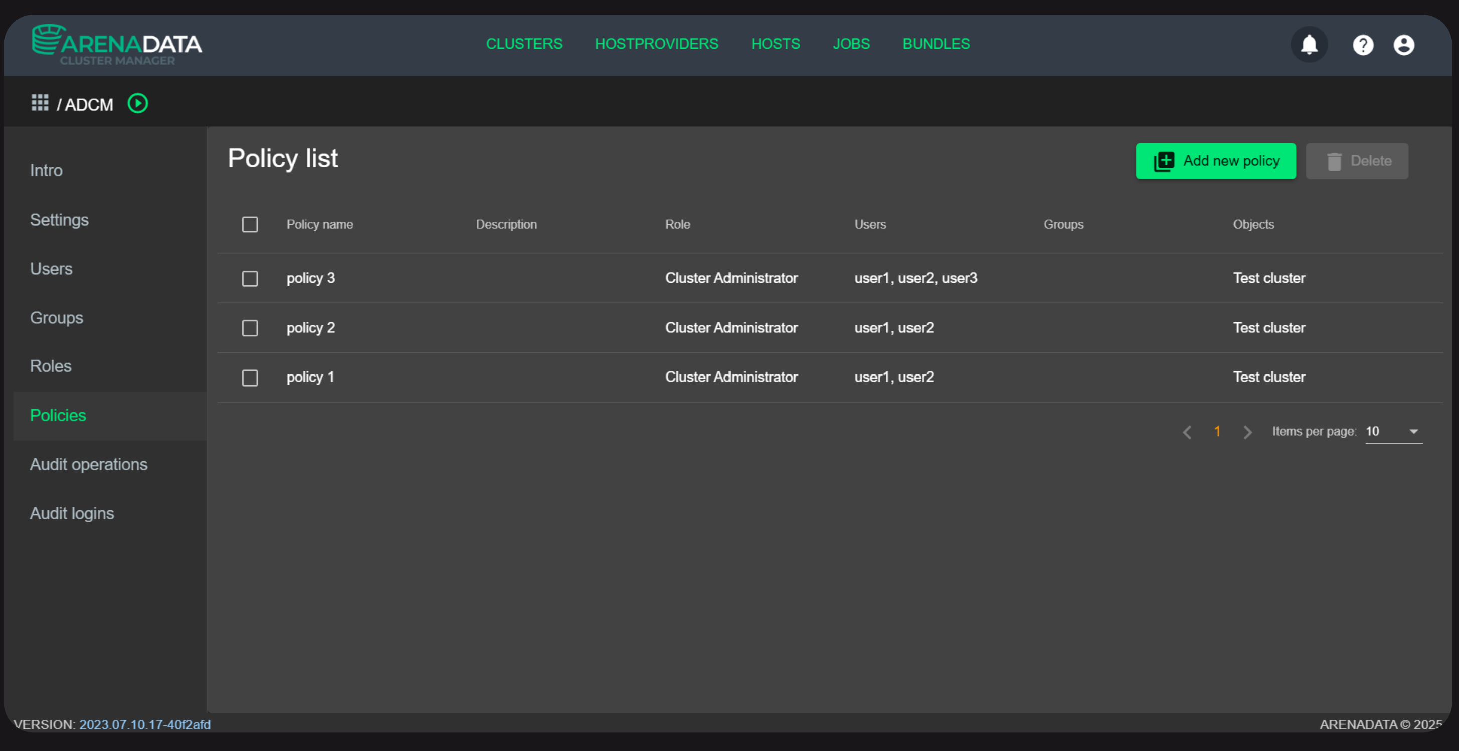
ARENADATA © 2025

ARENADATA
CLUSTER MANAGER
CLUSTERS
HOSTPROVIDERS
HOSTS
JOBS
BUNDLES
/ ADCM
Intro
Settings
Users
Groups
Roles
Policies
Audit operations
Audit logins
Add new user
Delete
1
Items per page: 10

User list before migration

Suppose the following:

- the `user1` and `user2` users are under three policies: `policy 1`, `policy 2`, and `policy 3`.
- the `user3` user is only under `policy 3`.



Policy list

Policy name	Description	Role	Users	Groups	Objects
policy 3		Cluster Administrator	user1, user2, user3		Test cluster
policy 2		Cluster Administrator	user1, user2		Test cluster
policy 1		Cluster Administrator	user1, user2		Test cluster

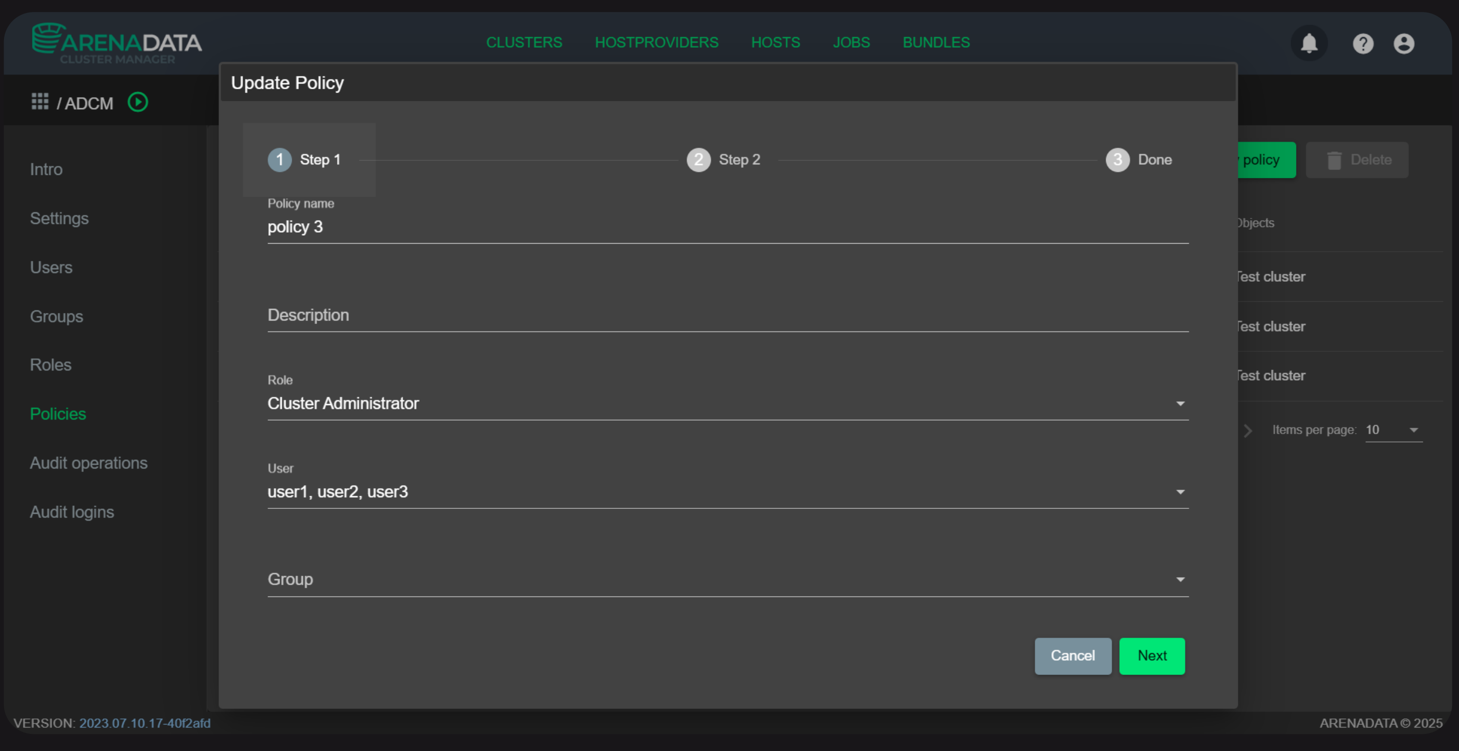
VERSION: 2023.07.10.17-40f2afd

ARENADATA © 2025

ARENADATA
CLUSTER MANAGER
CLUSTERS
HOSTPROVIDERS
HOSTS
JOBS
BUNDLES
/ ADCM
Intro
Settings
Users
Groups
Roles
Policies
Audit operations
Audit logins
Add new policy
Delete
1
Items per page: 10

Policy list before migration

Inside the policies, the users affected by these policies are specified.



Update Policy

1 Step 1 — 2 Step 2 — 3 Done

Policy name: policy 3

Description

Role: Cluster Administrator

User: user1, user2, user3

Group

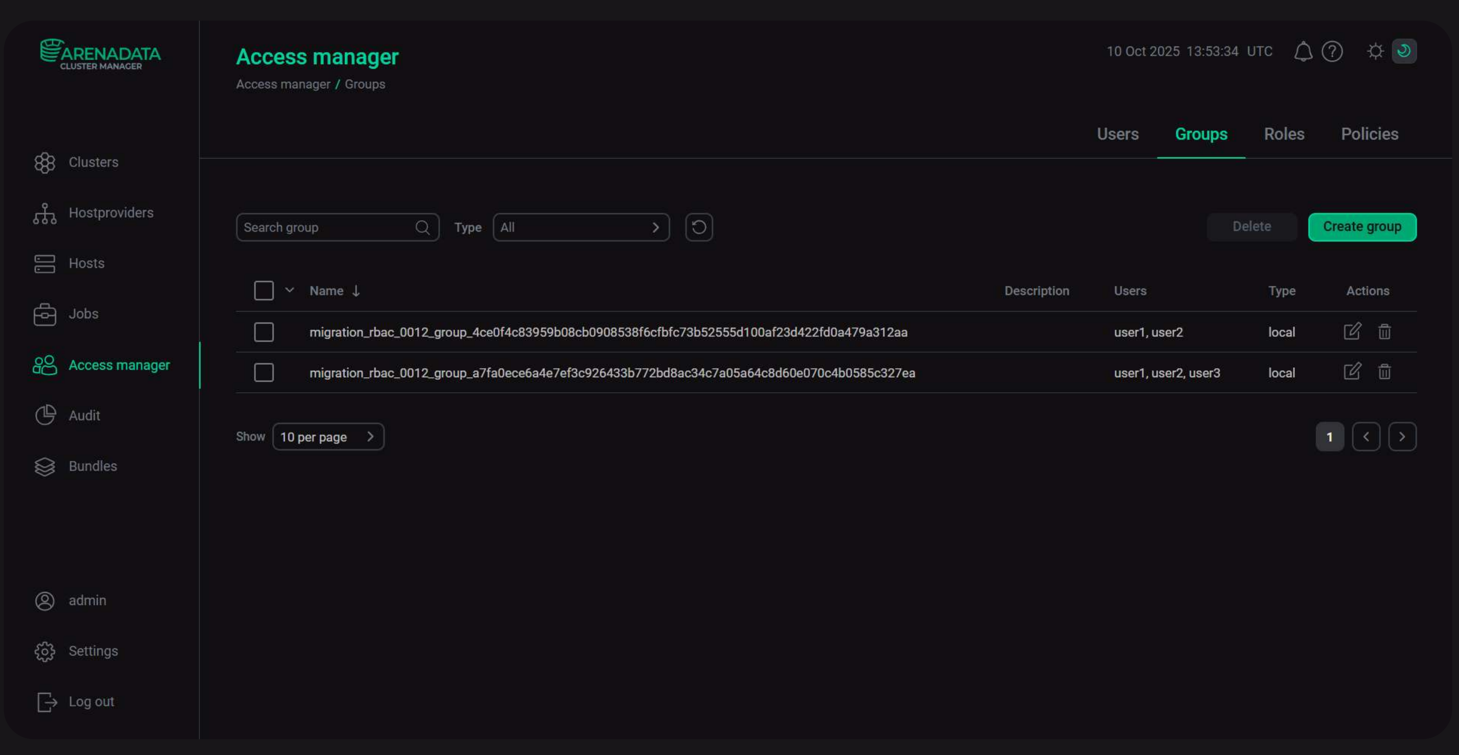
VERSION: 2023.07.10.17-40f2afd

ARENADATA © 2025

ARENADATA
CLUSTER MANAGER
CLUSTERS
HOSTPROVIDERS
HOSTS
JOBS
BUNDLES
/ ADCM
Intro
Settings
Users
Groups
Roles
Policies
Audit operations
Audit logins
Cancel
Next

Policy 3 before migration

Now upgrade the ADCM. When upgrading, ADCM will automatically create user groups for all users in the policies.



Access manager

Access manager / Groups

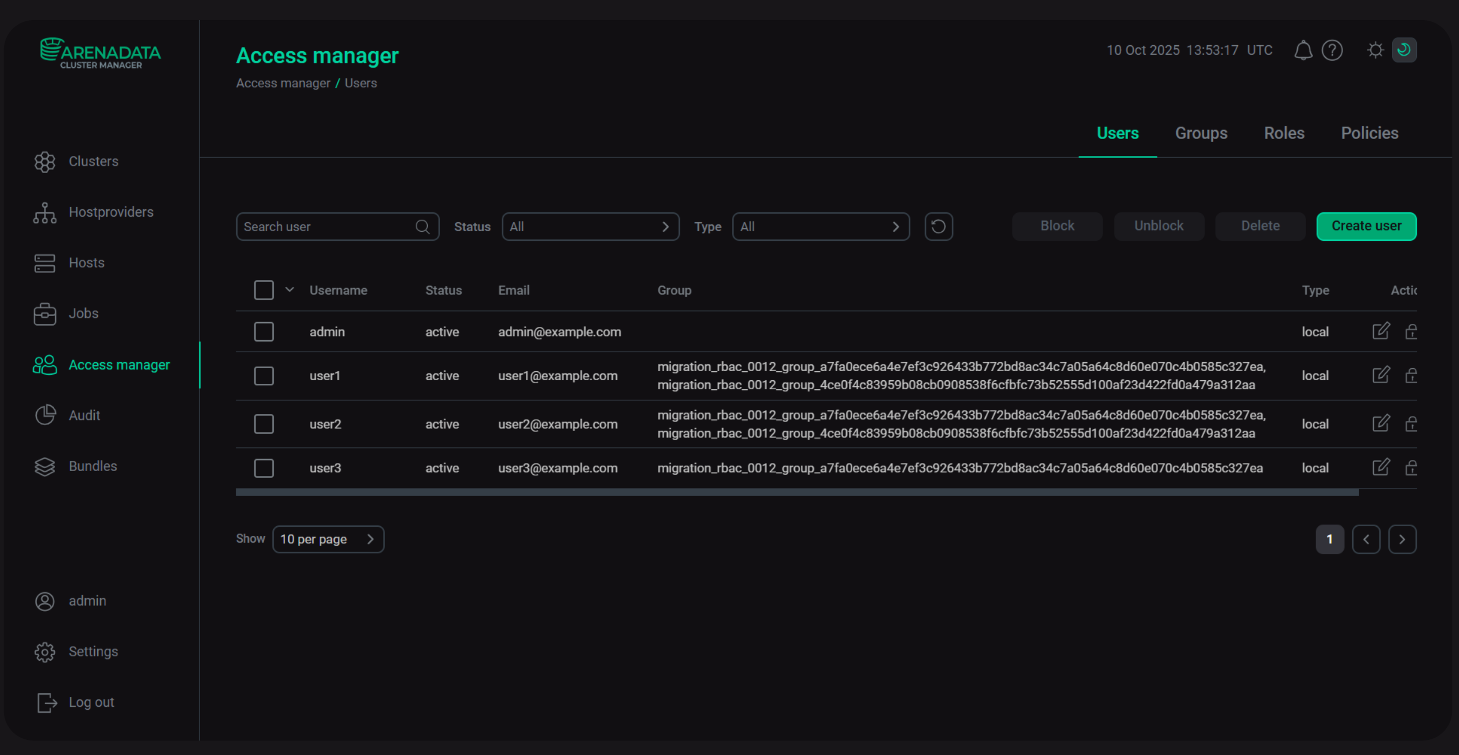
10 Oct 2025 13:53:34 UTC

Name	Description	Users	Type
migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa		user1, user2	local
migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea		user1, user2, user3	local

ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Users
Groups
Roles
Policies
Search group
Type
All
Delete
Create group
Actions
Show
10 per page
1

Group list after migration

In this case, `user1` and `user2` will be in the `migration_rbac_0012_group_4ce0f4c83959b…` user group. The `migration_rbac_0012_group_a7fa0ece6a4e7…` user group will contain all three users: `user1`, `user2`, and `user3`. Full names of the groups are not given here for brevity.



Access manager

Access manager / Users

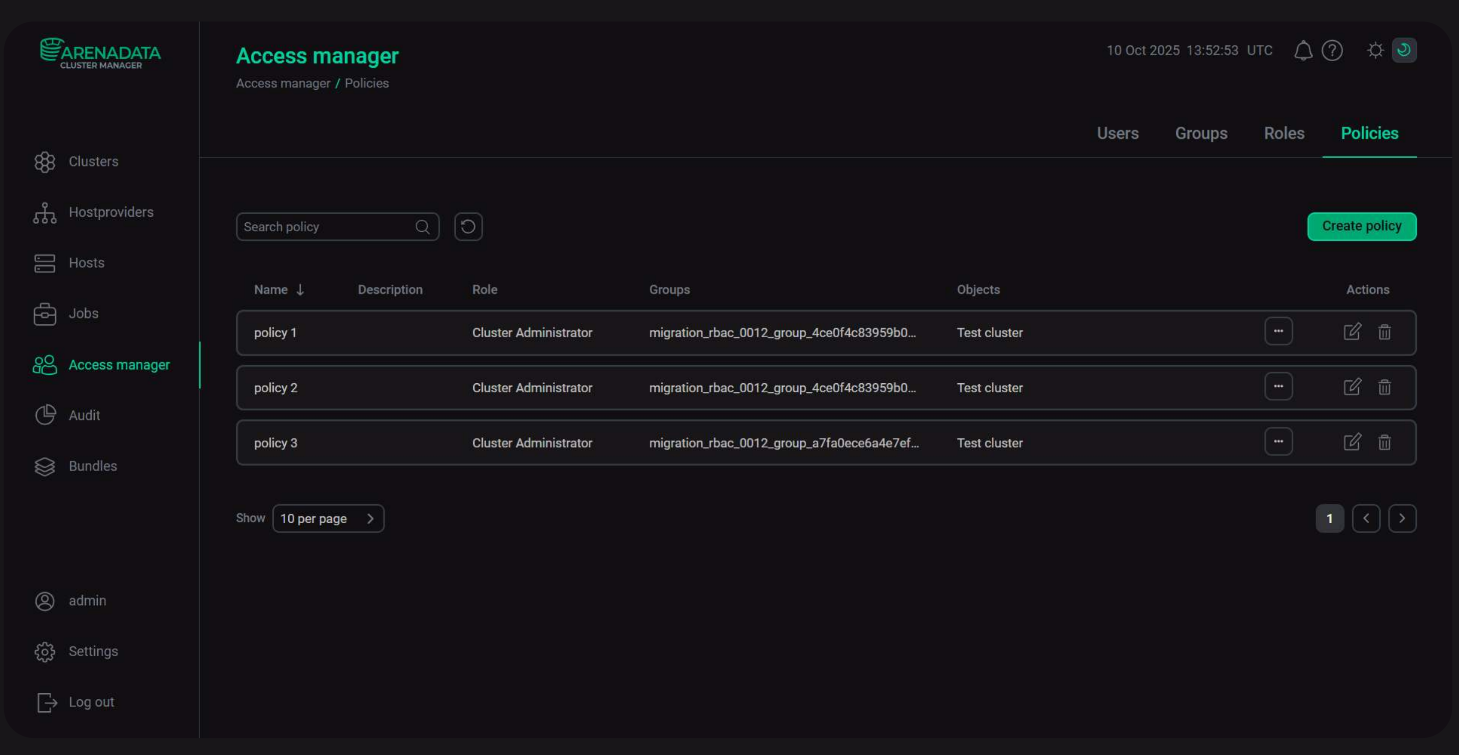
10 Oct 2025 13:53:17 UTC

Username	Status	Email	Group	Type
admin	active	admin@example.com		local
user1	active	user1@example.com	migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea, migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa	local
user2	active	user2@example.com	migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea, migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa	local
user3	active	user3@example.com	migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local

ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Users
Groups
Roles
Policies
Search user
Status
All
Type
All
Block
Unblock
Delete
Create user
Actic
Show
10 per page
1

User list after migration

Then the policies `policy 1` and `policy 2` will be associated with the same `migration_rbac_0012_group_4ce0f4c83959b…` user group, and `policy 3` will be associated only with `migration_rbac_0012_group_a7fa0ece6a4e7…`.



Access manager

Access manager / Policies

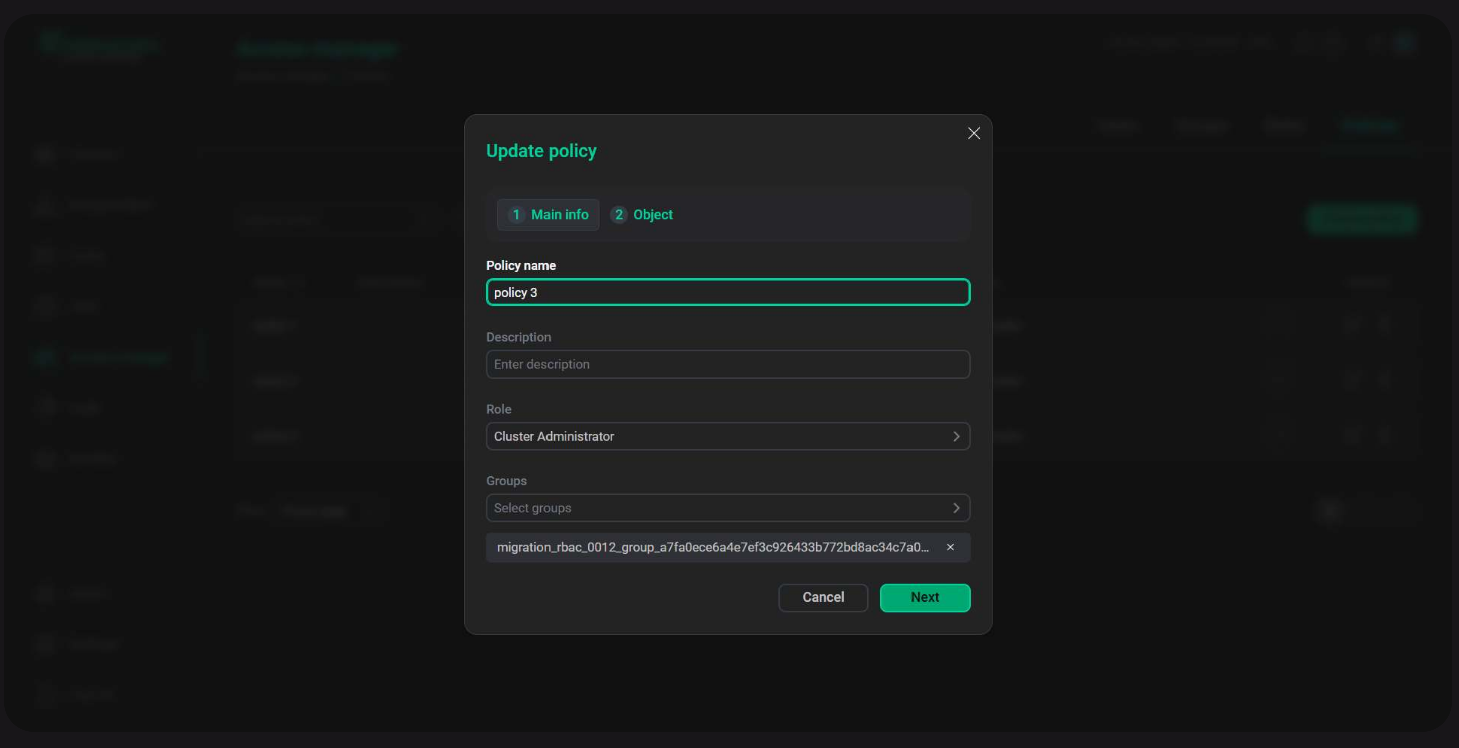
10 Oct 2025 13:52:53 UTC

Name	Description	Role	Groups	Objects
policy 1		Cluster Administrator	migration_rbac_0012_group_4ce0f4c83959b0…	Test cluster
policy 2		Cluster Administrator	migration_rbac_0012_group_4ce0f4c83959b0…	Test cluster
policy 3		Cluster Administrator	migration_rbac_0012_group_a7fa0ece6a4e7ef…	Test cluster

ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Users
Groups
Roles
Policies
Search policy
Create policy
Actions
Show
10 per page
1

Policy list after migration

Groups are substituted for users inside the policies.



Update policy

1 Main info 2 Object

Policy name: policy 3

Description: Enter description

Role: Cluster Administrator

Groups: Select groups

migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a0…

Cancel
Next

Policy 3 after migration

Configure custom roles

Mikhail Serov

Contents

To Table of Contents

Example

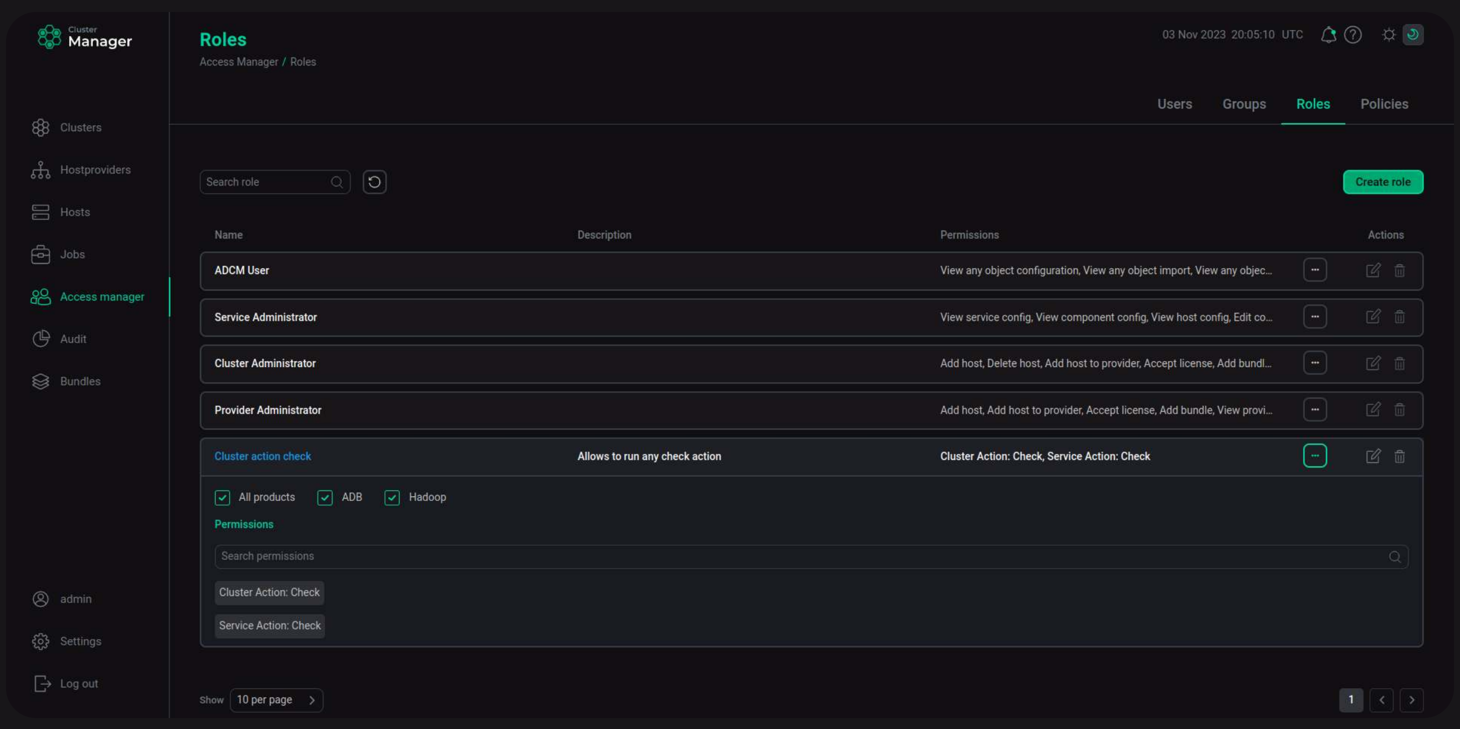
In some cases, the functionality of the built-in ADCM roles is not enough to grant the users sufficiently granular access. In this case, you can create a new role with appropriate permissions.

The process of creating a new role is described on the Create roles page.

Example

Suppose you need a role that only allows to run any check action on a cluster. To create this role, do the following:

1. In the ***Create role*** window on the ***Roles*** tab enter `Cluster action check` as the role name and include the `Cluster Action: Check` and `Service Action: Check` permissions.
2. Click ***Create***.


Cluster
Manager
Roles
Access Manager / Roles
03 Nov 2023 20:05:10 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Users
Groups
Roles
Policies
Search role
Create role
Search permissions
Show
10 per page
1

Name	Description	Permissions	Actions
ADCM User		View any object configuration, View any object import, View any objec...	
Service Administrator		View service config, View component config, View host config, Edit co...	
Cluster Administrator		Add host, Delete host, Add host to provider, Accept license, Add bundl...	
Provider Administrator		Add host, Add host to provider, Accept license, Add bundle, View provi...	
Cluster action check	Allows to run any check action	Cluster Action: Check, Service Action: Check	

☑ All products ☑ ADB ☑ Hadoop

Permissions

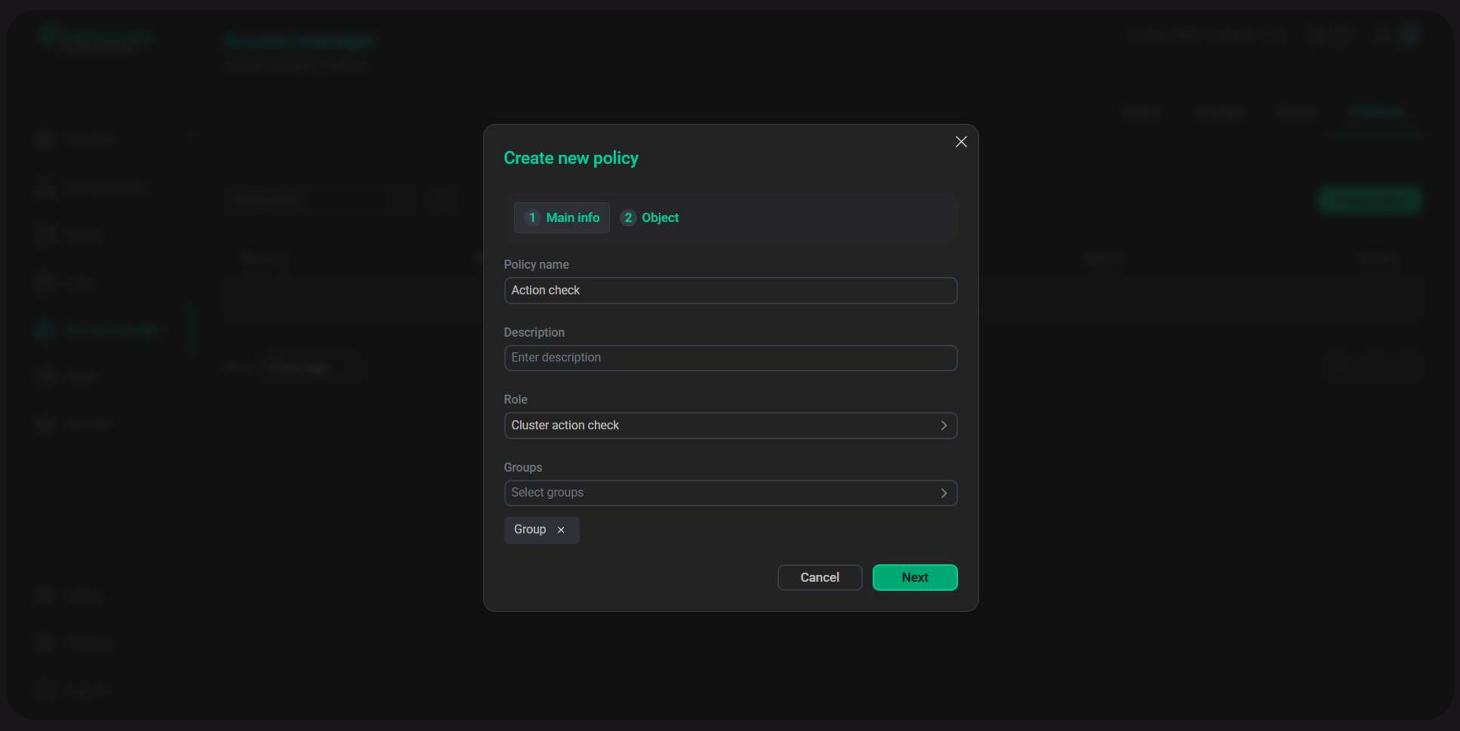
Cluster Action: Check

Service Action: Check

The Cluster action check role

To implement the `Cluster action check` role, create a new policy by following the steps below:

1. In the ***Create new policy*** window on the ***Policies*** tab enter `Action check` as a policy name.
2. Select the `user` user to be affected by this policy.
3. Click ***Next***.


Create new policy
1 Main info
2 Object
Cancel
Next

Policy name: Action check

Description: Enter description

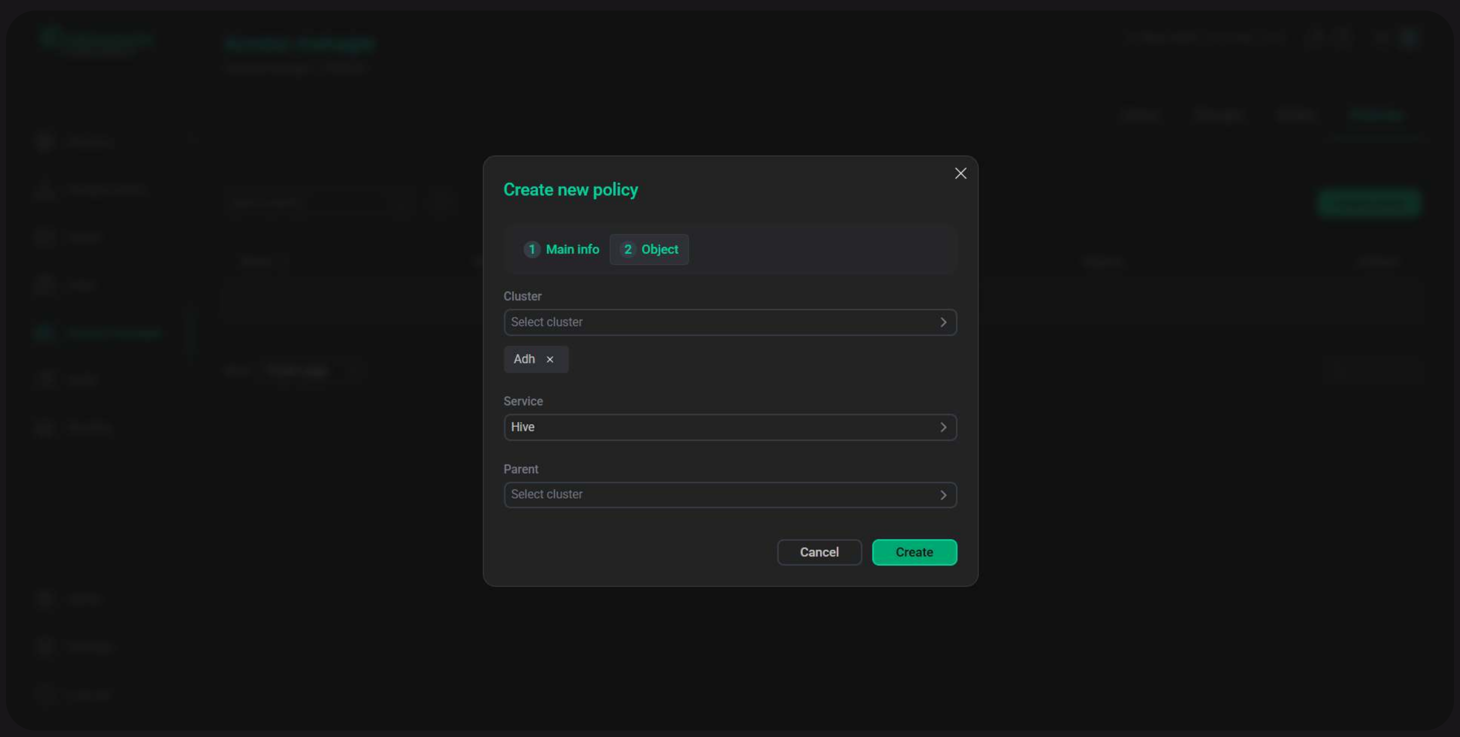
Role: Cluster action check

Groups: Select groups

Group ×

The first step of creating the Action check policy

4. Select the `Hive` service and the `ADH` cluster for the `Action check` policy.
5. Click ***Create***.


Create new policy
1 Main info
2 Object
Cancel
Create

Cluster: Select cluster

Adh ×

Service: Hive

Parent: Select cluster

The second step of creating the Action check policy

After the `Action check` policy is created, the `user` is able to run check actions for the `ADH` cluster and the `Hive` service.

If the `Service Action: Check` permission was not included into the `Cluster action check` role, you will not be able to select any service when creating the `Action check` policy. The objects available for selection when creating a new policy depend on the permissions included in the role a new policy is based on.

Configure user permissions

Mikhail Serov

Contents

To Table of Contents

Configure cluster administrator permissions

Configure service administrator permissions

Configure read-only permissions

ADCM role model is designed for configuring user permissions via policies. As a result of a policy being applied the user has access only to the actions and ADCM objects that are included in that policy.

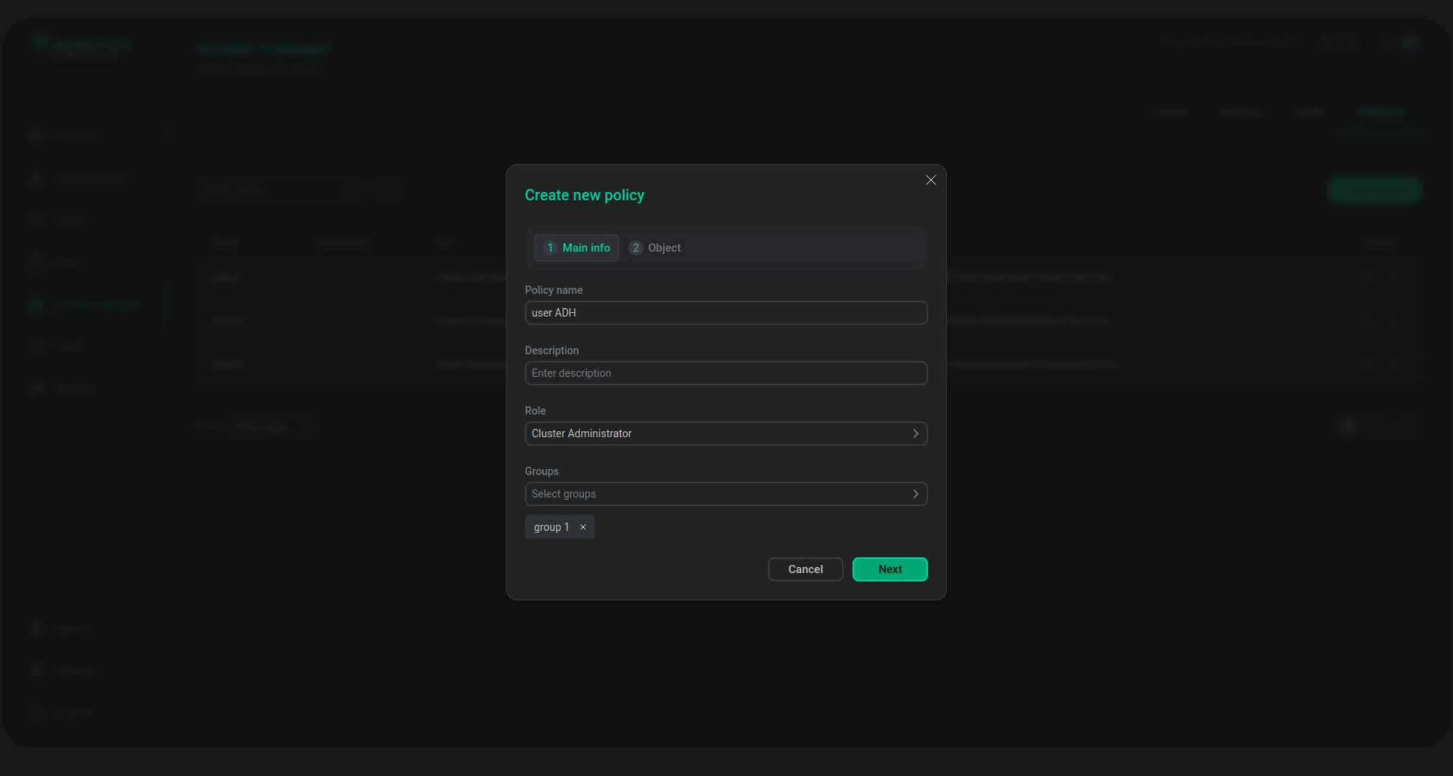
The ADCM offers a few ready-made user roles (sets of rules). You also can create your own roles to get the most suitable roles for your tasks.

Below are examples of configuring user permissions with the help of created policies.

Configure cluster administrator permissions

Let there exist two ADCM clusters named `ADH` and `ADH Prod`, and an ADCM user named `user`, who belongs to the `group 1` group and is not an administrator of any cluster. To create a policy that allows `user` to obtain cluster administrator permissions for the `ADH` cluster, do the following:

1. Enter the policy name — `user ADH`.
2. Select the `Cluster Administrator` role and the `group 1` group.



Create new policy
1 Main info
2 Object
Cancel
Next

Policy name: user ADH

Description: Enter description

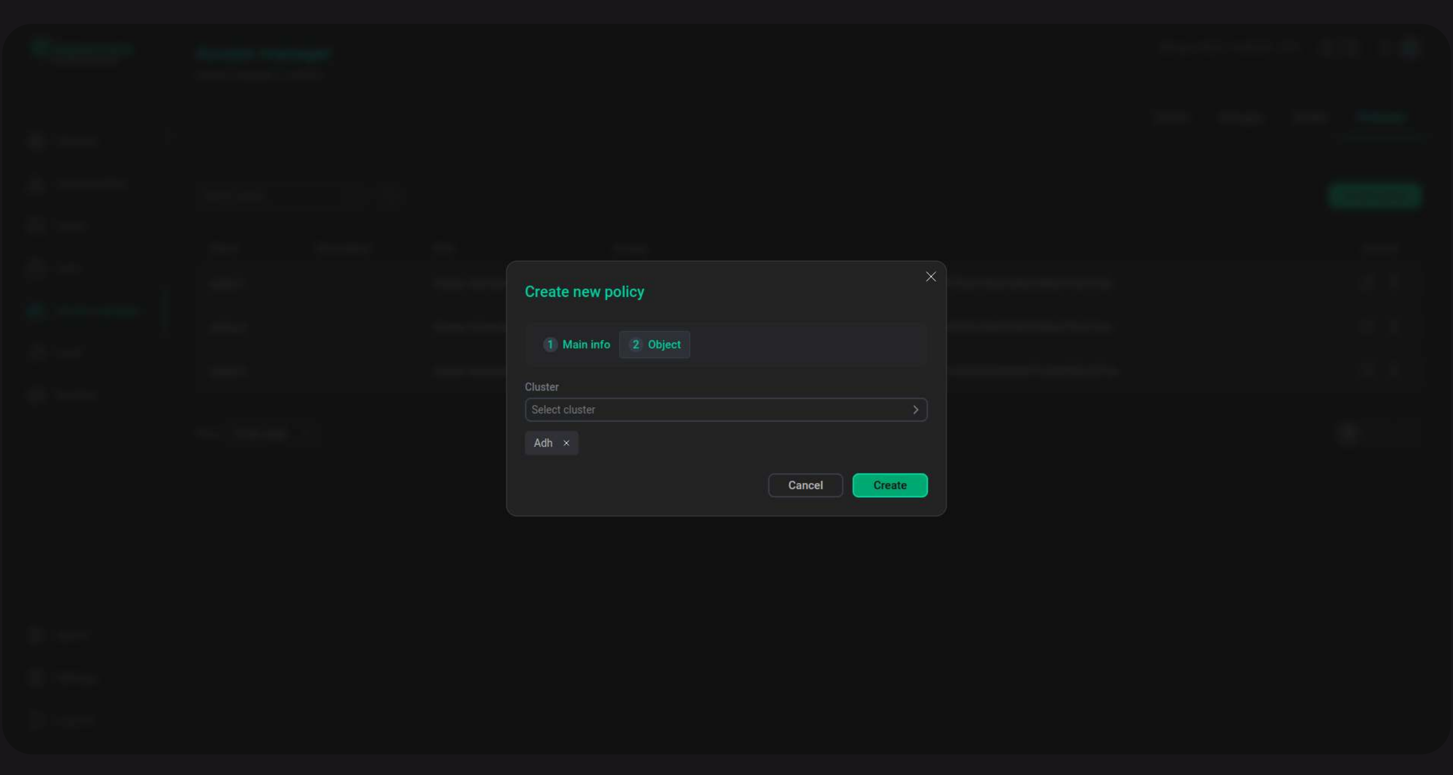
Role: Cluster Administrator

Groups: Select groups

group 1

The first step of creating a policy for the user

3. Select the `ADH` cluster.



Create new policy
1 Main info
2 Object
Cancel
Create

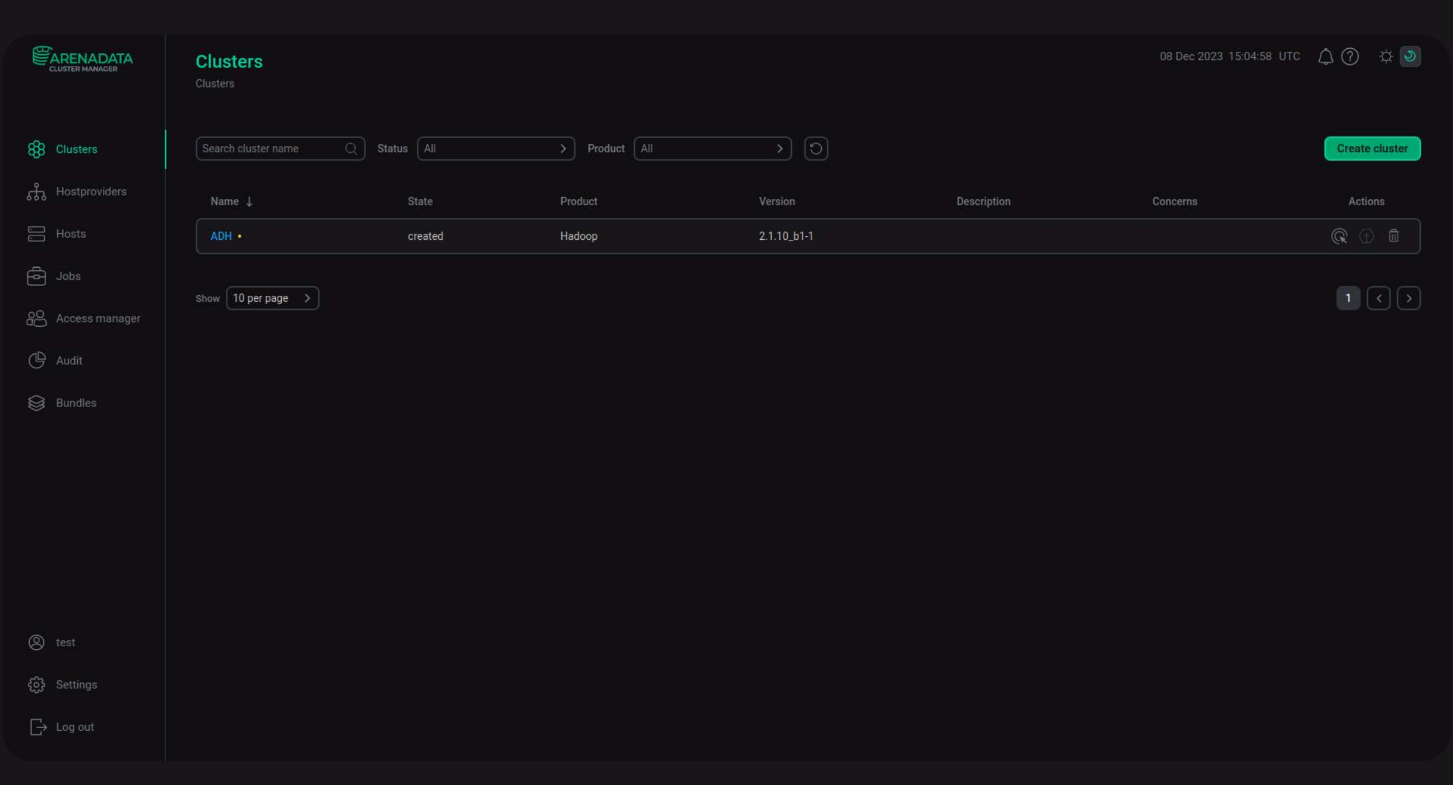
Cluster: Select cluster

Adh

The second step of creating a policy for the user

4. Apply the policy by clicking ***Create***.

After the policy is applied, `user` is able to see only the `ADH` cluster on the ***Clusters*** page. The `user` also has the cluster administrator permissions for the `ADH` cluster.



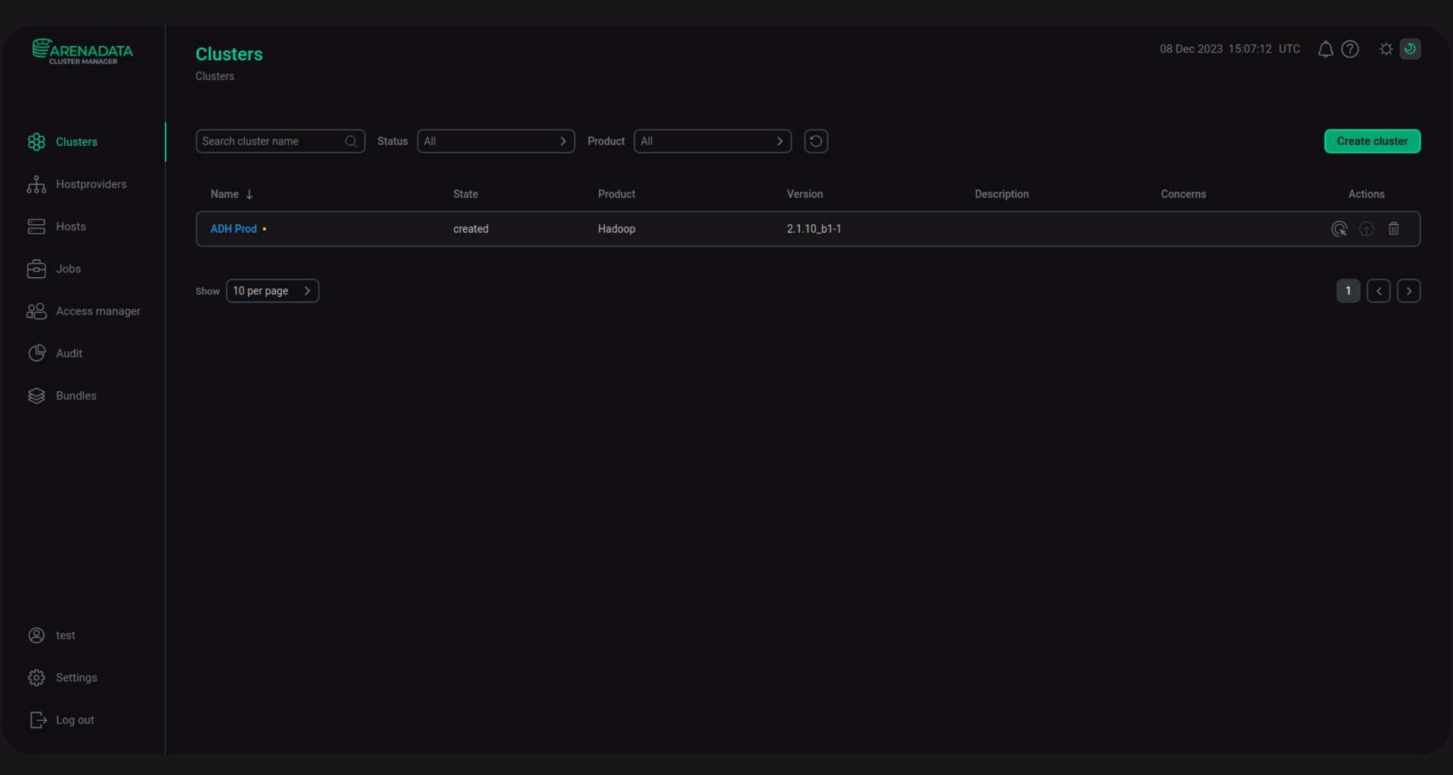
ARENADATA CLUSTER MANAGER
Clusters
Clusters
08 Dec 2023 15:04:58 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
test
Settings
Log out

Name	State	Product	Version	Description	Concerns	Actions
ADH	created	Hadoop	2.1.10_b1-1			

The Clusters page as seen by the user

Let there exist an ADCM user named `another_user`, who belongs to the `group 2` group and is not an administrator of any cluster. Make that user the `ADH Prod` cluster administrator via the actions similar to the actions above.

After the policy is applied, `another_user` is able to see only the `ADH Prod` cluster on the ***Clusters*** page. `another_user` also has the cluster administrator permissions for the `ADH Prod` cluster.

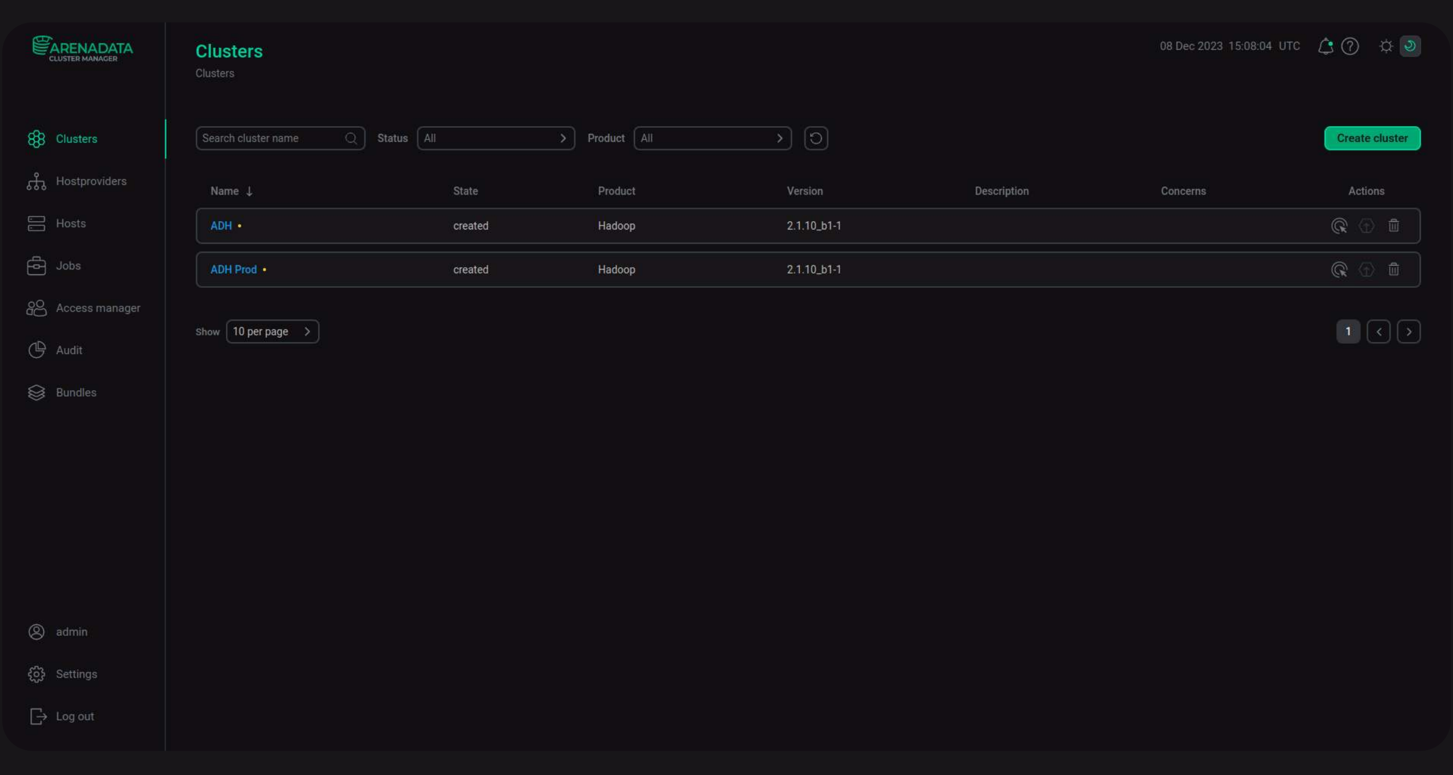


ARENADATA CLUSTER MANAGER
Clusters
Clusters
08 Dec 2023 15:07:12 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
test
Settings
Log out

Name	State	Product	Version	Description	Concerns	Actions
ADH Prod	created	Hadoop	2.1.10_b1-1			

The Clusters page as seen by another_user

The ADCM administrator is able to view all clusters in the system.



ARENADATA CLUSTER MANAGER
Clusters
Clusters
08 Dec 2023 15:08:04 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search cluster name
Status All
Product All
Create cluster
Show 10 per page
admin
Settings
Log out

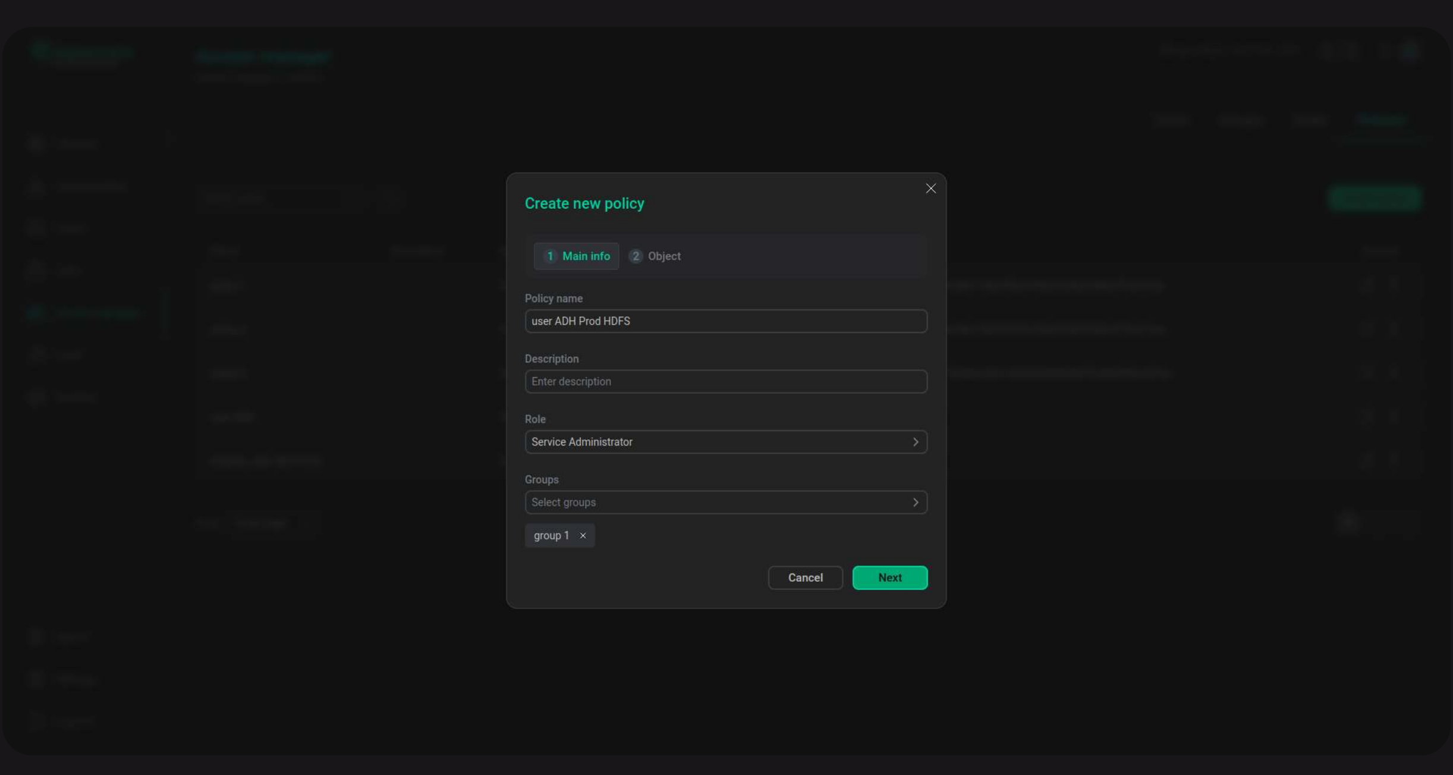
Name	State	Product	Version	Description	Concerns	Actions
ADH	created	Hadoop	2.1.10_b1-1			
ADH Prod	created	Hadoop	2.1.10_b1-1			

The Clusters page as seen by the ADCM administrator

Configure service administrator permissions

Let there exist an ADCM cluster named `ADH Prod` that has the `HDFS` service among others. Also let there exist an ADCM user named `user`, who belongs to the `group 1` group and is not an administrator of any service. To create a policy that allows `user` to obtain service administrator permissions for the `HDFS` service in the `ADH Prod` cluster, do the following:

1. Enter the policy name — `user ADH Prod HDFS`.
2. Select the `Service Administrator` role and the `group 1` group.



Create new policy
1 Main info
2 Object
Cancel
Next

Policy name: user ADH Prod HDFS

Description: Enter description

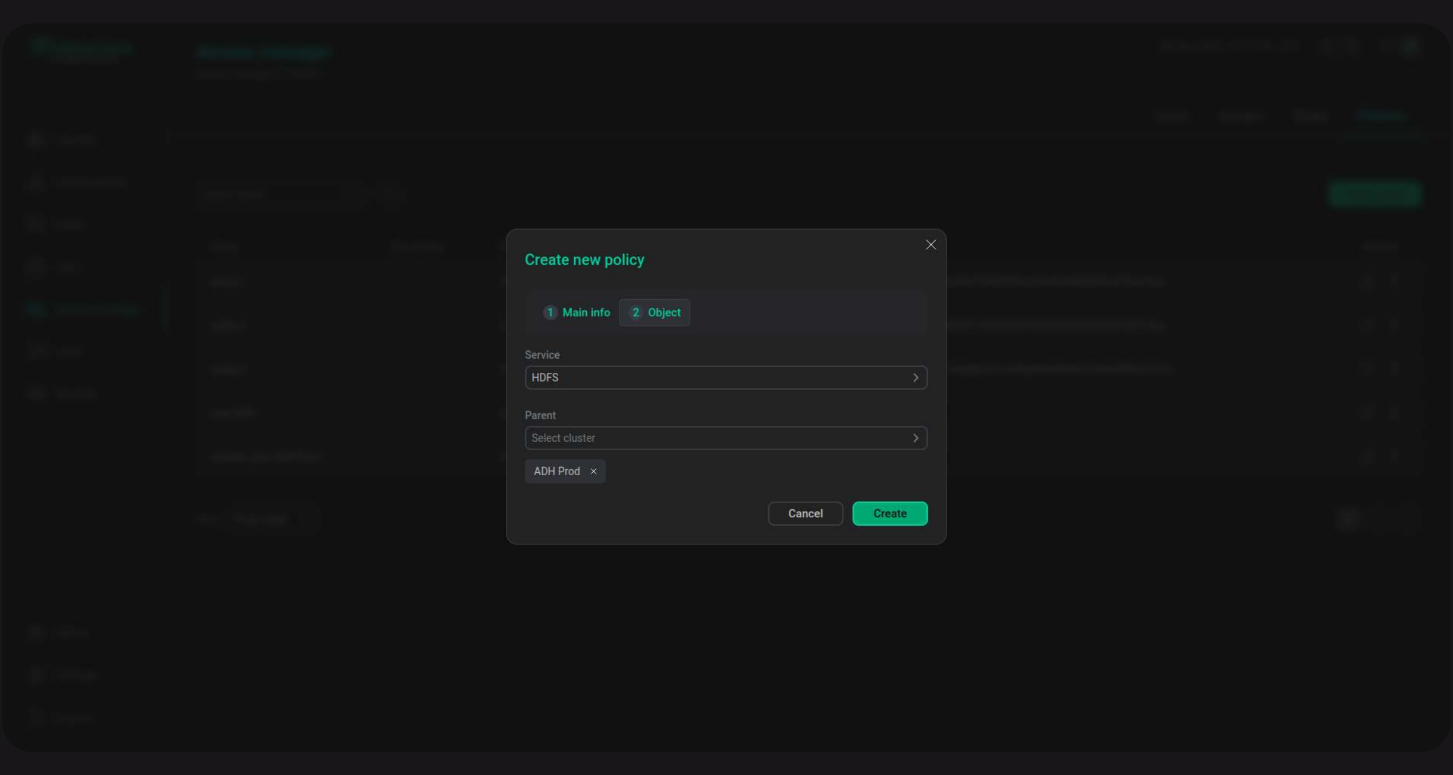
Role: Service Administrator

Groups: Select groups

group 1

The first step of creating a policy for the user

3. Select the `HDFS` service in the `ADH Prod` cluster.



Create new policy
1 Main info
2 Object
Cancel
Create

Service: HDFS

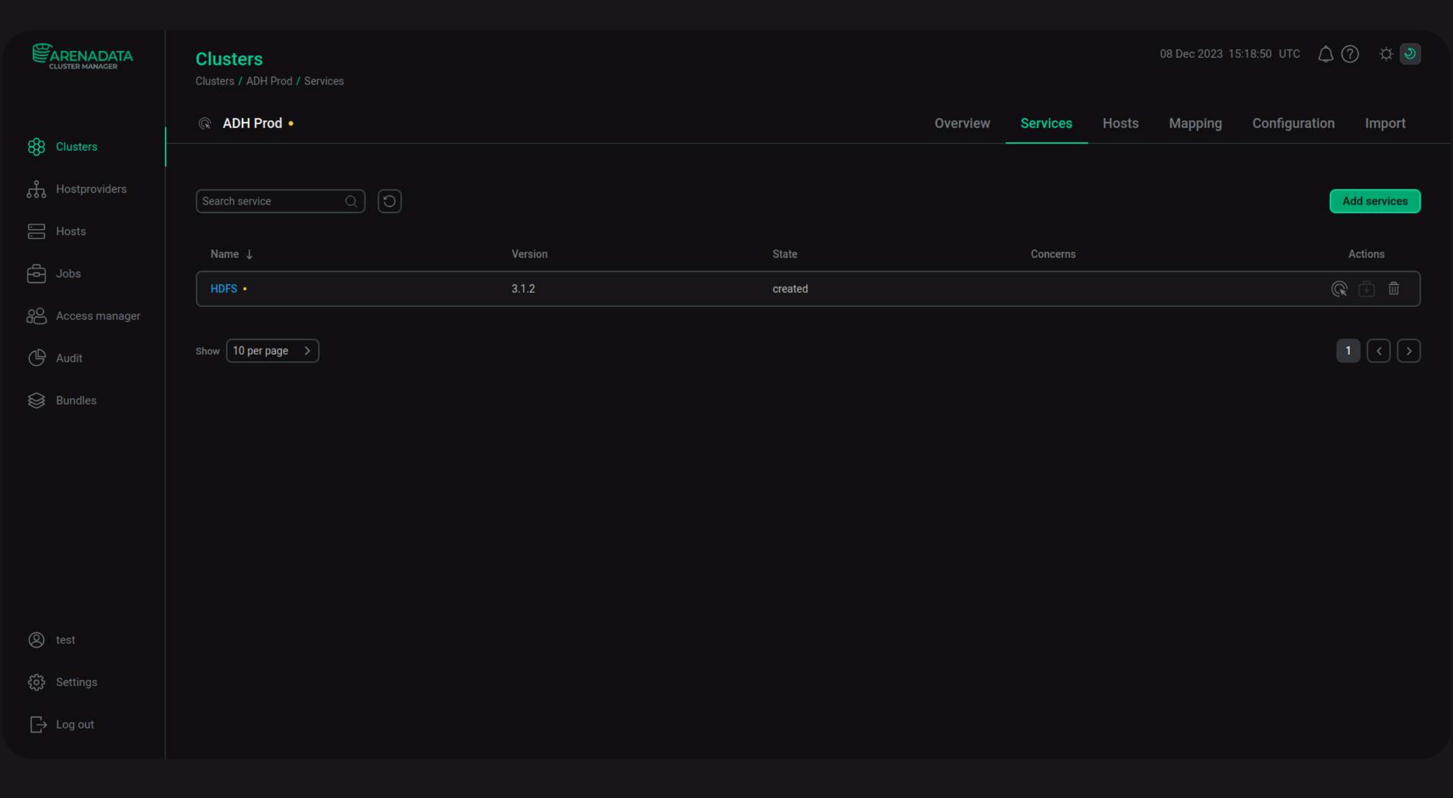
Parent: Select cluster

ADH Prod

The second step of creating a policy for the user

4. Apply the policy by clicking ***Create***.

After the policy is applied, `user` is able to see only the `HDFS` service on the ***Services*** tab of the `ADH Prod` cluster page. `user` also has the service administrator permissions for the `HDFS` service.

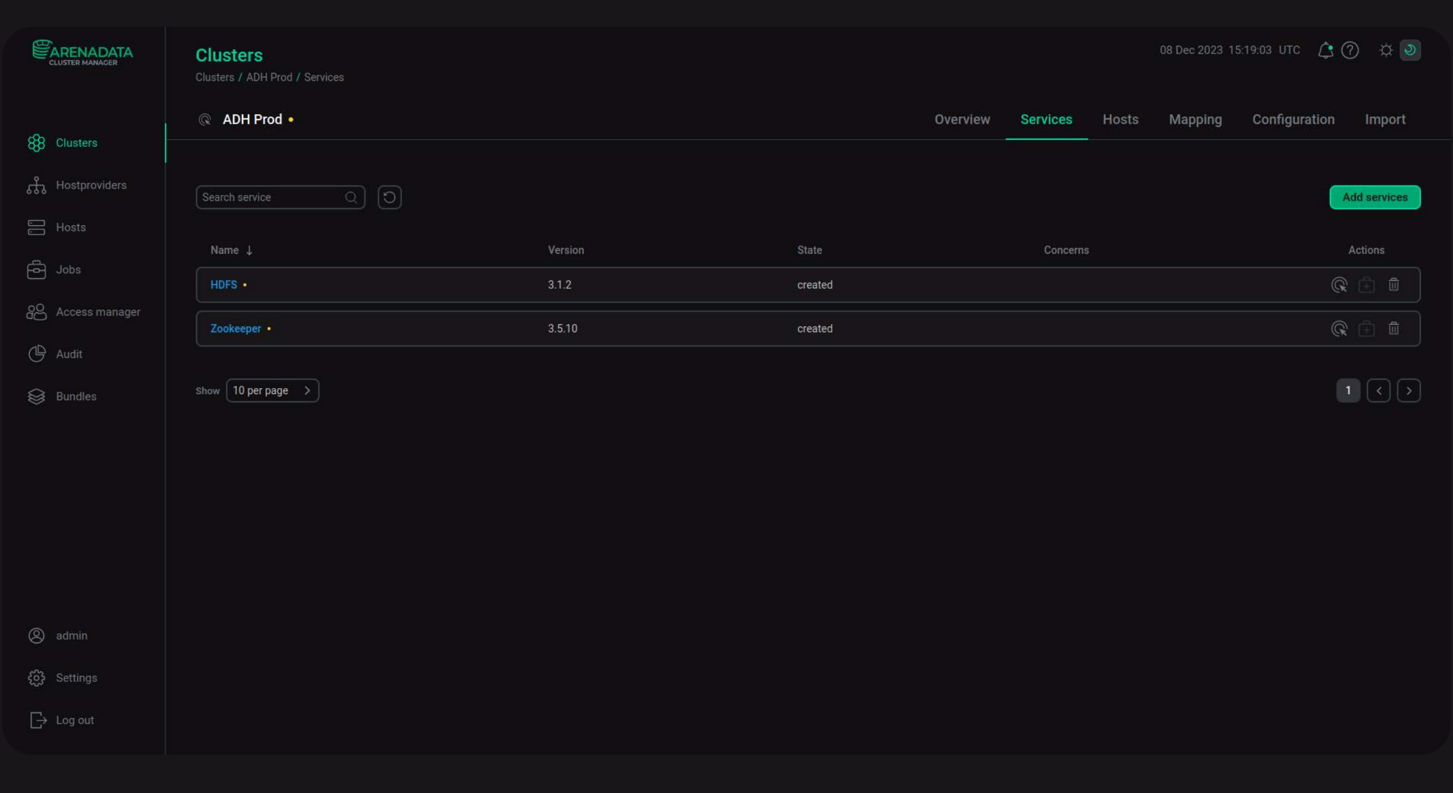


ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH Prod / Services
08 Dec 2023 15:18:50 UTC
ADH Prod
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add services
Show 10 per page
test
Settings
Log out

Name	Version	State	Concerns	Actions
HDFS	3.1.2	created		

The Services tab as seen by user

The ADCM administrator is able to view all services of the `ADH Prod` cluster.



ARENADATA CLUSTER MANAGER
Clusters
Clusters / ADH Prod / Services
08 Dec 2023 15:19:03 UTC
ADH Prod
Overview
Services
Hosts
Mapping
Configuration
Import
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Search service
Add services
Show 10 per page
admin
Settings
Log out

Name	Version	State	Concerns	Actions
HDFS	3.1.2	created		
Zookeeper	3.5.10	created		

The Services tab as seen by the ADCM administrator

Configure read-only permissions

The `ADCM User` role allows the user to view ADCM objects along with their configurations and components, yet does not allow the user to make any edits. This mode is, in fact, a read-only mode. The `ADCM User` role is available when creating policies likewise in the same way as above.

Users management

Mikhail Serov

Contents

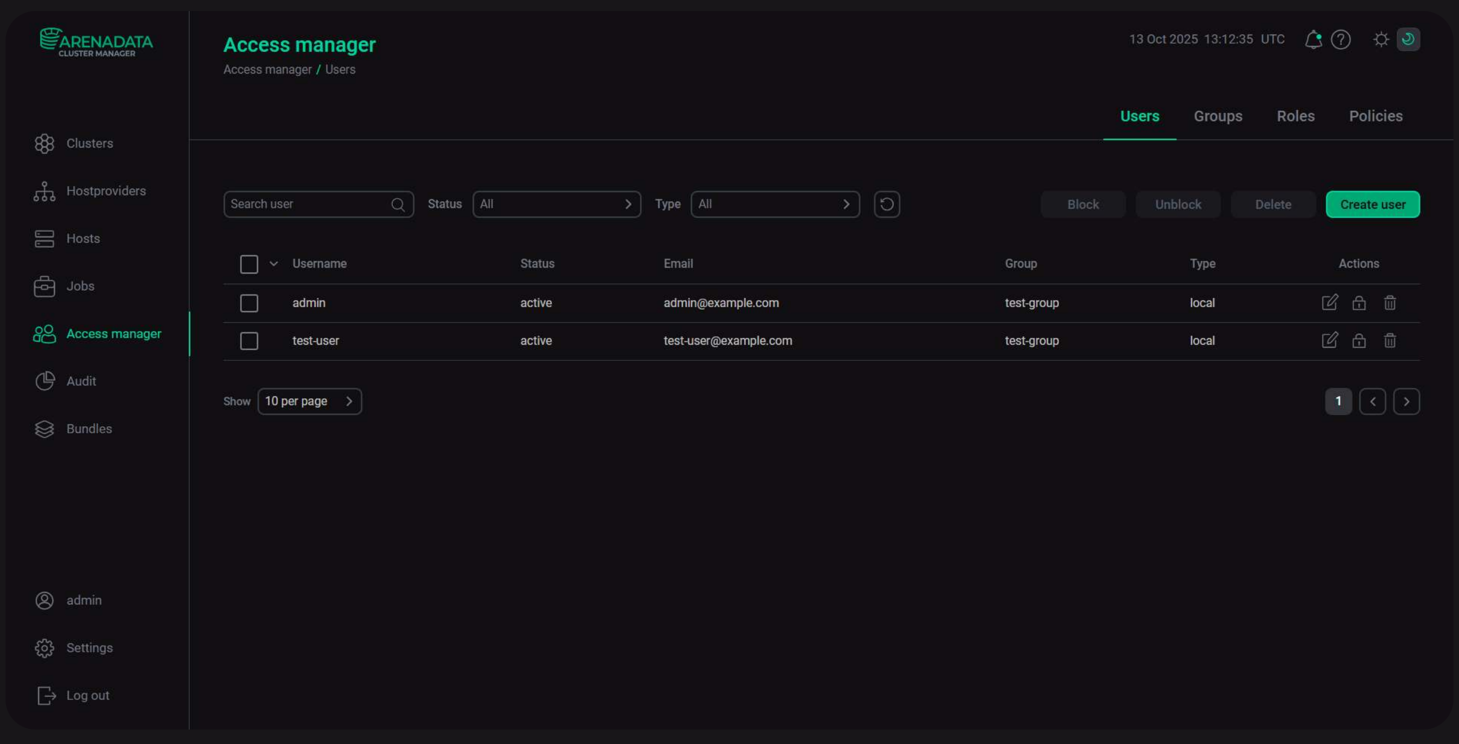
To Table of Contents

View, add, and edit users

Block users

View, add, and edit users

An ADCM user who has the rights to create, edit, and delete users can view existing users on the ***Users*** tab of the ***Access Manager*** page.

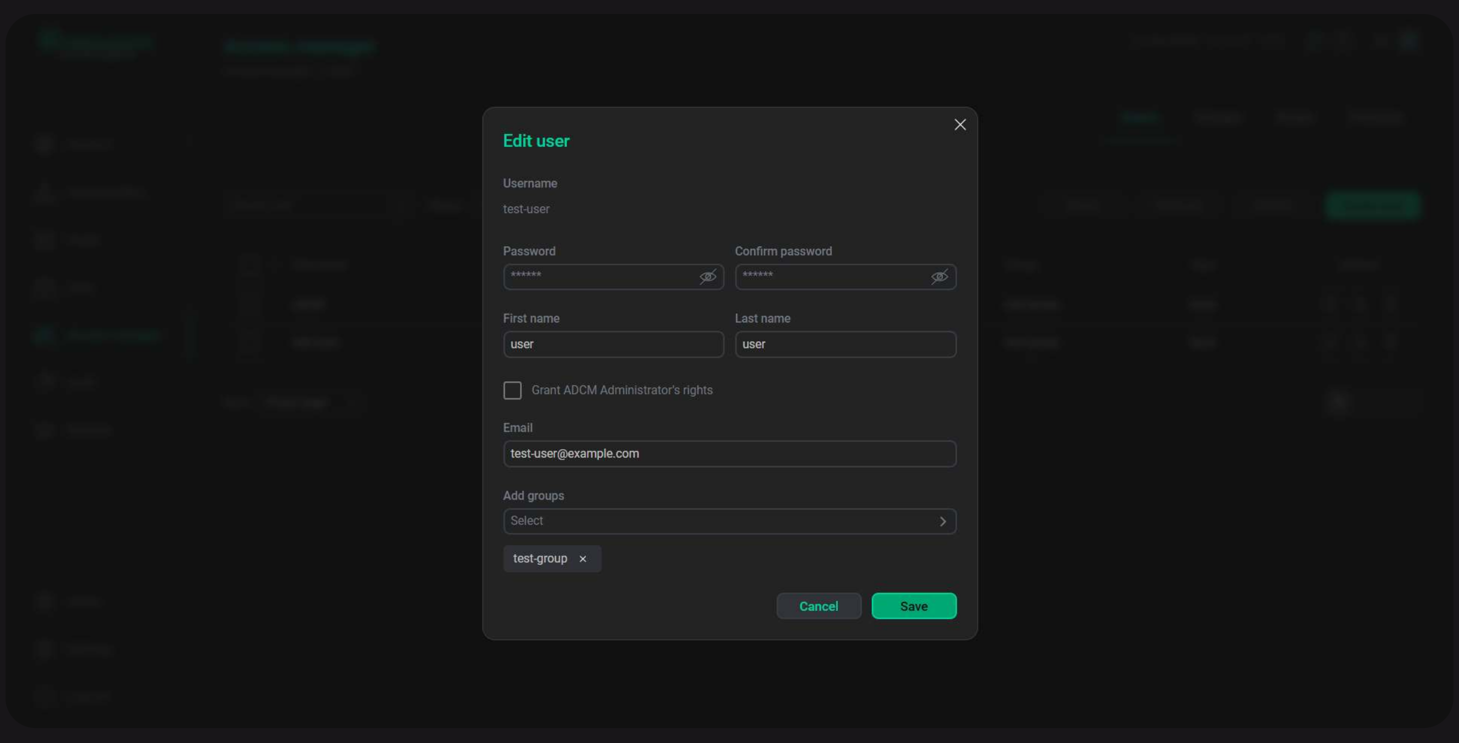

ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Access manager
Access manager / Users
13 Oct 2025 13:12:35 UTC
Users
Groups
Roles
Policies
Search user
Status
All
Type
All
Block
Unblock
Delete
Create user
Show
10 per page
1

☐	Username	Status	Email	Group	Type	Actions
☐	admin	active	admin@example.com	test-group	local	
☐	test-user	active	test-user@example.com	test-group	local	

The Users tab

An ADCM user who has the rights to create, edit, and delete users can add new users using ***Create user*** and remove users using ***Delete*** or the 🗑 icon located at the right part of the corresponding row.

An ADCM user who has the rights to create, edit, and delete users also can edit user parameters via the ***Edit user*** window. This window can be accessed via the ✎ icon also located at the right part of the corresponding row.


Edit user
Cancel
Save

Username: test-user

Password: ******

Confirm password: ******

First name: user

Last name: user

☐ Grant ADCM Administrator's rights

Email: test-user@example.com

Add groups: Select

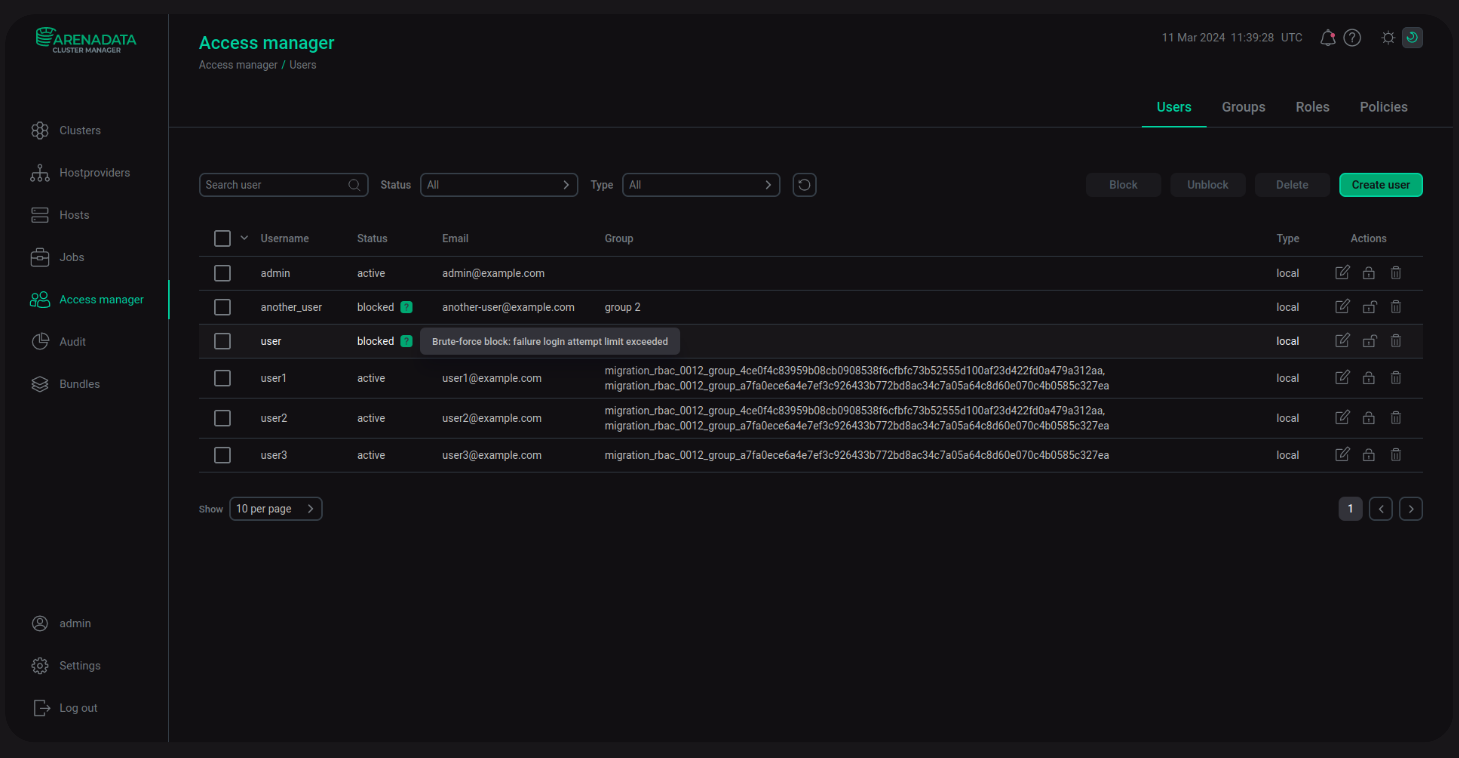
test-group ×

Edit user window

Block users

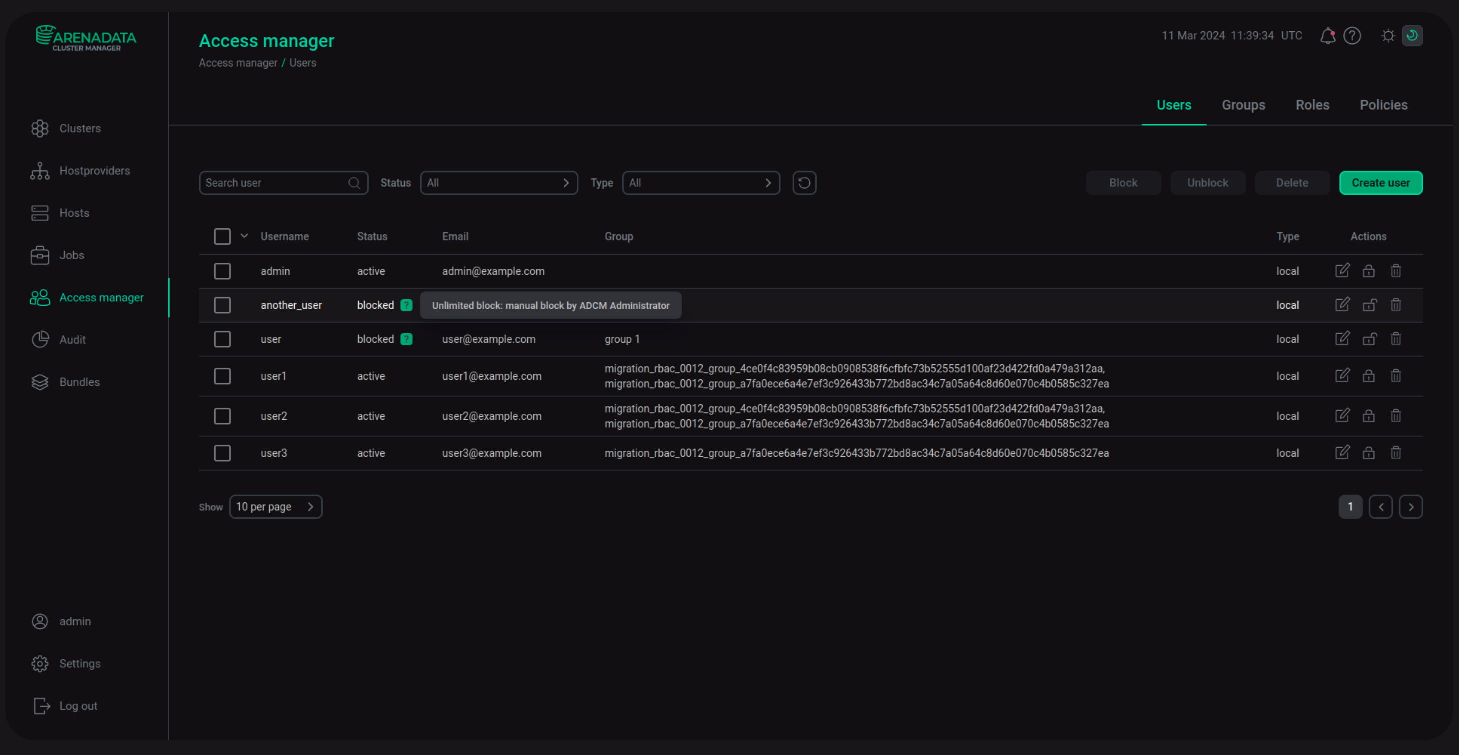
An ADCM user can be blocked. There are two reasons for a user to be blocked:

- Exceeding the number of failed login attempts. The user has exceeded the limit of login attempts and was automatically blocked by ADCM.
- The user was manually blocked by ADCM Administrator.


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Access manager
Access manager / Users
11 Mar 2024 11:39:28 UTC
Users
Groups
Roles
Policies
Search user
Status
All
Type
All
Block
Unblock
Delete
Create user
Show
10 per page
1

☐	Username	Status	Email	Group	Type	Actions
☐	admin	active	admin@example.com		local	
☐	another_user	blocked ?	another-user@example.com	group 2	local	
☐	user	blocked ?	Brute-force block: failure login attempt limit exceeded		local	
☐	user1	active	user1@example.com	migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa, migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local	
☐	user2	active	user2@example.com	migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa, migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local	
☐	user3	active	user3@example.com	migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local	

Example of blocking in case of exceeding the number of failed login attempts


ARENADATA
CLUSTER MANAGER
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Access manager
Access manager / Users
11 Mar 2024 11:39:34 UTC
Users
Groups
Roles
Policies
Search user
Status
All
Type
All
Block
Unblock
Delete
Create user
Show
10 per page
1

☐	Username	Status	Email	Group	Type	Actions
☐	admin	active	admin@example.com		local	
☐	another_user	blocked ?	Unlimited block: manual block by ADCM Administrator		local	
☐	user	blocked ?	user@example.com	group 1	local	
☐	user1	active	user1@example.com	migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa, migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local	
☐	user2	active	user2@example.com	migration_rbac_0012_group_4ce0f4c83959b08cb0908538f6cfbfc73b52555d100af23d422fd0a479a312aa, migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local	
☐	user3	active	user3@example.com	migration_rbac_0012_group_a7fa0ece6a4e7ef3c926433b772bd8ac34c7a05a64c8d60e070c4b0585c327ea	local	

Example of manual blocking

NOTE

Only ADCM Administrator has the ability to manually block other users. ADCM Administrators cannot block themselves. Manual blocking is also unavailable for LDAP users.

To block a user, ADCM Administrator can select that user in the list and then either use the ***Block*** button, or the 🔒 icon. Users can be manually blocked even if they were already blocked for exceeding the limit of failed login attempts. In this case, the reason for the block changes from exceeding the number of login attempts to manual blocking performed by the ADCM Administrator.

To unblock a user, ADCM Admnistrator can select that user in the list and then either use the ***Unblock*** button, or the 🔓 icon. After being unblocked, the user will be able to access their account. The number of failed login attempts (if any) will be reset to 0.

Enable HTTPS

Mikhail Serov

To Table of Contents

To enable HTTPS, perform the following steps:

1. Prepare a server private key in PEM format and place it to the *<data_volume>/conf/ssl/key.pem* file.
2. Prepare a certificate chain in PEM format and place it to the *<data_volume>/conf/ssl/cert.pem* file.
3. Drop the current ADCM container and create it again with an SSL port.

```
$ docker rm adcm
$ docker create --name adcm -p 8000:8000 -p 8443:8443 -v /opt/adcm:/adcm/data 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`.

NOTE

Order of certs in the *cert.pem* file should be in sync with Nginx documentation.

Enable LDAP

Mikhail Serov

Contents

To Table of Contents

Definitions
LDAP description
Workflow

Definitions

This page includes the following terminology:

- **Active Directory (AD)** is an implementation of directory services that provides all kinds of functionality, such as authentication, group and user management, policy administration, and more.
- **Lightweight Directory Access Protocol (LDAP)** is an open cross-platform protocol designed for directory services authentication.
- **OpenLDAP** is an open implementation of LDAP that is distributed via its own OpenLDAP Public License.
- **Free Identity, Policy and Audit (FreeIPA)** is an open centralized system that is designed to manage user authorizations and to create audit and access policies for the UNIX-based networks.

IMPORTANT

- LDAP integration is supported only for a single domain.
- Username is unique. If there is a local user, the external user with a similar name cannot be added during synchronization.
- For LDAP users, only group membership attributes (the ***Add groups*** field) can be modified during synchronization. If there were changes related to LDAP group membership on the AD side for an LDAP user, these changes will be added each time the ***Run LDAP sync*** action is performed (including when this action is executed according to a schedule). If an LDAP user is added to any local ADCM group, this change will be preserved during synchronization.

LDAP description

ADCM provides tools to synchronize its users and groups with external systems that support LDAP (such as Active Directory, OpenLDAP, or FreeIPA). This allows automating the process of creating ADCM user accounts according to the already existing company policies, as well as to apply the existing ADCM access control methods (granting and revoking rights, changing group memberships, and so on) to external users.

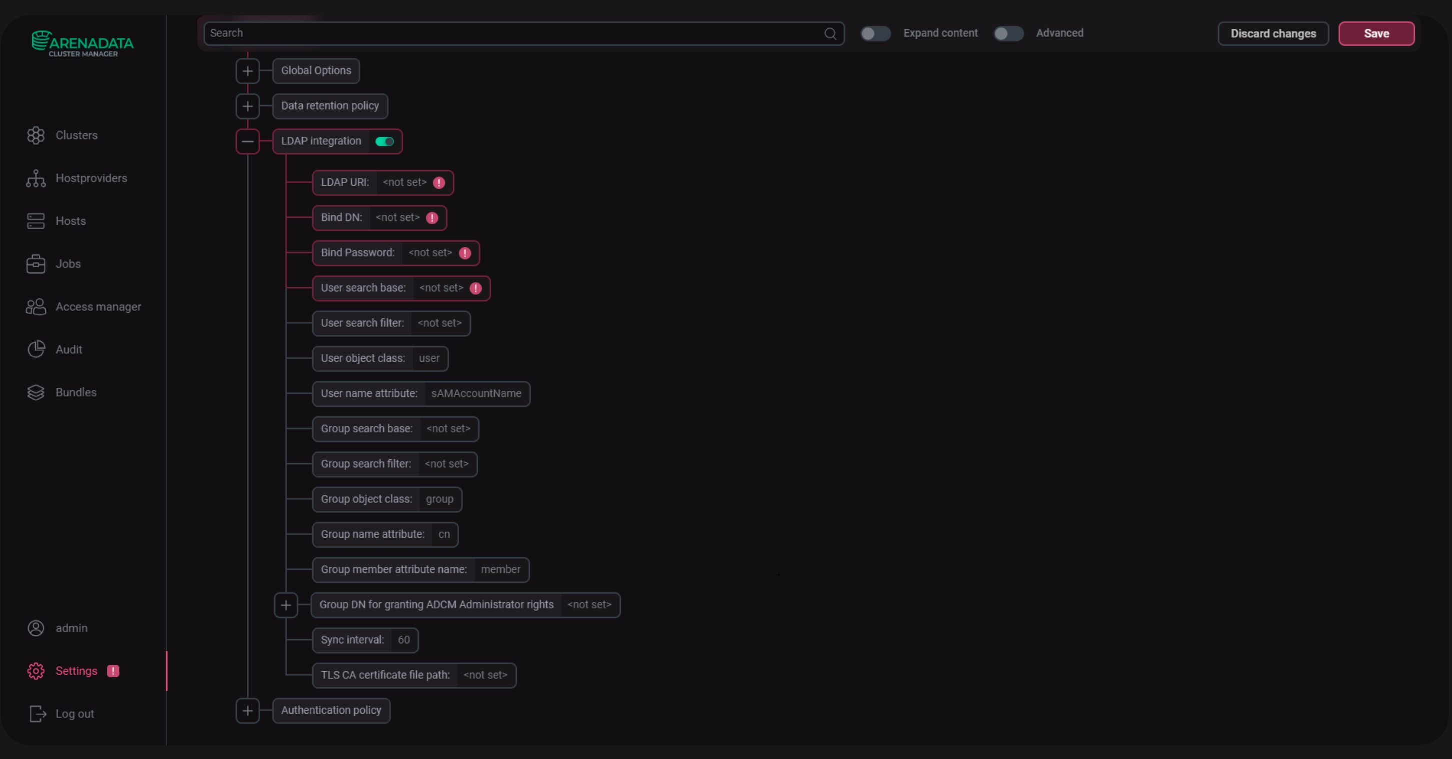
After configuring LDAP in ADCM, you are able to do the following:

- Automatically update user information (such as credentials) in case of it being changed in AD, OpenLDAP, or FreeIPA.
- Automatically deactivate user accounts in case of them being disabled in AD, OpenLDAP, or FreeIPA with ADCM.
- Automatically add new users if they are aligned with LDAP configuration settings.

Workflow

To enable LDAP integration, perform the following steps:

1. Navigate to the ***Settings*** page.
2. Activate the ***LDAP integration*** switch.
3. Configure LDAP settings that are listed in the Configuration settings table below.



LDAP URI: <not set>
Bind DN: <not set>
Bind Password: <not set>
User search base: <not set>
User search filter: <not set>
User object class: user
User name attribute: sAMAccountName
Group search base: <not set>
Group search filter: <not set>
Group object class: group
Group name attribute: cn
Group member attribute name: member
Group DN for granting ADCM Administrator rights <not set>
Sync interval: 60
TLS CA certificate file path: <not set>

LDAP integration parameters

4. Click ***Save***. Now two actions become possible: ***Run LDAP sync*** and ***Test LDAP connection***.
5. Run the ***Test LDAP connection*** action to test connection to LDAP. You can view the action result on the ***Jobs*** page.
6. Run the ***Run LDAP sync*** action to synchronize the users, groups, and their memberships from Active Directory, OpenLDAP, or FreeIPA into ADCM. You can view the action result on the ***Jobs*** page.

NOTE

If LDAP access is configured correctly, then any user that matches the configured LDAP settings is able to authenticate in ADCM (even if the ***Run LDAP sync*** action was not performed).

Configuration settings

Name	Description	Required
LDAP URI	URI of the LDAP server	Yes
Bind DN	Distinguished name (DN) of the user that is used to connect to the LDAP server. For example: `cn=admin,dc=ad,dc=ranger-test`	Yes
Bind Password	Password for access to the LDAP server of the user specified in the ***Bind DN*** field	Yes
User search base	Distinguished name of the directory object from which to search for entries. For example: `ou=Peoples,dc=ad,dc=ranger-test`	Yes
User search filter	Additional filter constraining the users selected for syncing	No
User object class	Object class to identify user entries. The default value is `user`	Yes
User name attribute	Attribute from user entry that would be treated as user login. The default value is `sAMAccountName`	Yes
Group search base	Distinguished name of the directory object from which to search for entries. For example: `ou=Groups,dc=ad,dc=ranger-test`	No
Group search filter	Additional filter constraining the groups selected for syncing	No
Group object class	Object class to identify group entries. The default value is `group`	Yes
Group name attribute	Attribute from group entry that would be treated as group name. The default value is `cn`	Yes
Group member attribute name	Attribute from group entry that is list of members. The default value is `member`	Yes
Group DN for granting the ADCM Administrator rights	List of unique distinguished name groups whose members need to be granted ADCM Administrator rights. For example: `cn=adcm_admin,ou=Groups,dc=ad,dc=ranger-test`	No
Sync interval	Amount of time in minutes that groups, group memberships, and user distinguished names are synchronized between Active Directory, OpenLDAP, or FreeIPA and ADCM. The default value is `60`	Yes
TLS CA certificate file path	CA certificate path in the ADCM container volume	No

Work with an external database

Mikhail Serov

Contents

To Table of Contents

Configuration

Migration

ADCM supports operations with external databases. In particular, PostgreSQL databases are supported.

IMPORTANT

A standalone PostgreSQL database is recommended for use in the product environment.

Configuration

The prerequisites to use the external PostgreSQL database are:

- The user, the database, and the user password must be created (`DATABASE_USERNAME`, `DATABASE_NAME`, `DATABASE_USER_PASSWORD` in the example below).
- The user must be the owner of the external PostgreSQL database.

```
CREATE USER <DATABASE_USERNAME> with encrypted password '<DATABASE_USER_PASSWORD>';
CREATE DATABASE <DATABASE_NAME> OWNER <DATABASE_USERNAME>;
```

Example of running ADCM with the PostgreSQL database:

```
$ sudo docker run -d --restart=always -p 8000:8000 -v /opt/adcm:/adcm/data -e DB_HOST="10.92.40.190" -e DB_PORT="5432" -e
DB_USER="admin" -e DB_NAME="postgres" -e DB_PASS="admin" --name adcm hub.arenadata.io/adcm/adcm:<version> -e DB_OPTIONS="
<DATABASE_OPTIONS>"
```

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`.

You can use `DB_OPTIONS` to establish an SSL connection to the database. For more information about the parameters available for the `DB_OPTIONS` variable, see the Installation article.

Prepare the SSL certificate and secret key files and place them into *<data_volume>/conf/ssl/*. Then, list the `DB_OPTIONS` parameters in JSON format:

```
DB_OPTIONS='{"sslmode": "verify-full", "sslcert": "/adcm/data/conf/ssl/client.crt", "sslkey": "/adcm/data/conf/ssl/client.key",
"sslrootcert": "/adcm/data/conf/ssl/server.crt"}'
```

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`.

Use `-v /opt/adcm:/adcm/data:Z` for SELinux.

Migration

IMPORTANT

- Starting with version 2.7.0, ADCM no longer supports SQLite.
- ADCM migration feature is available up to version 2.6.0 inclusively and not supported for ADCM versions 2.7.0 and higher.
- Migration must be performed within a single version of ADCM.

ADCM includes tools for data migration from SQlite to PostgreSQL. This functionality is supported from the version 2023.03.27.20 onwards. The functionality is useful if you have upgraded ADCM to the newest version and want to start using PostgreSQL.

To migrate from a source SQlite database to a target PostgreSQL database, you must fully prepare the target PostgreSQL database first.

After you have updated the ADCM docker container to the newest version, you can migrate from SQLite to PostgreSQL via the following commands:

1. Dump the SQLite database to a file:

```
$ docker exec -it adcm /adcm/python/manage.py dumpdata --natural-foreign --natural-primary -o /adcm/data/var/data.json
```

   where `-o` specifies the file path.

   Essentially, this command requests all data from the SQlite database, then creates a JSON representation of that data, and places it to the specified file system location as a *data.json* file.

2. Stop the ADCM container:

```
$ docker stop adcm
$ docker rm adcm
```

3. Start the ADCM container in `MIGRATION_MODE`:

```
$ docker run -d --restart=always -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 MIGRATION_MODE=1
--name adcm 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`.

   For SELinux, use `-v /opt/adcm:/adcm/data:Z`.

   Starting the ADCM container like this, you obtain an empty ADCM instance that has not yet been filled with data.

4. Load the data from a file to the PostgreSQL database:

```
$ docker exec -it adcm /adcm/python/manage.py loaddata /adcm/data/var/data.json
```

5. Stop the ADCM container:

```
$ docker stop adcm
$ docker rm adcm
```

6. Start the ADCM container:

```
$ docker run -d --restart=always -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 MIGRATION_MODE=0
--name adcm hub.arenadata.io/adcm/adcm:<version>
```

Start and stop

Daria Barysheva

To Table of Contents

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.

Backup and restore

Mikhail Serov, Daria Barysheva

Contents

To Table of Contents

Backup
Restore

IMPORTANT

If you are using an external database, you will need to make additional backups and restores on the side of that database.

Backup

To perform backup, you have to do the following:

1. Stop the ADCM container:

```
$ sudo docker stop adcm
```

2. Go to the container directory (by default, */opt/adcm*):

```
$ cd /opt/adcm
```

3. Create a backup for the container directory. The backup will be created in the home directory of the current user:

```
$ sudo tar czvf ~/backup.tar.gz ./
```

4. Start the ADCM container:

```
$ sudo docker start adcm
```

Restore

To perform restore, you have to do the following:

1. Stop the ADCM container:

```
$ sudo docker stop adcm
```

2. Create a new directory:

```
$ sudo mkdir /opt/adcm-restored
```

3. Go to the created directory:

```
$ cd /opt/adcm-restored
```

4. Restore data from the previously created backup to the new directory:

```
$ sudo tar xvf ~/backup.tar.gz
```

5. Remove the current ADCM container:

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

NOTE

If you do not need to reuse the `adcm` container name, you can skip the current step and use another container name instead of `adcm` in the next steps.

6. Create a new ADCM container (note that the */opt/adcm-restored* directory is used instead of */opt/adcm*):

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

where `<version>` is a desired ADCM image version. You can use a version number equal to or greater than the one for which the backup was taken. It is not recommended to update ADCM more than 5 minor versions ahead. ADCM downgrade is not supported. For more information, see Upgrade.

7. Start the new ADCM container:

```
$ sudo docker start adcm
```

Upgrade

Daria Barysheva, Adelya Zvyagintseva

Contents

To Table of Contents

Online update

Offline update

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
```

Configure

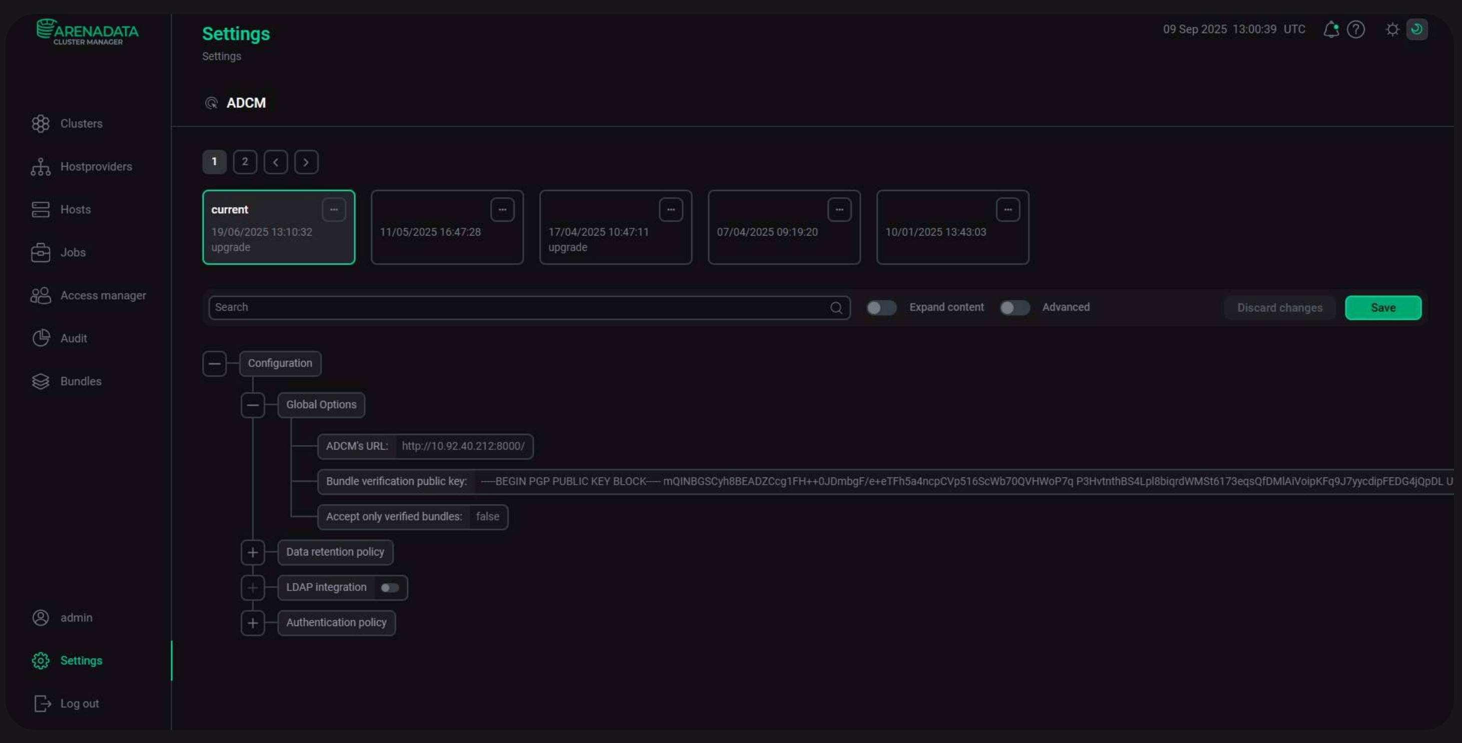
Mikhail Serov

Contents

To Table of Contents

- Global options
- Data retention policy
- LDAP integration
- Authentication policy
- Statistics collection

To configure ADCM, you can use the global configuration parameters that are located on the ***Settings*** page. The page can be accessed from the bottom left part of the screen.



ADCM's URL: http://10.92.40.212:8000/

Bundle verification public key: -----BEGIN PGP PUBLIC KEY BLOCK----- mQINBGSCyh8BEADZCcg1FH++0JDmbgF/e+eTFh5a4ncpCVp516ScWb70QVHWoP7q P3HvtnthBS4Lpl8biqrdWMSt6173eqsQfDMlAiVoipKFq9J7yycdipFEDG4jQpDL U

Accept only verified bundles: false

The Settings page

You can modify parameters and save them by clicking ***Save***. You can search for a parameter using the ***Search*** field. You can look through the ADCM previous configuration versions history and compare any two versions using the context menu in the version list located above the ***Search*** field.

Configuration parameters that are available for editing are listed in tables located in the sections below.

Global options

Right after you started ADCM, you should check if the host and port in use are specified correctly. You can do this in the ***Global Options*** section.

Parameter	Description
ADCM's URL	ADCM's URL to receive information from the host
Bundle verification public key	Bundle verification GPG public key to verify the bundle signature (the SIG file) when the bundle is uploaded to ADCM
Accept only verified bundles	Enables checking of a valid bundle signature when the bundle is uploaded to ADCM

Data retention policy

ADCM provides the way to erase the data related to jobs. Such data is, essentially, the job logs that are displayed on the ***Jobs*** tab. Time period can be specified for the database and for the container file system. When the specified time period to store the logs expires, the logs are deleted from the database and from the container file system. The default value for both time periods is `365` days.

ADCM also provides the way to erase the data related to objects (for example, cluster or service) configurations. The default value for the time period is `0` days. The `0` value means an unlimited storage period (configurations are not being deleted).

In addition, you can configure the data retention period for auditing operations and authorizations in ADCM.

Parameter	Description
Job log retention period from file system	Number of days after which the logs are deleted from the file system
Job log retention period from database	Number of days after which the logs are deleted from the database
Objects configurations retention period	Number of days after which the objects configuration is deleted from the database. `0` means infinite storing
Audit data retention period	Data storage period (in days) for operations and authorizations in ADCM
Enable audit data archiving	Enables archiving of data on operations and authorizations after the period specified in the ***Audit data retention period*** field

LDAP integration

See the configuration settings at the page about enabling LDAP.

Authentication policy

Below are the settings for the password policy. The password length must be between the specified minimum and maximum values. In addition, you can set the allowed number of login attempts to counteract brute-forcing methods. If a user exceeds the specified number of login attempts, a user account is blocked for a specified period of time.

After the user's blocking period expires, the login attempts counter for this user is reset to zero. Users with ADCM administrator rights have the ability to reset the login attempts counters of other users.

Parameter	Description
Minimum password length	Minimum allowed password length
Maximum password length	Maximum allowed password length
Login attempt limit	Number of login attempts allowed before the user account is locked
Blocking time period	Defines a period (in minutes) of inactivity after which old failed login attempts will be cleared

Statistics collection

ADCM is able to collect and transmit non-confidential statistics on user clusters created with the community edition bundles. Switch off this option if you do not want to send those statistics to Arenadata. The option is available if the ***Advanced*** switch is turned on.

Uninstall ADCM

Adelya Zvyagintseva

Contents

To Table of Contents

Uninstall via ADCM Installer

Manually uninstall

Depending on the selected ADCM installation method, the following options of removing ADCM are available:

- uninstall via ADCM Installer, if the ADCM installation was deployed using this tool;
- manually uninstall the ADCM container if ADCM was installed via a Docker Image.

Uninstall via ADCM Installer

To remove a deployed ADCM installation via ADCM Installer, run the following command:

```
$ adi delete --volumes --yes
```

NOTE

Note that when this command is executed, the Docker container and its associated volumes are removed. For more information on uninstalling via ADCM Installer, refer to the Use ADCM Installer article.

Manually uninstall

To manually uninstall ADCM, follow these steps:

1. Stop the Docker container `adcm`:

```
$ sudo docker stop adcm
```

2. Remove the Docker container `adcm`:

```
$ sudo docker rm adcm
```

3. Remove volumes associated with the Docker container `adcm`:

```
$ sudo docker volume rm <volume_name>
```

where `<volume_name>` is the volume name. If there are several volumes, enter their names separated by spaces.

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

- to display information about all Docker containers:

```
$ sudo docker ps -a
```

- to display information about all volumes:

```
$ sudo docker volume ls
```

Make sure that the Docker container `adcm` and the volumes associated with this Docker container no longer appear in the lists.

Configure logging

Adelya Zvyagintseva

Contents

To Table of Contents

Log level

uWSGI logging settings

Log rotation

During the operation of ADCM, logs of different types are created in the files located in the */<data_volume>/log/* directory. Its files and subdirectories contain the following information:

- *adcm.log* and *adcm_debug.log* — records of events and errors arising during the ADCM operation, respectively;
- *nginx/* — the Nginx logs;
- *audit.log* — records of the audit events;
- *cron_task.log* — information about background tasks that are executed by a task scheduler, for example, erase the data related to objects configurations, such as cluster or service;
- *task_runner.err* — error messages in case of unsuccessfully launched actions;
- *ldap.log* — messages about authentication in ADCM via LDAP;
- *status.log* — records of events arising during the Status Server operations;
- *wsgi.log* — records of events arising during the uWSGI operations.

The following configuration options that allow you to manage logs and record only the required events to log files are available for the abovementioned log files:

- Log level configuration.
- Configuration of the uWSGI logging settings.
- Log rotation configuration.

Log level configuration can be done for a specific log type. The log level value specified for one type does not affect the other types and is applied during the ADCM launch.

Log level

To configure the log level, you can use special environment variables when creating the container according to the log type:

- `STATUS_LOG_LEVEL` — log level for the Status Server;
- `ADCM_LOG_LEVEL` — log level for the Backend Server;
- `AUDIT_LOG_LEVEL` — log level for *audit.log*;
- `LDAP_LOG_LEVEL` — log level for *ldap.log*;
- `BACKGROUND_TASKS_LOG_LEVEL` — log level for the background jobs;
- `TASK_RUNNER_LOG_LEVEL` — log level for *task_runner.err*.

Here is an example of a command for starting the Docker container along with configuring the log level for the *status.log* file:

```
$ sudo docker run -d --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -e STATUS_LOG_LEVEL="ERROR" 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`.

For more information on special environment variables and possible values, refer to Online installation via a Docker Image.

The `LOG_LEVEL` environment variable is available to all log types in order to provide a backward compatibility.

uWSGI logging settings

Configuration of the required settings in the *adcm.cfg* file located inside the Docker container is the main method of managing the uWSGI logs. The file contains the `[uwsgi]` section and is used to set up the uWSGI configuration.

For more information on formats and methods of loading the configuration file supported by uWSGI, refer to the uWSGI documentation.

The *adcm.cfg* file structure is presented below.

```
[uwsgi]
touch-reload = /etc/adcm/adcm.cfg
module = adcm.wsgi
master = True
processes = 4
harakiri = 6000
pidfile = /run/uwsgi.pid
socket = /run/adcm.sock
chmod-socket = 777
max-requests = 5000
log-4xx = true
log-5xx = true
disable-logging = true
logto = /adcm/data/log/wsgi.log
log-maxsize = 2000000
```

The adcm.cfg file parameters

Parameter	Description
touch-reload	Allows you to automatically restart uWSGI when you change the specified file
module	Allows you to specify a name of the file which is responsible for running Backend Server on the uWSGI web server
master	Allows you to run uWSGI in the `master` mode
processes	Determines a maximum allowable number of processes, i.e. the Backend Server instances
harakiri	Determines a maximum amount of time (in seconds) allotted to a process for performing a task. After the allotted time, the process is aborted and started again
pidfile	Allows you to specify the path to a file to store the process identifier (PID). This parameter is useful in process management
socket	Allows you to specify the path of the socket file for interaction between uWSGI and Nginx
chmod-socket	Allows you to specify access permissions for the socket file. In *adcm.cfg*, the socket file is available to be read, written, and executed for all categories of users
max-requests	Allows you to restart processes after handling the specified number of requests. You can prevent memory leaks by setting this parameter
log-4xx	Allows you to log responses with HTTP status codes of 4xx
log-5xx	Allows you to log responses with HTTP status codes of 5xx
disable-logging	Allows you to disable uWSGI's standard logging
logto	Allows you to specify the log file path
log-maxsize	Determines a maximum log size for log rotation

To specify the uWSGI custom logging settings:

1. Create a new directory for the *adcm.cfg* file.
   For example:

```
$ sudo mkdir /opt/myconf
```

2. Create a new configuration file with the necessary logging settings via the `vi` or `vim` command:

```
$ sudo vi adcm.cfg
```

3. Specify the read, write, and execute permissions for the owner of the file:

```
$ sudo chmod u=rwx /opt/myconf/adcm.cfg
```

4. Start the new `adcm` container with the path of the created file:

```
$ sudo docker run -d --name adcm -p 8000:8000 -v /opt/adcm:/adcm/data -v /<file_path>/adcm.cfg/:/etc/adcm/adcm.cfg hub.arenadata.io/adcm/adcm:<version>
```

   where `<file_path>` is the absolute path to the *adcm.cfg* configuration file.

Log rotation

By default, ADCM has no automatic log rotation. Therefore, `logrotate` is used to solve this problem. This utility archives log files when the specified maximum log size is exceeded.

For more information, please refer to the article Use logrotate to configure log rotation.

The `logrotate` utility applies to all abovementioned log files except the *wsgi.log* since the log file is automatically rotated as soon as the maximum log size specified in *adcm.cfg* is reached.

NOTE

You can use `log-backupname` in the *adcm.cfg* file to specify the required log file name after its rotation, for example, `log-backupname = /opt/adcm/log/uwsgi.log.1`.

Use logrotate to configure log rotation

Adelya Zvyagintseva

Contents

To Table of Contents

Main configuration parameters
Install and configure logrotate
Create a configuration file
Check the configuration file

The `logrotate` utility is designed to manage log files to prevent increasing their size. For example, you can compress, delete, and send log files to another server when specified conditions are met. For more information about `logrotate`, see the GitHub logrotate documentation.

The frequency of log rotation can take the following values:

- `hourly` — log files are rotated every hour;
- `daily` — log files are rotated every day;
- `weekly` — log files are rotated every week;
- `monthly` — log files are rotated every month;
- `yearly` — log files are rotated every year.

By default, log files are rotated weekly.

The `logrotate` configuration is stored in the following paths:

- */etc/logrotate.conf* — the main configuration file that contains default parameters;
- */etc/logrotate.d* — directory that includes additional configuration files.

Main configuration parameters

The table below contains main parameters that can be set in the configuration files.

Configuration parameters

Parameter	Description
size <number>	Determines the size of a log file above which it is rotated. You can specify the log file size in KB (`k`), MB (`M`), or GB (`G`)
create <mode> <owner> <group>	Allows you to create a new log file with the specified permissions, user, and group after rotation
missingok	Allows you to go on to the next step in the absence of a specified log file without issuing an error message
nomail	Allows not sending a rotated log file via email
notifempty	Allows not rotating an empty log file
rotate <count>	Determines the number of rotations, i.e. the number of rotated log files. For example, when the number of rotations equals `0`, rotated log files are removed
sharedscripts	Leads to one-time execution of scripts that added in a configuration file, no matter how many log files match a wildcard pattern, for example, `/var/log/news/*`
postrotate <command> endscript	Leads to execution of commands written between these parameters using the Bourne shell after log rotation. It can be useful for such tasks as restarting a service to ensure a new log file is used for event data recording

Install and configure logrotate

Depending on the operating system and package manager you use, follow the steps below to install and configure `logrotate`:

1. Install `logrotate`:

   YUM

   ```
   $ sudo yum install logrotate
   ```

   APT

   ```
   $ sudo apt install logrotate
   ```

2. Add the following lines to the */etc/logrotate.conf* file:

   - for the inclusion of additional parameters from the */etc/logrotate.d* directory to the configuration file:

   ```
   include /etc/logrotate.d
   ```

   - for adding a user with `root` privileges:

   ```
   # use the root group by default, since this is the owning group
   # of /var/log/syslog.
   su root root
   ```

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

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

   The content of the *logrotate.conf* file should look like this:

   ```
   # see "man logrotate" for details
   # rotate log files weekly
   weekly

   # use the root group by default, since this is the owning group
   # of /var/log/syslog.
   su root root

   # keep 4 weeks worth of backlogs
   rotate 4

   # create new (empty) log files after rotating old ones
   create

   # use date as a suffix of the rotated file
   dateext

   # uncomment this if you want your log files compressed
   #compress

   # RPM packages drop log rotation information into this directory
   include /etc/logrotate.d

   # no packages own wtmp and btmp -- we'll rotate them here
   /var/log/wtmp {
       monthly
       create 0664 root utmp
           minsize 1M
       rotate 1
   }

   /var/log/btmp {
       missingok
       monthly
       create 0600 root utmp
       rotate 1
   }

   # system-specific logs may be also be configured here.
   ```

Create a configuration file

As an example, consider the process of creating the `logrotate` configuration files:

- for Nginx;
- for Status Server;
- for Backend Server (below is an example of the *audit.log* rotation).

To add a configuration file in the */etc/logrotate.d* directory:

1. Create the configuration file via the `vi` or `vim` command:

   - for Nginx:

   ```
   $ sudo vi /etc/logrotate.d/nginx
   ```

   - for Status Server:

   ```
   $ sudo vi /etc/logrotate.d/status-server
   ```

   - for Backend Server:

   ```
   $ sudo vi /etc/logrotate.d/audit
   ```

2. Add a log rotation configuration to the file.

   Here is an example of the *nginx* configuration file:

   ```
   /opt/adcm/log/nginx/*.log {
       su root root
       size 200M
       create 0644
       missingok
       nomail
       notifempty
       rotate 5
       sharedscripts
       postrotate
           docker exec -it adcm sh -c '/usr/sbin/nginx -s reopen'
       endscript
   }
   ```

   where:

   - `/opt/adcm/log/nginx/*.log` is the location of log files with the *.log* extension that should be rotated.
   - `size 200M` is a log file size. When the size of the log file exceeds the configured value, the log file is rotated.
   - `create 0644` is the parameter for creating a new log file with `0644` permission after rotation, meaning that the file's owner can read and write, and all others (group and users) can only read the file.
   - `rotate 5` is a number of rotated log files that have to be saved in `logrotate`. Each log file contains events for a period of the rotation frequency.
   - `docker exec -it adcm sh -c '/usr/sbin/nginx -s reopen'` is the command for reopening the Nginx executable files in the ADCM container and recording the Nginx events to a new log file.

   Here is an example of the *status-server* configuration file:

   ```
   /opt/adcm/log/status.log {
       su root root
       size 200M
       create 0644
       missingok
       rotate 5
       sharedscripts
       postrotate
           docker exec -it adcm sh -c 'killall -USR1 runstatus'
       endscript
   }
   ```

   where `docker exec -it adcm sh -c 'killall -USR1 runstatus'` is the command for sending the `USR1` signal to running processes to close a rotated log file and continue recording events to a new log file.

   Here is an example of the *audit* configuration file:

   ```
   /opt/adcm/log/audit.log {
   size 1k
   create 0644
   missingok
   nomail
   rotate 7
   sharedscripts
   }
   ```

   NOTE

   In a similar way, you can create configuration files to rotate the following log files: *adcm.log*, *adcm_debug.log*, *ldap.log*, and *cron_task.log*.

Check the configuration file

After you add a file to the */etc/logrotate.d* directory, make sure logrotate is working properly by running `logrotate` in the verbose mode that allows you to get detailed information about what `logrotate` is doing:

```
$ sudo logrotate -v /etc/logrotate.d/nginx
```

Example of the command output

```
reading config file /etc/logrotate.d/nginx
Allocating hash table for state file, size 15360 B

Handling 1 logs

rotating pattern: /opt/adcm/log/nginx/*.log  10 bytes (5 rotations)
empty log files are not rotated, old logs are removed
considering log /opt/adcm/log/nginx/access.log
  log does not need rotating (log size is below the 'size' threshold)
considering log /opt/adcm/log/nginx/error.log
  log needs rotating
rotating log /opt/adcm/log/nginx/error.log, log->rotateCount is 5
dateext suffix '-20241002'
glob pattern '-[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
renaming /opt/adcm/log/nginx/error.log.5 to /opt/adcm/log/nginx/error.log.6 (rotatecount 5, logstart 1, i 5),
old log /opt/adcm/log/nginx/error.log.5 does not exist
renaming /opt/adcm/log/nginx/error.log.4 to /opt/adcm/log/nginx/error.log.5 (rotatecount 5, logstart 1, i 4),
old log /opt/adcm/log/nginx/error.log.4 does not exist
renaming /opt/adcm/log/nginx/error.log.3 to /opt/adcm/log/nginx/error.log.4 (rotatecount 5, logstart 1, i 3),
old log /opt/adcm/log/nginx/error.log.3 does not exist
renaming /opt/adcm/log/nginx/error.log.2 to /opt/adcm/log/nginx/error.log.3 (rotatecount 5, logstart 1, i 2),
old log /opt/adcm/log/nginx/error.log.2 does not exist
renaming /opt/adcm/log/nginx/error.log.1 to /opt/adcm/log/nginx/error.log.2 (rotatecount 5, logstart 1, i 1),
old log /opt/adcm/log/nginx/error.log.1 does not exist
renaming /opt/adcm/log/nginx/error.log.0 to /opt/adcm/log/nginx/error.log.1 (rotatecount 5, logstart 1, i 0),
old log /opt/adcm/log/nginx/error.log.0 does not exist
log /opt/adcm/log/nginx/error.log.6 doesn't exist -- won't try to dispose of it
renaming /opt/adcm/log/nginx/error.log to /opt/adcm/log/nginx/error.log.1
creating new /opt/adcm/log/nginx/error.log mode = 0644 uid = 100 gid = 0
running postrotate script
```

Audit operations

Mikhail Serov

Contents

To Table of Contents

Operations

Logins

ADCM allows you to audit all operations (either the ones invoked by the user or the background ones). Audit is presented in the ADCM interface on the ***Audit*** page that contains two tabs:

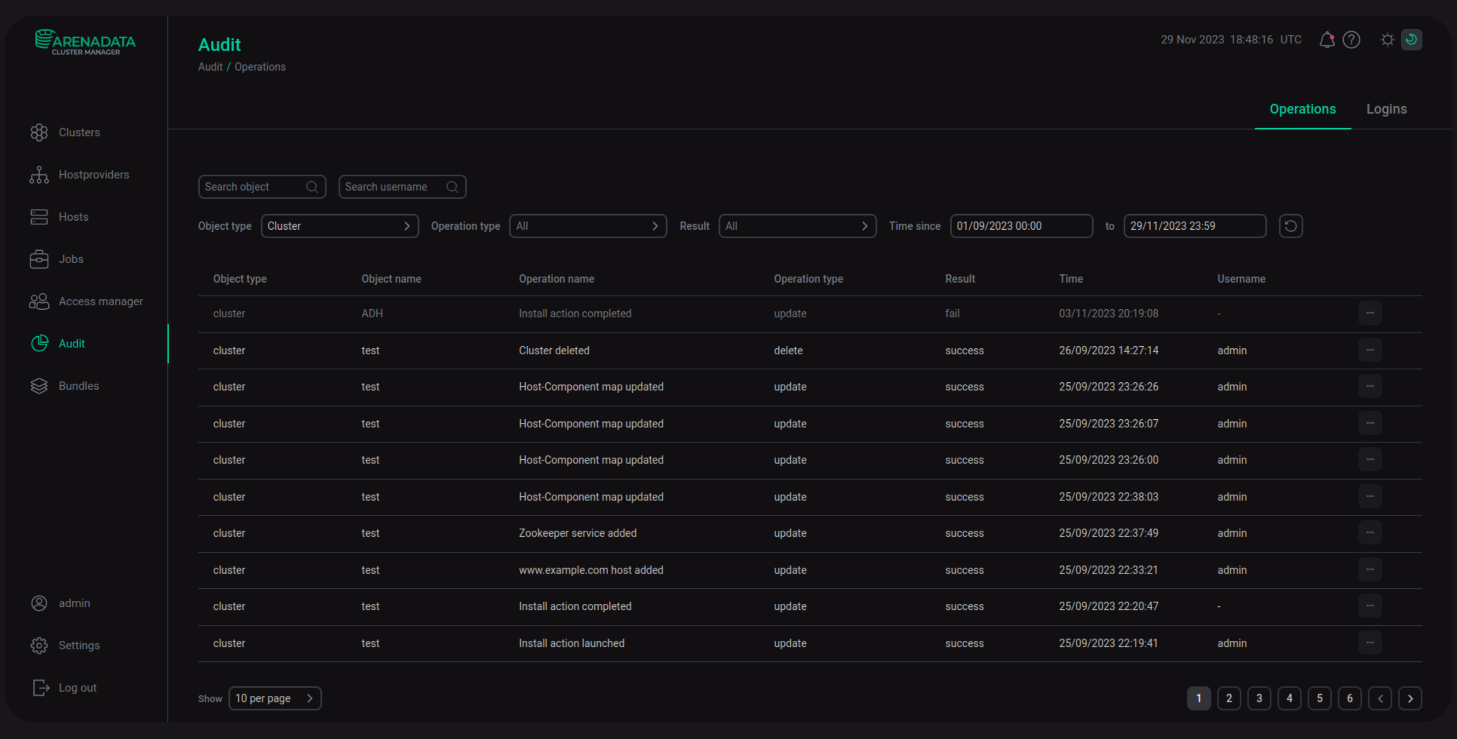
- *Operations*
- *Logins*

Operations

NOTE

The ***Audit*** → ***Operations*** page can display a `system` user, which is used to start scheduled tasks.

All ADCM operations are listed on this tab. Operations can be performed both via UI and API.



Object type: Cluster; Operation type: All; Result: All; Time since 01/09/2023 00:00 to 29/11/2023 23:59

Object type	Object name	Operation name	Operation type	Result	Time	Username
cluster	ADH	Install action completed	update	fail	03/11/2023 20:19:08	-
cluster	test	Cluster deleted	delete	success	26/09/2023 14:27:14	admin
cluster	test	Host-Component map updated	update	success	25/09/2023 23:26:26	admin
cluster	test	Host-Component map updated	update	success	25/09/2023 23:26:07	admin
cluster	test	Host-Component map updated	update	success	25/09/2023 23:26:00	admin
cluster	test	Host-Component map updated	update	success	25/09/2023 22:38:03	admin
cluster	test	Zookeeper service added	update	success	25/09/2023 22:37:49	admin
cluster	test	www.example.com host added	update	success	25/09/2023 22:33:21	admin
cluster	test	Install action completed	update	success	25/09/2023 22:20:47	-
cluster	test	Install action launched	update	success	25/09/2023 22:19:41	admin

The Operations tab

Operations are represented as rows in the audit operations table. For any operation, the audit record consists of the fields that coincide with the columns of the audit operations table. The table of audit operations is described in detail below.

Table of audit operations

Column name	Description
Object type	Type of the object the operation was performed on. Blank in case of a background operation (invoked by the `system` user)
Object name	Name of the object the operation was performed on. Blank in case of a background operation (invoked by the `system` user)
Operation name	Name of the operation
Operation type	Type of the operation. Can take the following values: `create`, `update`, `delete`
Result	Result of the operation. Can take the following values: `success`, `fail`, `denied`
Time	Time the operation was performed at
Username	Name of the user who invoked the operation. Blank if the user has already been deleted

To view the object attribute changes regarding the audited operation, click the ⋯ button located at the end of each row of the audit operation table. This is the way to obtain additional information about the audited operation. Information on object attribute changes is generated for the following operations:

- Any operation on policies, roles, users, and groups.
- Renaming operations on clusters and hosts.
- Switching maintenance mode on and off for hosts, components, and services.

You can select various filters to sort the operation records. Filters are located above the audit operations table. You can also create composite filters (for example, you can select the ***Object type*** and ***Operation type*** filters simultaneously).

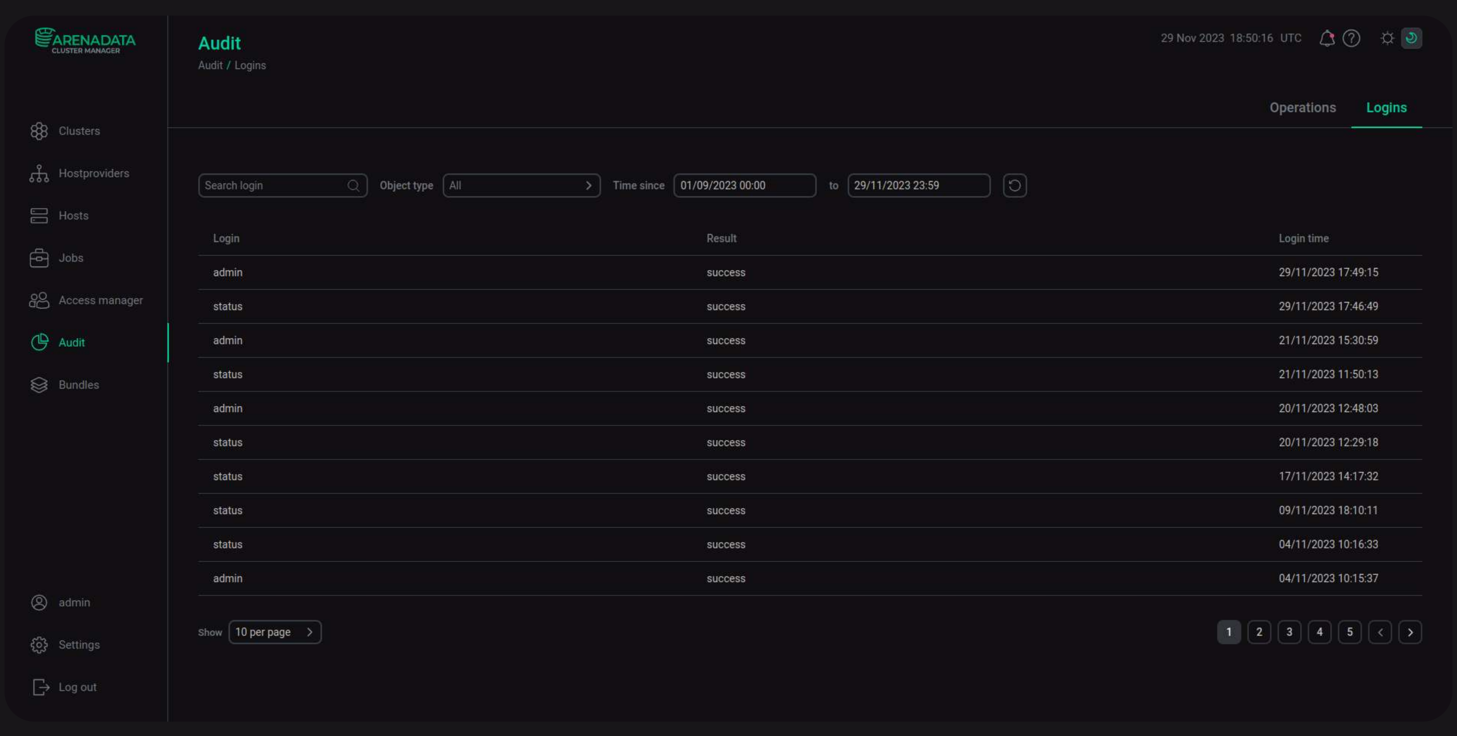
You also can view the audit results in a continuously generated log file.

Logins

NOTE

The ***Audit*** → ***Logins*** page can display a system user `status`, which is used to gather and send states of hosts and service components.

The audit of login operations is presented separately from the audit of other operations.



Object type: All; Time since 01/09/2023 00:00 to 29/11/2023 23:59

Login	Result	Login time
admin	success	29/11/2023 17:49:15
status	success	29/11/2023 17:46:49
admin	success	21/11/2023 15:30:59
status	success	21/11/2023 11:50:13
admin	success	20/11/2023 12:48:03
status	success	20/11/2023 12:29:18
status	success	17/11/2023 14:17:32
status	success	09/11/2023 18:10:11
status	success	04/11/2023 10:16:33
admin	success	04/11/2023 10:15:37

The Logins tab

All login operations that occurred during the current session are represented as rows in the audit logins table. For any login operation, the audit record consists of the fields that coincide with the columns of the audit logins table. The table of audit logins is described in detail below.

Table of audit logins

Column name	Description
Login	User who invoked the login operation
Result	Result of the login operation. Can take the following values: `success`, `fail`, `denied`
Login time	Time the login operation was performed at

View audit logs

Mikhail Serov

Contents

To Table of Contents

Example

Audit results are stored in a log file that is located at *<container mount volume>/log/audit.log* (for example, */opt/adcm/log/audit.log*).

The *audit.log* file is continuously generated every day until midnight. At midnight, the postfix is added to the log file name. At the same moment, a new empty *audit.log* file is created to accumulate logs for the coming day.

Inside the log file the data is stored in the CEF format. The CEF format includes the following fields: `Version|Device Vendor|Device Product|Device Version|Signature ID|Name|Severity|Extension`. The fields are described in the table below.

CEF format fields

Field name	Description
Version	CEF format version
Device Vendor, Device Product, Device Version	Fields that uniquely identify the source of events. There are no products that have the same set of these three values
Signature ID	Event type unique identifier
Name	Human-readable event description
Severity	Event severity (from `0` to `10`)
Extension	Set of key/value pairs. Each product itself defines a dictionary of keys

The `Extension` field is, in fact, a set of fields (key/value pairs) that are designed specifically for ADCM. You can find the detailed description of extension fields in the table below.

Description of extension keys

Extension key	Description
actor	User under which the action was performed (operation or authorization)
act	Type of the ADCM operation. Can take the following values: `create`, `update`, `delete`. This key is used only for operations
operation	ADCM operation name. In case of authorization, the operation name will always be `User logged`
resource	Name of the ADCM object on which the operation is performed. This key is not used when writing authentication data or background jobs. This key can be empty if `act` equals `"create"` and `result` equals `'fail'` or `'denied'`
result	Result of authorization operation. Can take the following values: `success`, `fail`, `denied`. The `success` value means the operation has finished successfully (`Severity = 1`). The `fail` value means the operation has finished unsuccessfully (`Severity = 1`). The `denied` value means that someone has attempted to launch an operation while having no permissions to do so (`Severity = 3`)
timestamp	Date and time of the ADCM operation
address	Last IP address in `HTTP_X_FORWARDED_FOR` if available, otherwise `REMOTE_ADDR`, or null if neither is present

Example

You can view the log file using the `tail` command:

```
$ tail -100f  /opt/adcm/log/audit.log
```

The result is:

```
CEF: 0|Arenadata Software|Arenadata Cluster Manager|2022.10.04.17|api/v1/provider/<int:provider_id>/host/|Host
created|1|actor="admin" act="create" operation="Host created" resource="my-host-1" result="success" timestamp="2022-10-14
13:32:03.389458+00:00"
```

Overview of the Bundles tab in the ADCM UI

Adelya Zvyagintseva

Contents

To Table of Contents

List of bundles

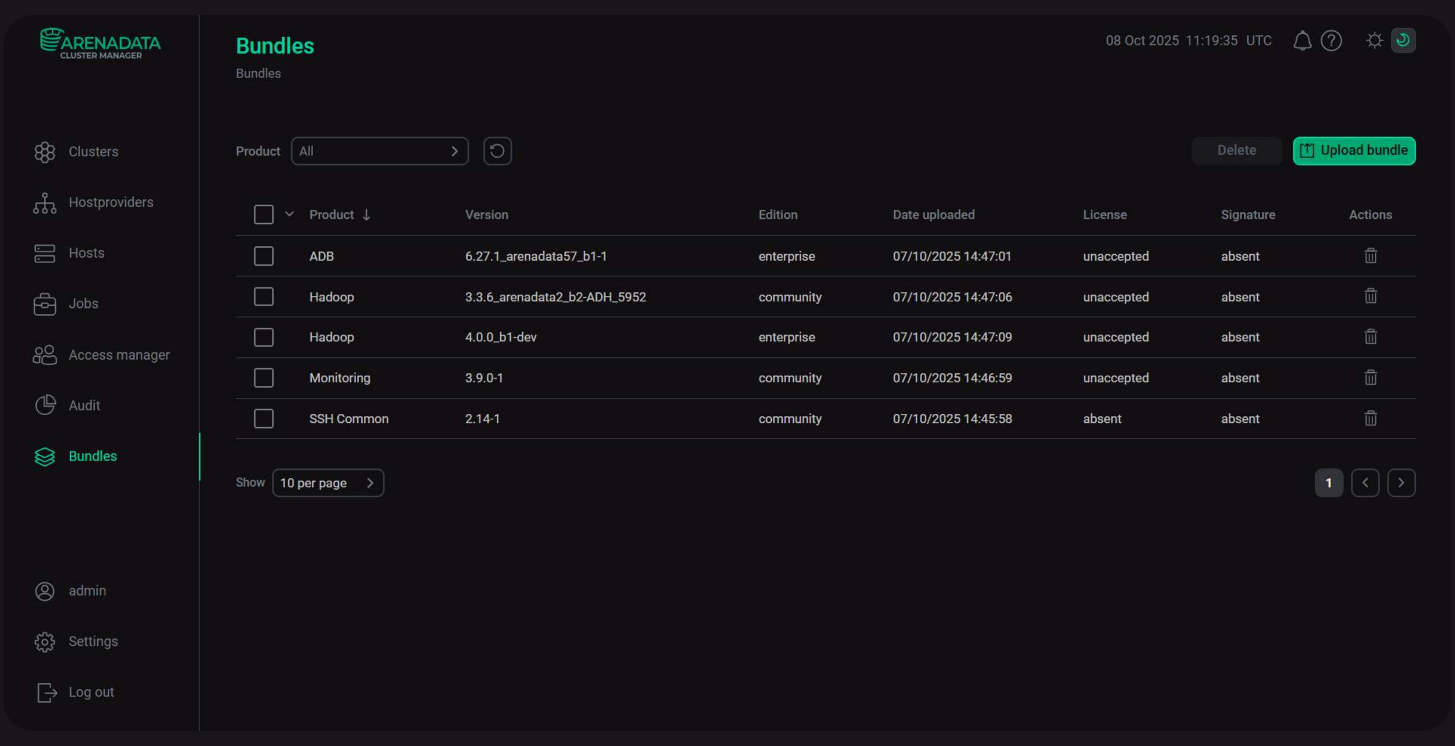
Operations with a bundle

Bundle details

The ***Bundles*** tab in the ADCM web interface allows you to view the uploaded bundles that are used to create clusters and to add hostproviders according to the bundle type.

List of bundles

On the ***Bundles*** tab, you can see the list of uploaded bundles in tabular form which contains information described below.



☐	Product ↓	Version	Edition	Date uploaded	License	Signature	Actions
☐	ADB	6.27.1_arenadata57_b1-1	enterprise	07/10/2025 14:47:01	unaccepted	absent	🗑
☐	Hadoop	3.3.6_arenadata2_b2-ADH_5952	community	07/10/2025 14:47:06	unaccepted	absent	🗑
☐	Hadoop	4.0.0_b1-dev	enterprise	07/10/2025 14:47:09	unaccepted	absent	🗑
☐	Monitoring	3.9.0-1	community	07/10/2025 14:46:59	unaccepted	absent	🗑
☐	SSH Common	2.14-1	community	07/10/2025 14:45:58	absent	absent	🗑

The Bundles tab

Fields of the Bundles tab

Field	Description
Product	Name of the object (cluster or hostprovider depending on the bundle type)
Version	Version of the uploaded bundle
Edition	Edition of the object. Possible values: ▪ `enterprise` ▪ `community`
Date uploaded	Bundle uploading time in the `DD/MM/YYYY HH:mm:ss` format
License	Status of the license agreement. Possible values: ▪ `unaccepted` — the license agreement is not signed; ▪ `accepted` — the license agreement is accepted.
Signature	Status of the bundle signature. Possible values: ▪ `absent` — the signature is missing; ▪ `valid` — the signature has passed the verification via a public key; ▪ `invalid` — the signature has not been verified via the public key.
Actions	Icons for the bundle management. In the current implementation, the 🗑 icon is available to remove the bundle from ADCM

To sort the bundles alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Product*** column.

To filter bundles by the object name, select the required value from the drop-down list in the field next to ***Product***. To reset your filters, click the ↺ icon.

Operations with a bundle

ADCM allows you to manage bundles using operations described below.

To upload a new bundle, follow these steps:

1. Open the ***Bundles*** tab.
2. On the page that opens, click ***Upload bundle***.
3. For further action, refer to the Upload a bundle to ADCM article.

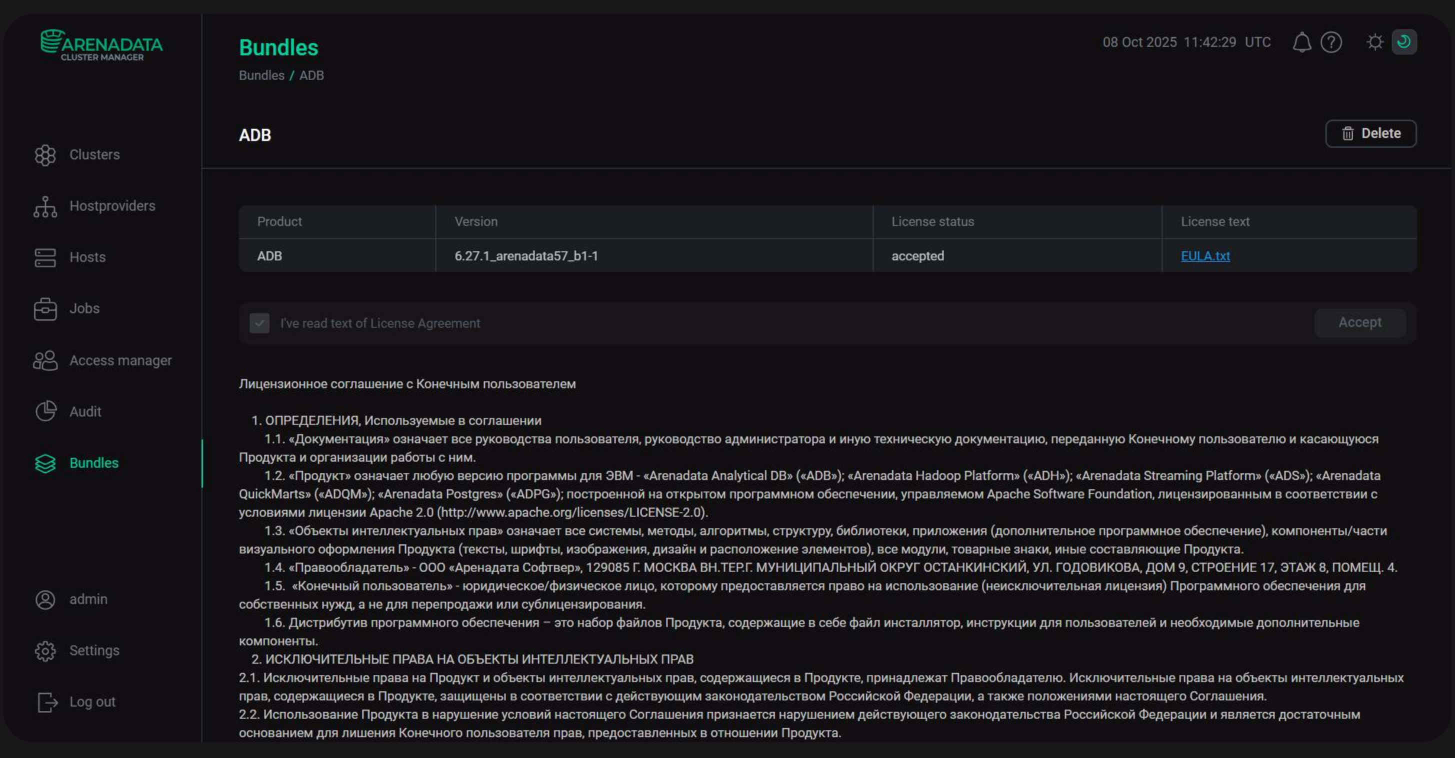
To remove an uploaded bundle from ADCM, follow these steps:

1. Open the ***Bundles*** tab.
2. On the page that opens, select the checkbox next to the desired bundle and click ***Delete***.

> **NOTE**
>
> The delete option is also available on the bundle page. To delete several bundles at once, select checkboxes next to desired bundles and click ***Delete***.

Bundle details

To get detailed information about a bundle, click on the bundle name in the ***Name*** column of the table located on the ***Bundles*** tab.



Product	Version	License status	License text
ADB	6.27.1_arenadata57_b1-1	accepted	EULA.txt

☑ I've read text of License Agreement

Лицензионное соглашение с Конечным пользователем

1. ОПРЕДЕЛЕНИЯ, Используемые в соглашении

1.1. «Документация» означает все руководства пользователя, руководство администратора и иную техническую документацию, переданную Конечному пользователю и касающуюся Продукта и организации работы с ним.

1.2. «Продукт» означает любую версию программы для ЭВМ - «Arenadata Analytical DB» («ADB»); «Arenadata Hadoop Platform» («ADH»); «Arenadata Streaming Platform» («ADS»); «Arenadata QuickMarts» («ADQM»); «Arenadata Postgres» («ADPG»); построенной на открытом программном обеспечении, управляемом Apache Software Foundation, лицензированным в соответствии с условиями лицензии Apache 2.0 (http://www.apache.org/licenses/LICENSE-2.0).

1.3. «Объекты интеллектуальных прав» означает все системы, методы, алгоритмы, структуру, библиотеки, приложения (дополнительное программное обеспечение), компоненты/части визуального оформления Продукта (тексты, шрифты, изображения, дизайн и расположение элементов), все модули, товарные знаки, иные составляющие Продукта.

1.4. «Правообладатель» - ООО «Аренадата Софтвер», 129085 Г. МОСКВА ВН.ТЕР.Г. МУНИЦИПАЛЬНЫЙ ОКРУГ ОСТАНКИНСКИЙ, УЛ. ГОДОВИКОВА, ДОМ 9, СТРОЕНИЕ 17, ЭТАЖ 8, ПОМЕЩ. 4.

1.5. «Конечный пользователь» - юридическое/физическое лицо, которому предоставляется право на использование (неисключительная лицензия) Программного обеспечения для собственных нужд, а не для перепродажи или сублицензирования.

1.6. Дистрибутив программного обеспечения – это набор файлов Продукта, содержащие в себе файл инсталлятор, инструкции для пользователей и необходимые дополнительные компоненты.

2. ИСКЛЮЧИТЕЛЬНЫЕ ПРАВА НА ОБЪЕКТЫ ИНТЕЛЛЕКТУАЛЬНЫХ ПРАВ

2.1. Исключительные права на Продукт и объекты интеллектуальных прав, содержащиеся в Продукте, принадлежат Правообладателю. Исключительные права на объекты интеллектуальных прав, содержащиеся в Продукте, защищены в соответствии с действующим законодательством Российской Федерации, а также положениями настоящего Соглашения.

2.2. Использование Продукта в нарушение условий настоящего Соглашения признается нарушением действующего законодательства Российской Федерации и является достаточным основанием для лишения Конечного пользователя прав, предоставленных в отношении Продукта.

The bundle page

Bundle details

Field	Description
Product	Name of the object. For the detailed description of this field, see Fields of the Bundles tab
Version	Version of the uploaded bundle
License status	Status of the license agreement. For the detailed description of this field, see Fields of the Bundles tab
License text	File of the license agreement in the TXT format

To accept the license agreement, carefully read the terms of the agreement, select the checkbox next to ***I've read text of License Agreement***, and click ***Accept***.

> **NOTE**
>
> Signing the license agreement is necessary to create a cluster or a hostprovider.

To download the license agreement file, click on its name in the ***License text*** column.

Upload a bundle to ADCM

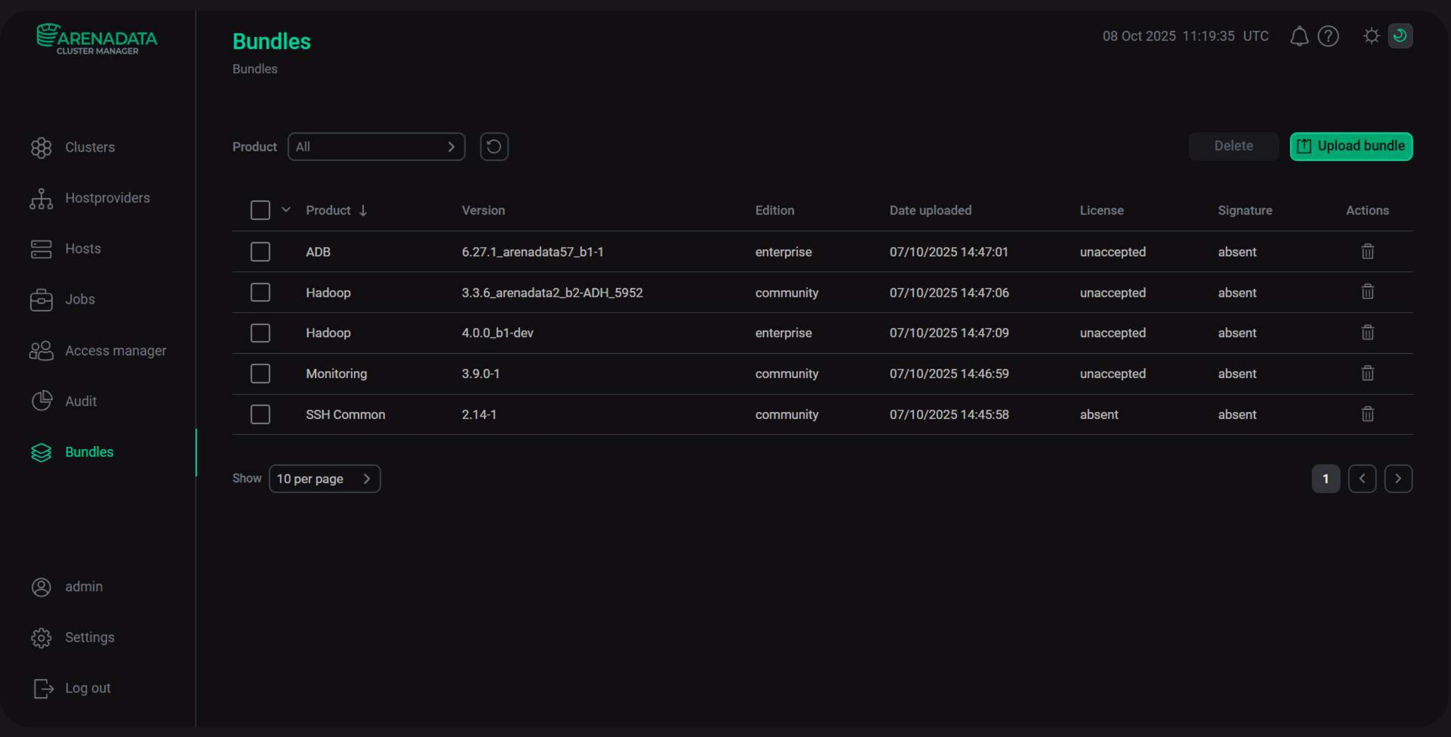
Mikhail Serov

Contents

To Table of Contents

Bundle signature verification

After a bundle is created, it should be uploaded to ADCM. To do this, navigate to the ***Bundles*** page, click ***Upload bundle***, and select the bundle archive. Once the upload is complete, the bundle will appear on the ***Bundles*** page.



Bundles

Bundles

08 Oct 2025 11:19:35 UTC

Product: All

Product	Version	Edition	Date uploaded	License	Signature	Actions
ADB	6.27.1_arenadata57_b1-1	enterprise	07/10/2025 14:47:01	unaccepted	absent	
Hadoop	3.3.6_arenadata2_b2-ADH_5952	community	07/10/2025 14:47:06	unaccepted	absent	
Hadoop	4.0.0_b1-dev	enterprise	07/10/2025 14:47:09	unaccepted	absent	
Monitoring	3.9.0-1	community	07/10/2025 14:46:59	unaccepted	absent	
SSH Common	2.14-1	community	07/10/2025 14:45:58	absent	absent	

Show 10 per page

The Bundles page

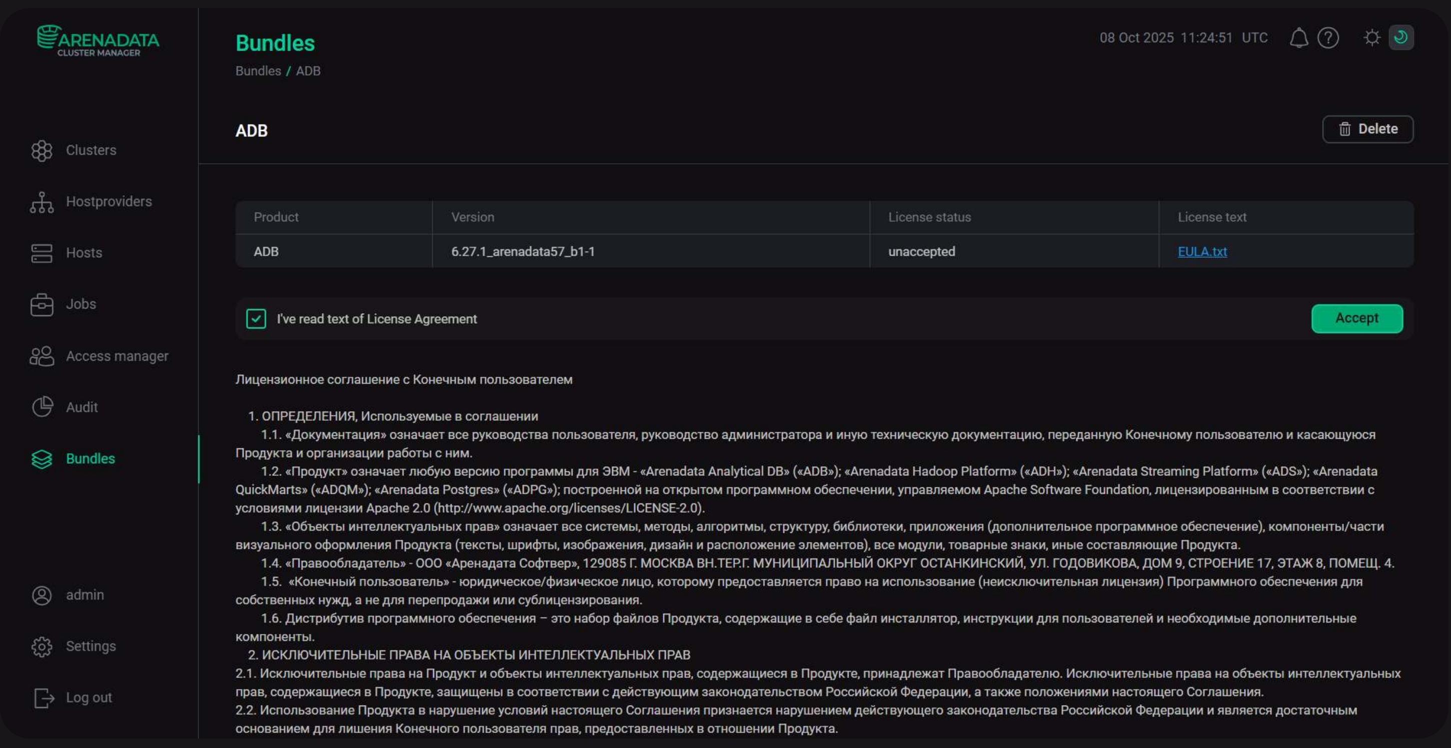
NOTE

You can see the bundles developed for ADCM by Arenadata at https://network.arenadata.io/.

To create a cluster or a hostprovider after the bundle is uploaded to ADCM, you have to accept the license agreement. To accept the license agreement, navigate to the bundle page by clicking the bundle on the ***Bundles*** page. Once there, read the text of the license agreement, select the checkbox, and click ***Accept***.

NOTE

If the user has previously accepted the license agreement and the text of the agreement has not changed, the checkbox will be inactive when downloading a new bundle to update the product.



Bundles

Bundles / ADB

08 Oct 2025 11:24:51 UTC

ADB

Product	Version	License status	License text
ADB	6.27.1_arenadata57_b1-1	unaccepted	EULA.txt

☑ I've read text of License Agreement

Лицензионное соглашение с Конечным пользователем

1. ОПРЕДЕЛЕНИЯ, Используемые в соглашении

1.1. «Документация» означает все руководства пользователя, руководство администратора и иную техническую документацию, переданную Конечному пользователю и касающуюся Продукта и организации работы с ним.

1.2. «Продукт» означает любую версию программы для ЭВМ - «Arenadata Analytical DB» («ADB»); «Arenadata Hadoop Platform» («ADH»); «Arenadata Streaming Platform» («ADS»); «Arenadata QuickMarts» («ADQM»); «Arenadata Postgres» («ADPG»); построенной на открытом программном обеспечении, управляемом Apache Software Foundation, лицензированным в соответствии с условиями лицензии Apache 2.0 (http://www.apache.org/licenses/LICENSE-2.0).

1.3. «Объекты интеллектуальных прав» означает все системы, методы, алгоритмы, структуру, библиотеки, приложения (дополнительное программное обеспечение), компоненты/части визуального оформления Продукта (тексты, шрифты, изображения, дизайн и расположение элементов), все модули, товарные знаки, иные составляющие Продукта.

1.4. «Правообладатель» - ООО «Аренадата Софтвер», 129085 Г. МОСКВА ВН.ТЕР.Г. МУНИЦИПАЛЬНЫЙ ОКРУГ ОСТАНКИНСКИЙ, УЛ. ГОДОВИКОВА, ДОМ 9, СТРОЕНИЕ 17, ЭТАЖ 8, ПОМЕЩ. 4.

1.5. «Конечный пользователь» - юридическое/физическое лицо, которому предоставляется право на использование (неисключительная лицензия) Программного обеспечения для собственных нужд, а не для перепродажи или сублицензирования.

1.6. Дистрибутив программного обеспечения – это набор файлов Продукта, содержащие в себе файл инсталлятор, инструкции для пользователей и необходимые дополнительные компоненты.

2. ИСКЛЮЧИТЕЛЬНЫЕ ПРАВА НА ОБЪЕКТЫ ИНТЕЛЛЕКТУАЛЬНЫХ ПРАВ

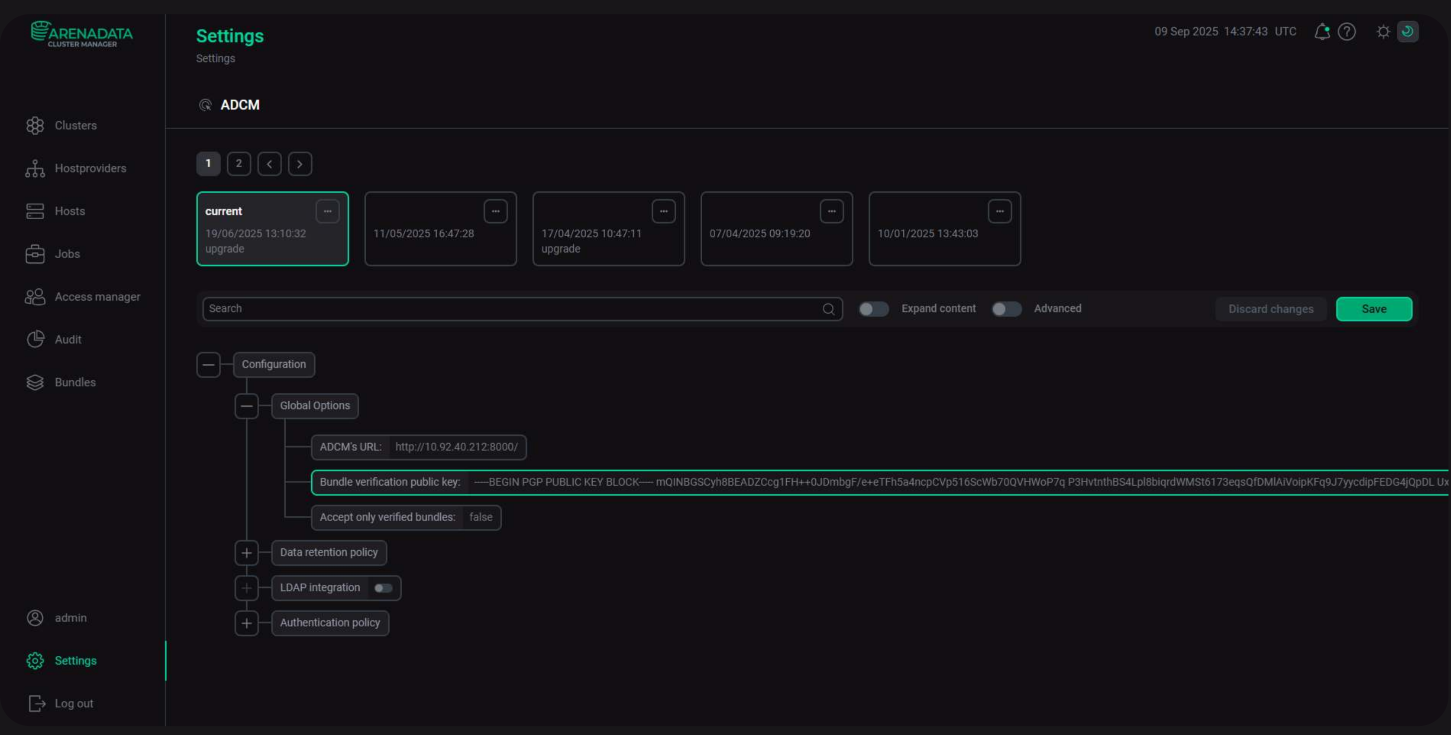
2.1. Исключительные права на Продукт и объекты интеллектуальных прав, содержащиеся в Продукте, принадлежат Правообладателю. Исключительные права на объекты интеллектуальных прав, содержащиеся в Продукте, защищены в соответствии с действующим законодательством Российской Федерации, а также положениями настоящего Соглашения.

2.2. Использование Продукта в нарушение условий настоящего Соглашения признается нарушением действующего законодательства Российской Федерации и является достаточным основанием для лишения Конечного пользователя прав, предоставленных в отношении Продукта.

The Bundle page

Bundle signature verification

Bundle has a signature that can be found in the file with the *.sig* extension which is located inside. When the bundle is uploaded, the signature is verified via the public key specified in the ***Bundle verification public key*** field on the configuration page.



Settings

Settings

09 Sep 2025 14:37:43 UTC

ADCM

current 19/06/2025 13:10:32 upgrade

11/05/2025 16:47:28

17/04/2025 10:47:11 upgrade

07/04/2025 09:19:20

10/01/2025 13:43:03

ADCM's URL: http://10.92.40.212:8000/

Bundle verification public key: -----BEGIN PGP PUBLIC KEY BLOCK----- mQINBGSCyh8BEADZCcg1FH++0JDmbgF/e+eTFh5a4ncpCVp516ScWb70QVHWoP7q P3HvtnthBS4Lpl8biqrdWMSt6173eqsQfDMlAiVoipKFq9J7yycdipFEDG4jQpDL Ux

Accept only verified bundles: false

Bundle verification public key field

Bundle signature status can be one of the following:

- `Valid` — the signature verification is successful.
- `Invalid` — the signature and the specified public key do not match or the public key is missing.
- `Absent` — the signature is missing.

IMPORTANT

Bundles that are issued by Arenadata before the 2023.09.28.17 release do not have signatures, so each of those bundles has the `Absent` status which is normal and does not affect the bundle functioning.

Overview of the Hostproviders tab in the ADCM UI

Adelya Zvyagintseva

Contents

To Table of Contents

List of hostproviders

Operations with a hostprovider

Hostprovider details

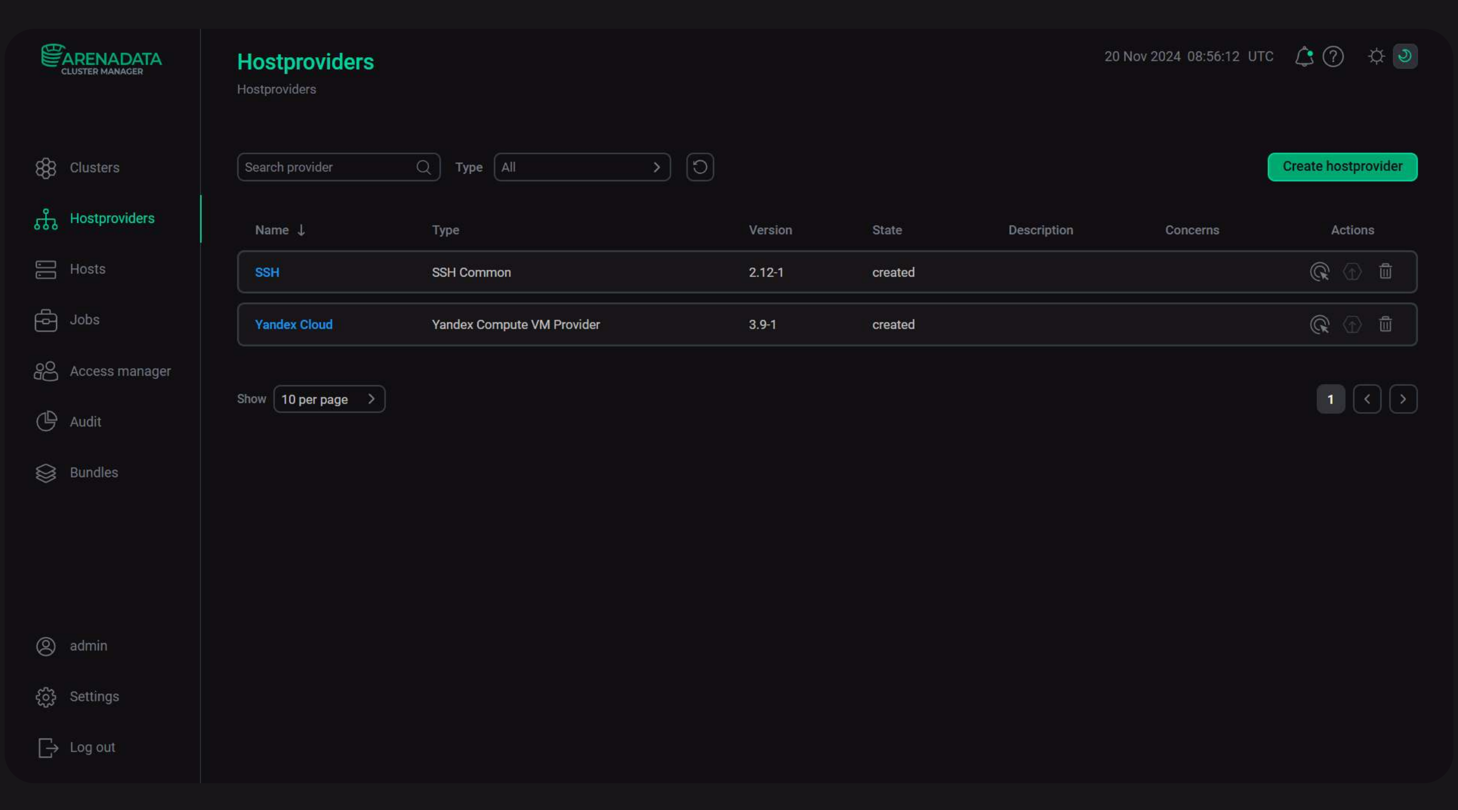
After you upload an infrastructure bundle, you can create and manage hostproviders on the ***Hostproviders*** tab in the ADCM web interface via the hostprovider actions that are described below.

ADCM supports the following types of hostproviders:

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

List of hostproviders

On the ***Hostproviders*** tab, you can see the list of created hostproviders in tabular form which contains information described below.



ARENADATA CLUSTER MANAGER
Hostproviders
Hostproviders
20 Nov 2024 08:56:12 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search provider
Type
All
Create hostprovider
Show
10 per page
1

Name ↓	Type	Version	State	Description	Concerns	Actions
SSH	SSH Common	2.12-1	created			
Yandex Cloud	Yandex Compute VM Provider	3.9-1	created			

The Hostproviders tab

Fields of the Hostproviders tab

Field	Description
Name	Name of the hostprovider specified during the hostprovider creation
Type	Type of the hostprovider. Possible values: - `SSH Common` - `Yandex Compute VM Provider` - `VMware vCloud Director` - `CloudRu_Advanced Provider` - `Croc Provider` - `ADB Cloud bundle` - `ADQM Cloud Bundle`
Version	Version of the bundle used to create the hostprovider
State	Status of the hostprovider. The only possible value is `created`
Description	Description of the hostprovider specified during the hostprovider creation
Concerns	One of the following icons: - ⓘ — indicates the presence of blocking concerns. - ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	Icons for managing the hostprovider: - — opens the drop-down list with actions to manage the hostprovider. - ⇧ — indicates whether a new version of the bundle is available and allows you to upgrade the hostprovider. If no version is available, the icon image is changed to ⇧ . - 🗑 — deletes the information about the hostprovider from ADCM.

NOTE

The delete option is also available on the hostprovider page.

To sort the hostproviders alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

To search for a hostprovider in the list, enter its name in the ***Search provider*** field.

To filter hostproviders by type, select the required value from the drop-down list in the field next to ***Type***. To reset your filters, click the ↻ icon.

Operations with a hostprovider

ADCM allows you to create and configure hostproviders.

To create and configure a new hostprovider, follow these steps:

1. Open the ***Hostproviders*** tab.
2. On the page that opens, click ***Create hostprovider***.
3. For further action, refer to the following links, depending on the hostprovider type:
    - SSH
    - Yandex Cloud
    - VCloud (VMware)
    - Cloud.ru Advanced
    - CROC Cloud
    - ADB Cloud Bundle
    - ADQM Cloud Bundle

After you create and configure the hostprovider, you can manage it on the ***Hostproviders*** tab. Click the icon in the ***Actions*** column to select one of the available hostprovider actions.

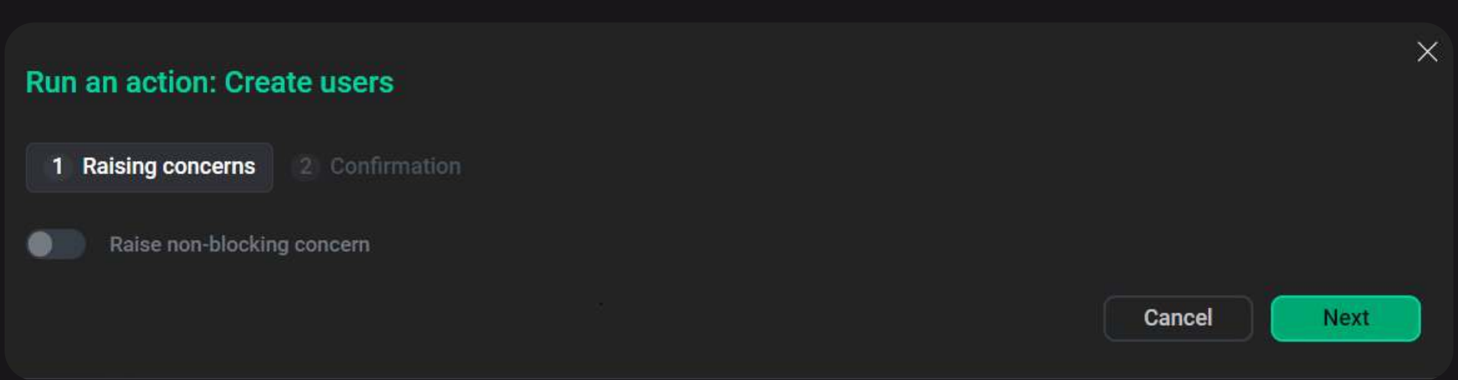
For more information on actions for each hostprovider type, refer to the following links:

- SSH
- Yandex Cloud
- Cloud.ru Advanced
- CROC Cloud

NOTE

For the VCloud (VMware), ADB Cloud Bundle, and ADQM Cloud Bundle hostproviders, actions are not provided in current versions.

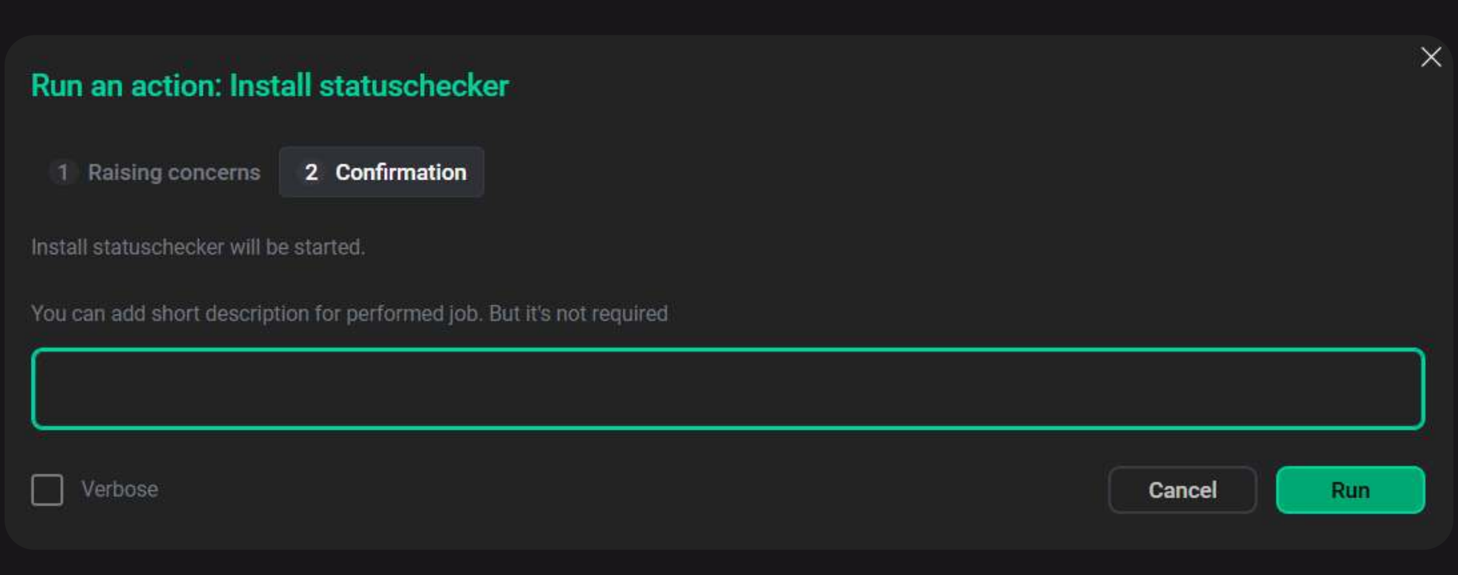
When you choose an action, ADCM displays a standard dialog box.



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

Dialog box to run the Create users action

In this dialog box, you can run the action without blocking the object if necessary, add a comment with a brief description of changes, and select the ***Verbose*** checkbox to see additional execution details on the ***Jobs*** page.



Run an action: Install statuschecker
1 Raising concerns
2 Confirmation
☐ Verbose
Cancel
Run

Install statuschecker will be started.

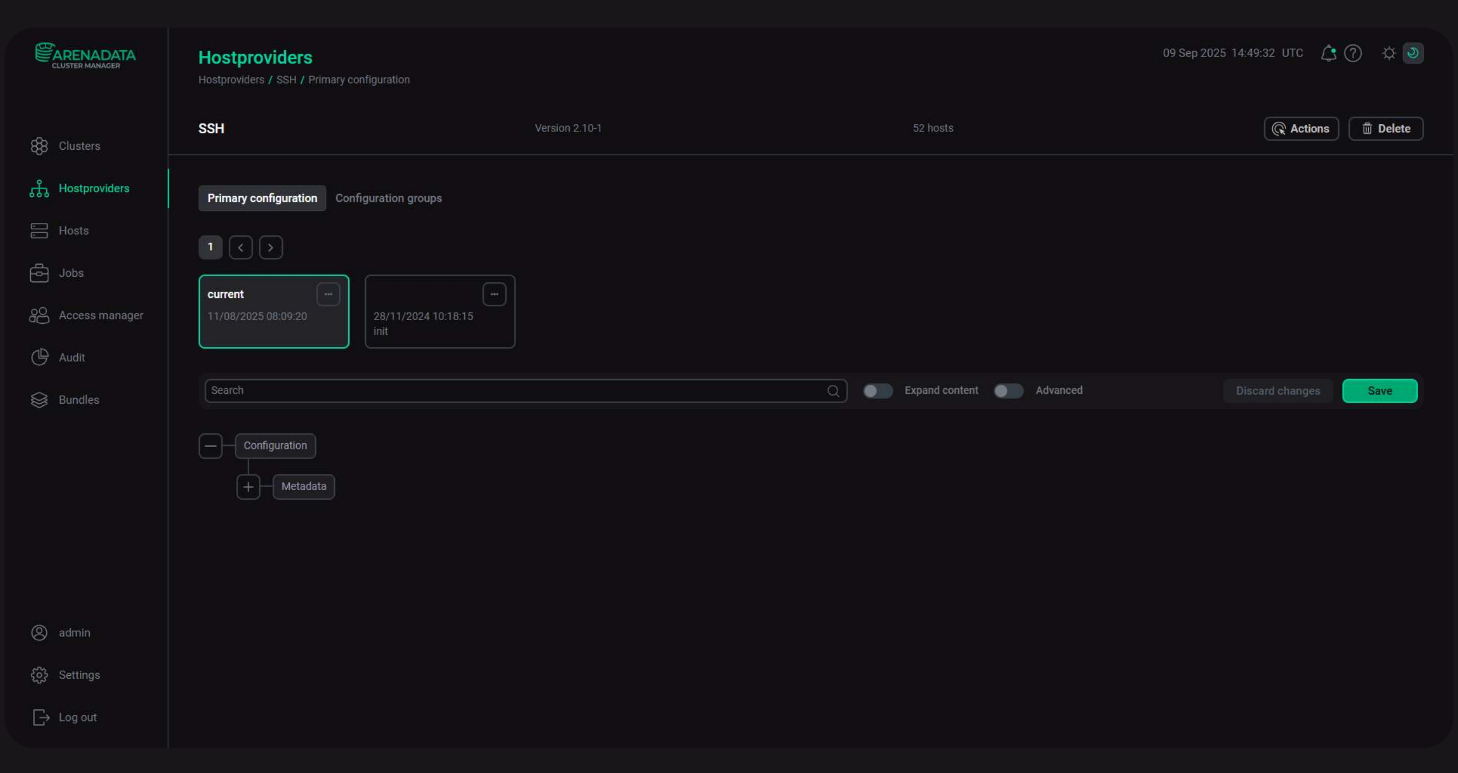
You can add short description for performed job. But it's not required

Dialog box to run the Install statuschecker action containing a comment field

For some actions, such as ***Create hosts***, you should first fill in additional options in a separate tab.

Hostprovider details

To get detailed information about the hostprovider, click on the hostprovider name in the ***Name*** column of the table located on the ***Hostproviders*** tab.



ARENADATA CLUSTER MANAGER
Hostproviders
Hostproviders / SSH / Primary configuration
09 Sep 2025 14:49:32 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
SSH
Version 2.10-1
52 hosts
Actions
Delete
Primary configuration
Configuration groups
1
current
11/08/2025 08:09:20
28/11/2024 10:18:15
init
Search
Expand content
Advanced
Discard changes
Save
Configuration
Metadata

The SSH hostprovider page

At the top of the hostprovider page, the following is displayed:

- bundle version;
- number of hosts created on the basis of the hostprovider;
- buttons ***Actions*** and ***Delete***.

Also, this page includes the following tabs:

- ***Primary configuration*** — allows you to configure the general hostprovider settings. The tab description and information about working with configuration parameters are given in the Configuration parameters in the ADCM UI article.
- ***Configuration groups*** — allows you to set up hostprovider configuration groups. For more information, refer to the article Set up configuration groups.

Overview of the Hosts tab in the ADCM UI

Adelya Zvyagintseva

Contents

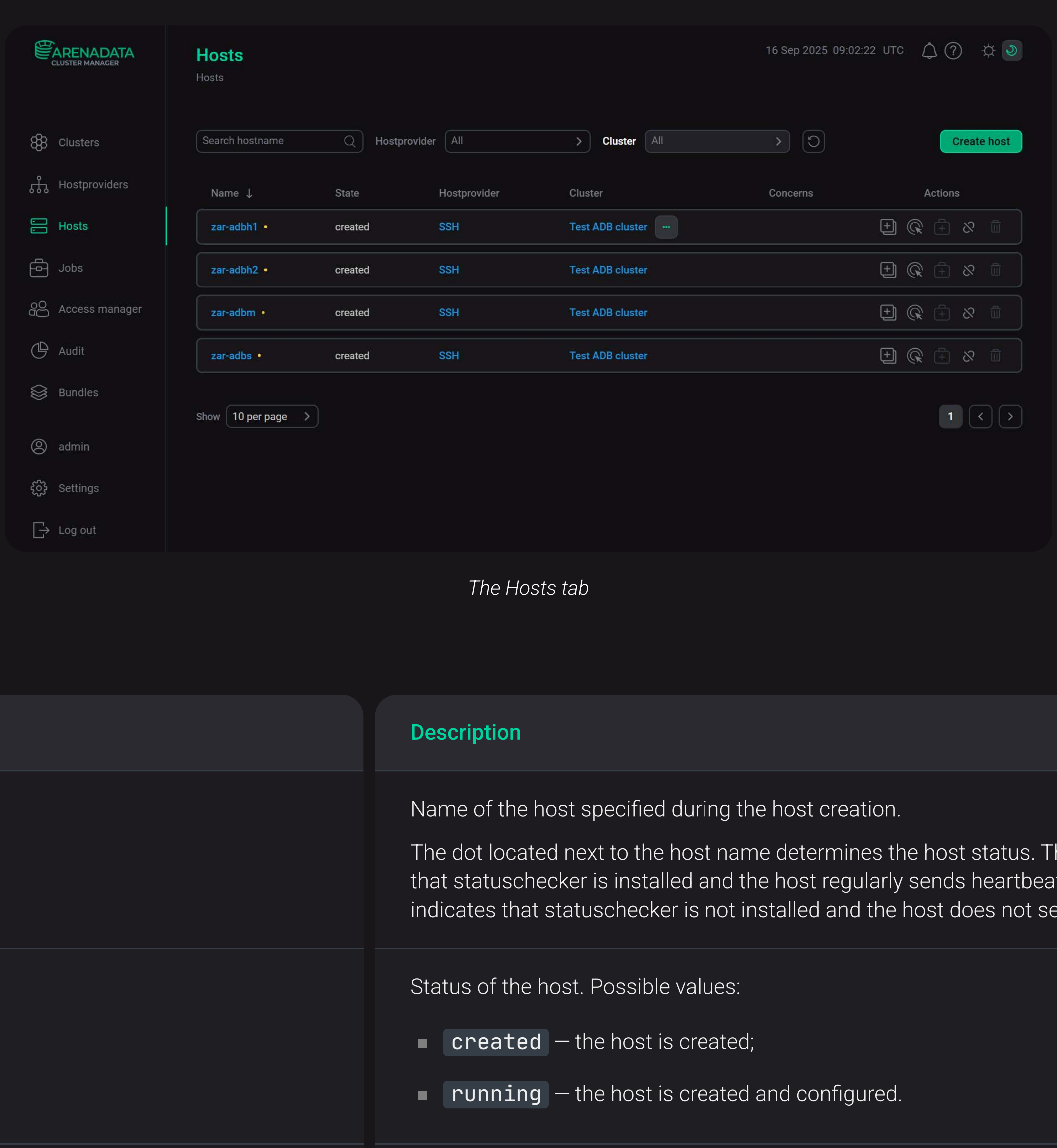
To Table of Contents

- List of hosts
- Operations with a host
  - Create and configure a host
  - Run host actions
  - Enable host sharing
- Host details

After you install and configure a hostprovider, you can create and manage hosts on the ***Hosts*** tab in the ADCM web interface via the host actions described below.

List of hosts

On the ***Hosts*** tab, you can see the list of created hosts in tabular form which contains information described below.



ARENADATA CLUSTER MANAGER
Hosts
Hosts
16 Sep 2025 09:02:22 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Cluster
All
Create host
Show
10 per page
1

Name	State	Hostprovider	Cluster	Concerns	Actions
zar-adbh1	created	SSH	Test ADB cluster		
zar-adbh2	created	SSH	Test ADB cluster		
zar-adbm	created	SSH	Test ADB cluster		
zar-adbs	created	SSH	Test ADB cluster		

The Hosts tab

Fields of the Hosts tab

Field	Description
Name	Name of the host specified during the host creation. The dot located next to the host name determines the host status. The green dot indicates that statuschecker is installed and the host regularly sends heartbeats. The yellow one indicates that statuschecker is not installed and the host does not send heartbeats to ADCM
State	Status of the host. Possible values: - `created` — the host is created; - `running` — the host is created and configured.
Hostprovider	Name of the hostprovider used to create the host
Cluster	Name of the cluster the host belongs to
Concerns	One of the following icons: - ⓘ — indicates the presence of blocking concerns. - ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	Icons for managing the host: - ⊞ — creates a subhost to enable host sharing between multiple clusters. The icon is available starting with ADCM 2.8.0. - ◎ — opens the drop-down list with actions to manage the host. - ⊕ — turns on the maintenance mode on the host if supported by the product in use. If maintenance mode is not supported by the product, the icon image is changed to ⊕ . After the action is executed, the host changes its state and the icon is changed to the ⊕ icon. In this state, the host can be transferred to normal mode. - 🔗 — adds the host to a cluster. After the action is executed, the icon is changed to the ⛓ icon which performs the opposite action. - 🗑 — deletes the information about the host from ADCM. The host removal is possible only after removing the host from a cluster using the ***Unlink*** action.

> **NOTE**
>
> Add/remove and delete options, and the ◎ icon are also available on the host page.

To sort the hosts alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

To search for a host in the list, enter its name in the ***Search hostname*** field.

To filter hosts:

- by hostprovider name — select the required value from the drop-down list in the field next to ***Hostprovider***;
- by cluster name — select the required value from the drop-down list in the field next to ***Cluster***.

To reset your filters, click the ↻ icon.

Operations with a host

Create and configure a host

ADCM allows you to create and configure hosts.

To create and configure a single host, follow these steps:

1. Open the ***Hosts*** tab.
2. On the page that opens, click ***Create host***.
3. For further action, refer to the following links, depending on the hostprovider type:
   - SSH
   - Yandex Cloud
   - VCloud (VMware)
   - Cloud.ru Advanced
   - CROC Cloud
   - ADB Cloud Bundle
   - ADQM Cloud Bundle

> **NOTE**
>
> To add several hosts with the SSH or Yandex Cloud hostproviders simultaneously, run the hostprovider action ***Create hosts***.

Run host actions

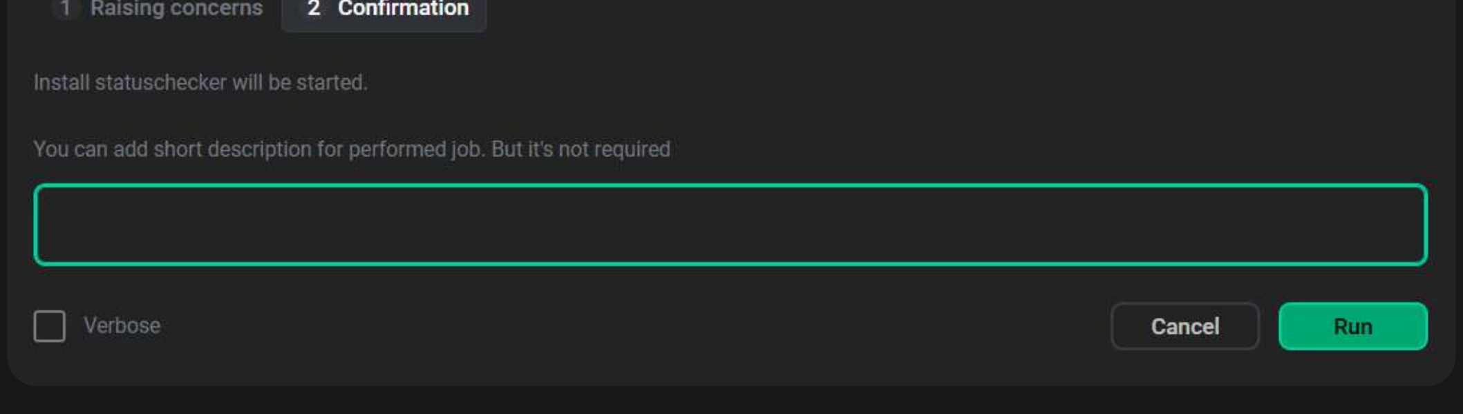
After you create and configure a new host, you can manage it on the ***Hosts*** page. Click the ◎ icon in the ***Actions*** column to select one of the available host actions.

For more information on actions for hosts based on different hostprovider types, refer to the following links:

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

> **NOTE**
>
> For hosts based on the ADB Cloud Bundle and ADQM Cloud Bundle hostproviders, actions are not yet provided.

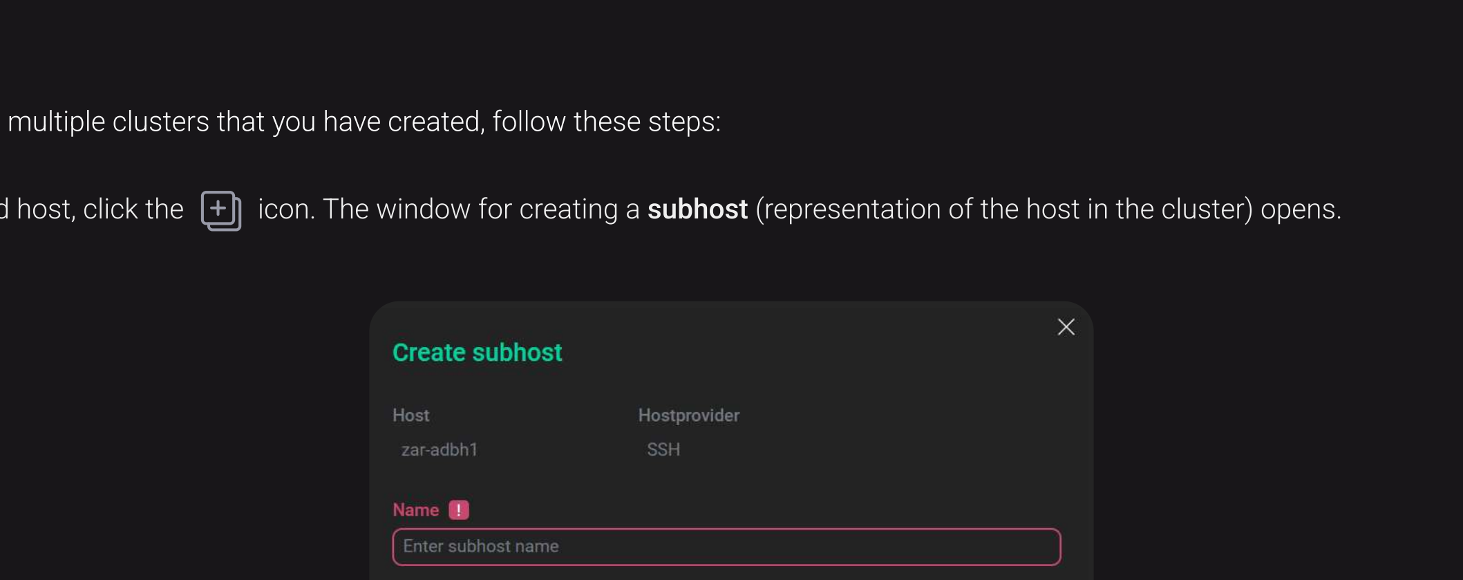
When you choose an action, ADCM displays a standard dialog box.


Run an action: Check connection

1 Raising concerns
2 Confirmation
Raise non-blocking concern
Cancel
Next

Dialog box to run the Check connection action

In this dialog box, you can run the action without blocking the object if necessary, add a comment with a brief description of changes, and select the ***Verbose*** checkbox to see additional execution details on the ***Jobs*** page.


Run an action: Install statuschecker

1 Raising concerns
2 Confirmation
Verbose
Cancel
Run

Install statuschecker will be started.

You can add short description for performed job. But it's not required

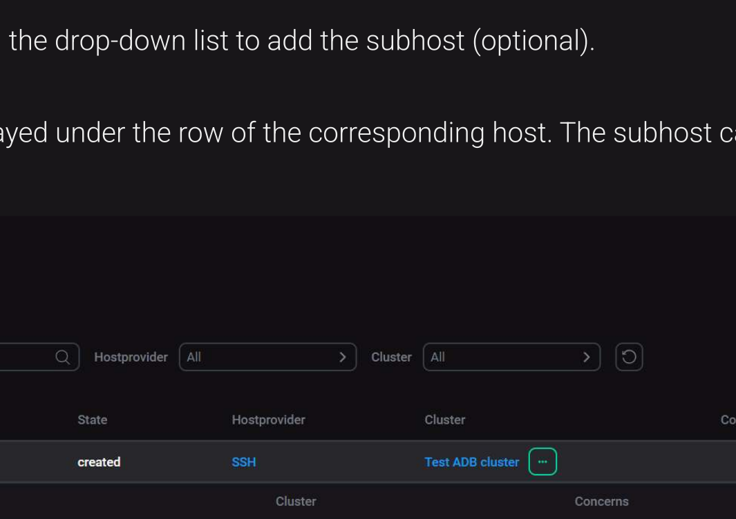
Dialog box to run the Install statuschecker action containing a comment field

For some actions, such as ***Create user***, you should first fill in additional options on a separate tab.

Enable host sharing

To enable host sharing between multiple clusters that you have created, follow these steps:

1. In the row with the required host, click the ⊞ icon. The window for creating a **subhost** (representation of the host in the cluster) opens.


Create subhost

Host: zar-adbh1

Hostprovider: SSH

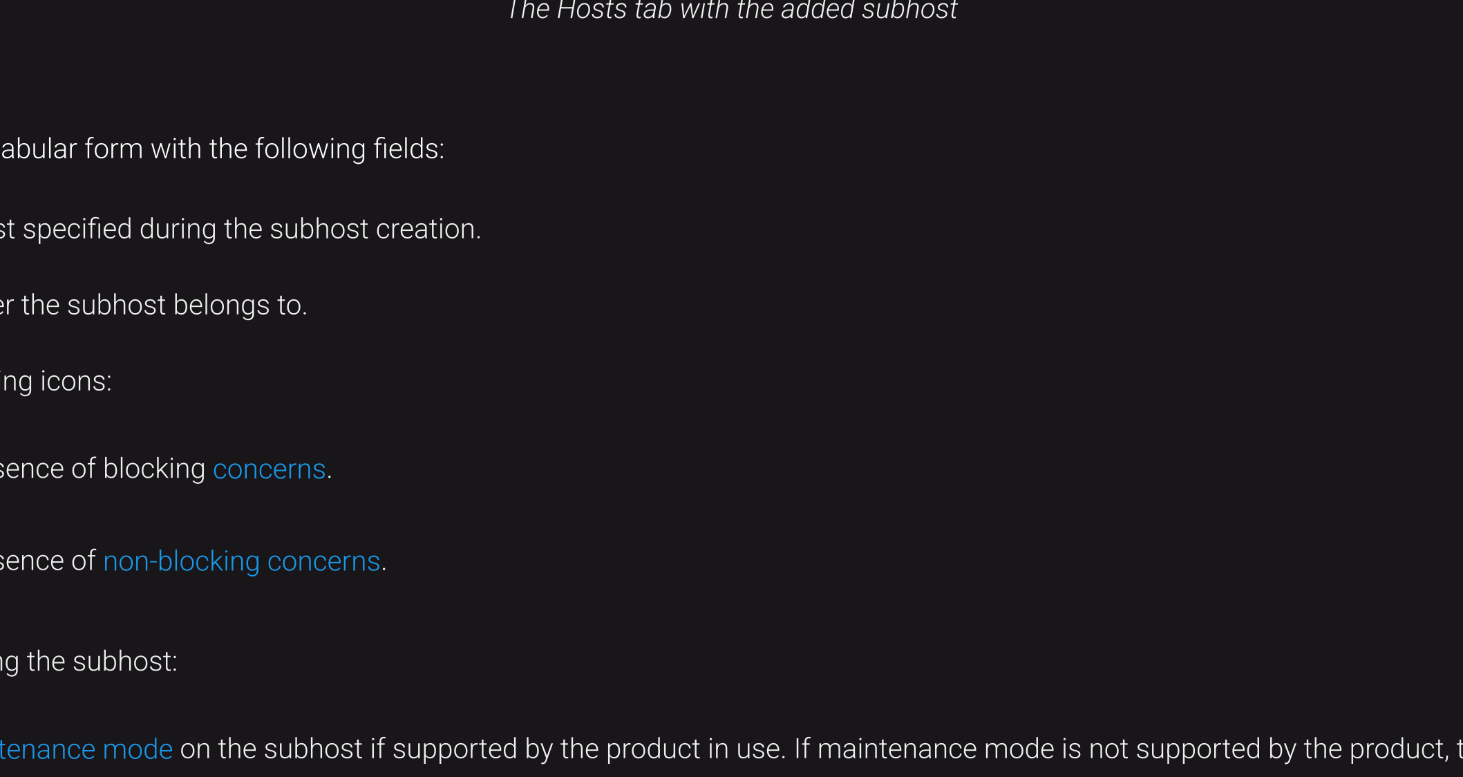
Name: Enter subhost name

Cluster: Select cluster

Cancel
Create

The Create subhost window

2. In the ***Create subhost*** window:
   - in the ***Name*** field, enter the name of the subhost;
   - in the ***Cluster*** field, select the name of a cluster from the drop-down list to add the subhost (optional).
3. Click ***Create***. As a result, the added subhost will be displayed under the row of the corresponding host. The subhost can be used to enable host sharing between multiple clusters.



ARENADATA CLUSTER MANAGER
Hosts
Hosts
16 Sep 2025 09:36:53 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Search hostname
Hostprovider
All
Cluster
All
Create host
Show
10 per page
1

Name	State	Hostprovider	Cluster	Concerns	Actions
zar-adbh1	created	SSH	Test ADB cluster		
Name	Cluster		Concerns		Actions
subhost-1	AD1				
subhost-2	AD2				
zar-adbh2	created	SSH	Test ADB cluster		
zar-adbm	created	SSH	Test ADB cluster		
zar-adbs	created	SSH	Test ADB cluster		

The Hosts tab with the added subhost

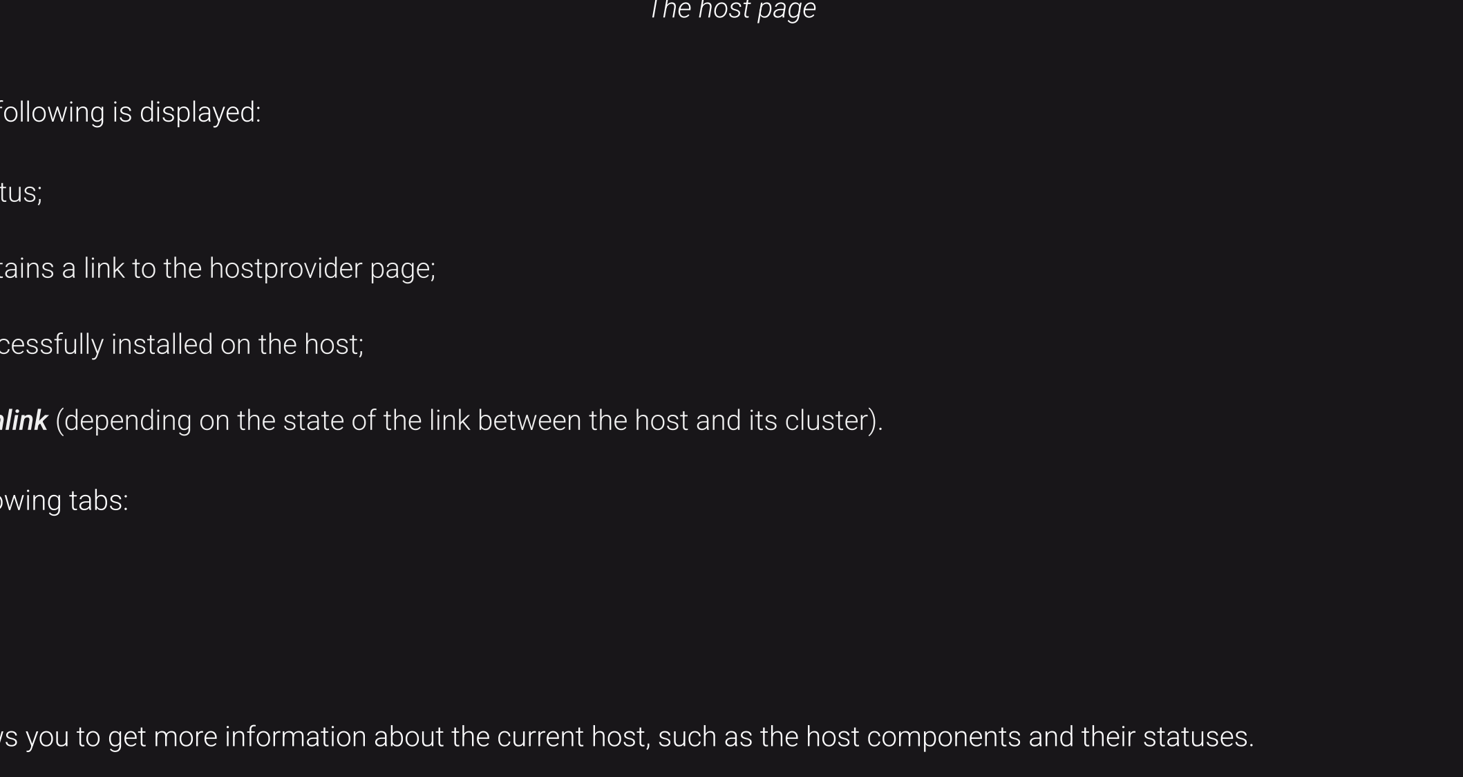
The subhost list is displayed in tabular form with the following fields:

- ***Name*** — name of the subhost specified during the subhost creation.
- ***Cluster*** — name of the cluster the subhost belongs to.
- ***Concern*** — one of the following icons:
  - ⓘ — indicates the presence of blocking concerns.
  - ⓘ — indicates the presence of non-blocking concerns.
- ***Actions*** — icons for managing the subhost:
  - ⊕ — turns on the maintenance mode on the subhost if supported by the product in use. If maintenance mode is not supported by the product, the icon image is changed to ⊕ . After the action is executed, the subhost changes its state and the icon is changed to the ⊕ icon. In this state, the subhost can be transferred to normal mode.
  - 🔗 — adds the subhost to a cluster from the drop-down list. After the action is executed, the icon is changed to the ⛓ icon which performs the opposite action.
  - 🗑 — deletes the information about the subhost from ADCM. The subhost removal is possible only after removing the subhost from a cluster using the ***Unlink*** action.

To expand or collapse the list of subhosts under the row of the corresponding host, click the ⋯ or ⋯ icon respectively.

Host details

To get detailed information about the host, click on the host name in the ***Name*** column of the table located on the ***Hosts*** tab.



ARENADATA CLUSTER MANAGER
Hosts
Hosts / h-3 / Primary configuration
10 Sep 2025 08:11:56 UTC
h-3
Yandex
1 / 1 successful components
Actions
Unlink
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Host-Components
Primary configuration
current
27/01/2025 09:38:35
ansible update
27/01/2025 09:37:52
ansible update
27/01/2025 09:37:10
ansible update
27/01/2025 09:37:09
ansible update
27/01/2025 09:37:09
ansible update
Search
Expand content
Advanced
Discard changes
Save
Configuration
instance

The host page

At the top of the host page, the following is displayed:

- host name with the host status;
- hostprovider name that contains a link to the hostprovider page;
- number of components successfully installed on the host;
- buttons ***Actions*** and ***Link***/***Unlink*** (depending on the state of the link between the host and its cluster).

Also, this page includes the following tabs:

- ***Host-Components***
- ***Primary configuration***

The ***Host-Components*** tab allows you to get more information about the current host, such as the host components and their statuses.

Fields of the Host-Components tab

Field	Description
Name	Name of the component. The dot located next to the component name determines the component status. The green dot indicates that a component operates correctly. The yellow one means that a problem occurs
Concerns	One of the following icons: - ⓘ — indicates the presence of blocking concerns. - ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	The ◎ icon opens the drop-down list with actions to manage the component

To sort the components alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

To search for a component in the list, enter its name in the ***Search component*** field.

The ***Primary configuration*** tab allows you to configure the general host settings. The tab description and information about working with configuration parameters are given in the Configuration parameters in the ADCM UI article.

Overview of the Clusters tab in the ADCM UI

Adelya Zvyagintseva

Contents

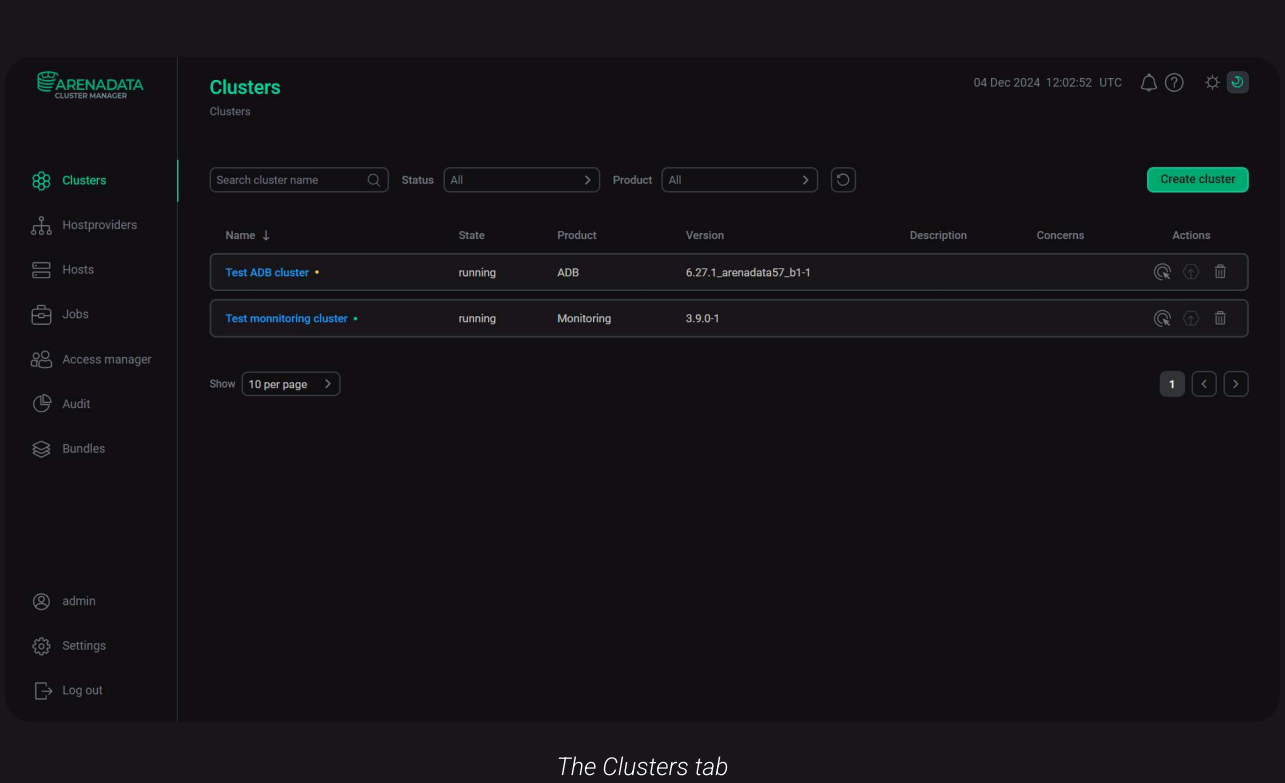
To Table of Contents

- List of clusters
- Change a name and description of a cluster
- Operations with a cluster
- Cluster details
  - Overview
  - Services
  - Hosts
  - Mapping
  - Configuration
  - Import

After you upload a cluster bundle, you can install and manage clusters on the ***Clusters*** tab in the ADCM web interface via the cluster actions that are described below.

List of clusters

On the ***Clusters*** tab, you can see the list of created clusters in tabular form which contains information described below.



Clusters

Name	State	Product	Version	Description	Concerns	Actions
Test ADB cluster	running	ADB	6.27.1_arenadata57_b1-1			
Test monitoring cluster	running	Monitoring	3.9.0-1			

The Clusters tab

Fields of the Clusters tab

Field	Description
Name	Name of the cluster specified during the cluster creation. The dot located next to the cluster name determines the cluster status. The green dot indicates that all cluster components and hosts work. The yellow one indicates that any of the components or hosts is stopped or is not available. If needed, it is possible to rename the cluster if it has the `created` status
State	Status of the cluster. The list of possible values depends on a product in use
Product	Product name
Version	Version of the cluster bundle used to install or upgrade the cluster
Description	Cluster description specified during the cluster creation. Starting with ADCM 2.8.0, it is possible to change the description of the cluster
Concerns	One of the following icons: ⓘ — indicates the presence of blocking concerns. ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	Icons for managing the cluster: ⚙ — opens the drop-down list with actions to manage the cluster. ⇧ — indicates whether a new version of the bundle is available and allows you to upgrade the cluster. If no version is available, the icon image is changed to ⇧. 🗑 — deletes the information about the cluster from ADCM.

NOTE

The ⚙ icon is also available on the cluster page.

To sort the clusters alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

To search for a cluster in the list, enter its name in the ***Search cluster name*** field.

To filter clusters:

- by status — select the required value from the drop-down list in the field next to ***Status***;
- by product name — select the required value from the drop-down list in the field next to ***Product***.

To reset your filters, click the ↻ icon.

Change a name and description of a cluster

To change the name of a cluster that has the `created` status, follow these steps:

1. Hover over the cluster row.
2. Click the ✎ icon located in the ***Name*** column.
3. In the dialog box, specify a new name for the cluster.
4. Click ***Save***.

When renaming a cluster, make sure that a new name meets the following requirements:

1. Consists of 2—150 characters.
2. Begins with Latin uppercase [A-Z] or lowercase [a-z] letter or digit [0-9].
3. Contains only the following characters: Latin uppercase [A-Z] or lowercase [a-z] letters, digits [0–9], period [.], hyphen [-], or underscore [_].
4. Does not end with a space, period [.], hyphen [-], or underscore [_].

To change the cluster description, follow these steps:

1. Hover over the cluster row.
2. Click the ✎ icon located in the ***Description*** column.
3. In the dialog box, specify a new description for the cluster. Note that a cluster description should not exceed 256 characters.
4. Click ***Save***.

Changing the cluster description is also available on the cluster page in the ***Description*** section.

Operations with a cluster

ADCM allows you to install and manage clusters. The installation process includes the following steps:

1. Create a cluster.
2. Add services.
3. Add hosts to a cluster.
4. Map components to hosts.
5. Configure services.
6. Configure a cluster.
7. Launch actions.
8. Upgrade a cluster.

To install a new cluster, see the following ***Get started*** guides, depending on the product in use:

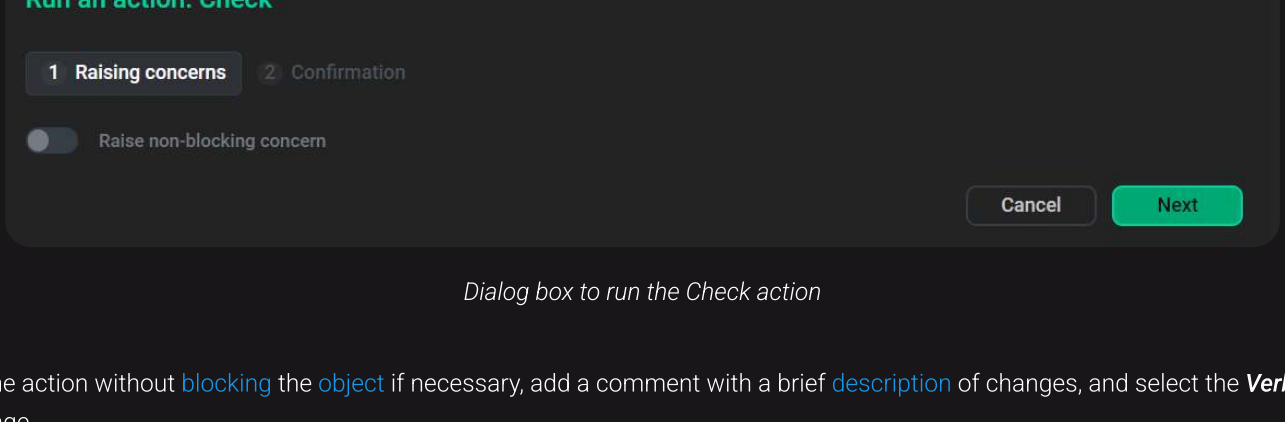
- ADB
- ADH
- ADQM
- ADQM Control
- ADS
- ADS Control
- ADPG
- ADPG Control
- ADPS

After you install the cluster, you can manage it on the ***Clusters*** tab. Click the ⚙ icon in the ***Actions*** column to select one of the available cluster actions.

For more information on actions for each product cluster, refer to the following links:

- ADB
- ADH
- ADQM
- ADQM Control
- ADS
- ADS Control
- ADPG
- ADPG Control
- ADPS

When you choose an action, ADCM displays a standard dialog box.



Run an action: Check

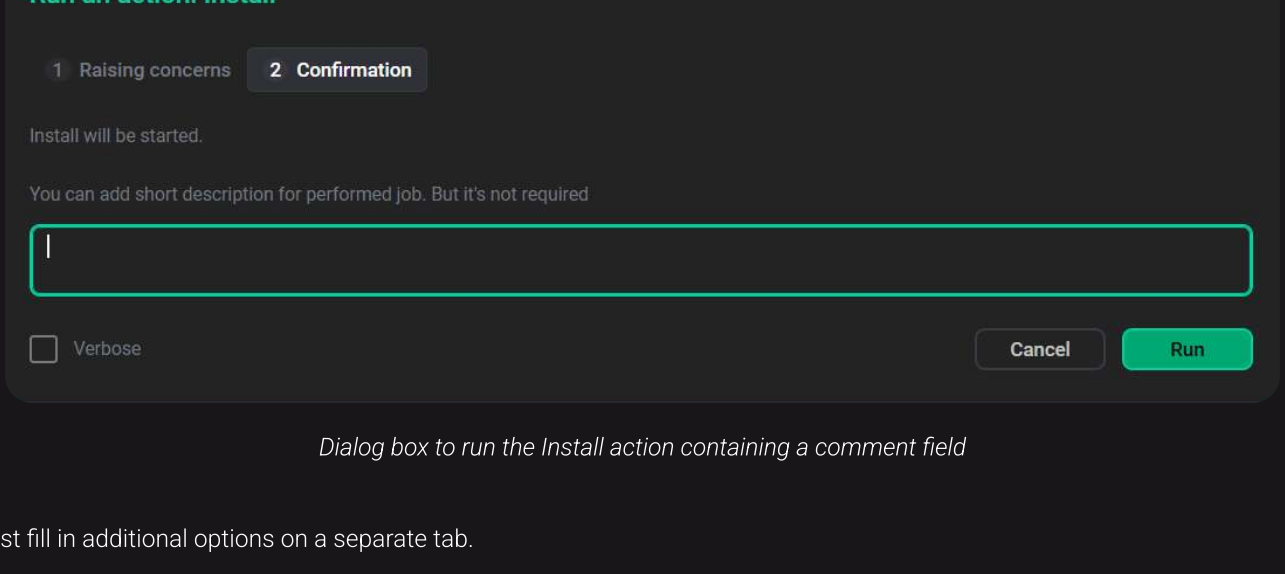
1 Raising concerns   2 Confirmation

Raise non-blocking concern

Cancel   Next

Dialog box to run the Check action

In this dialog box, you can run the action without blocking the object if necessary, add a comment with a brief description of changes, and select the ***Verbose*** checkbox to see additional execution details on the ***Jobs*** page.



Run an action: Install

1 Raising concerns   2 Confirmation

Install will be started.

You can add short description for performed job. But it's not required

☐ Verbose

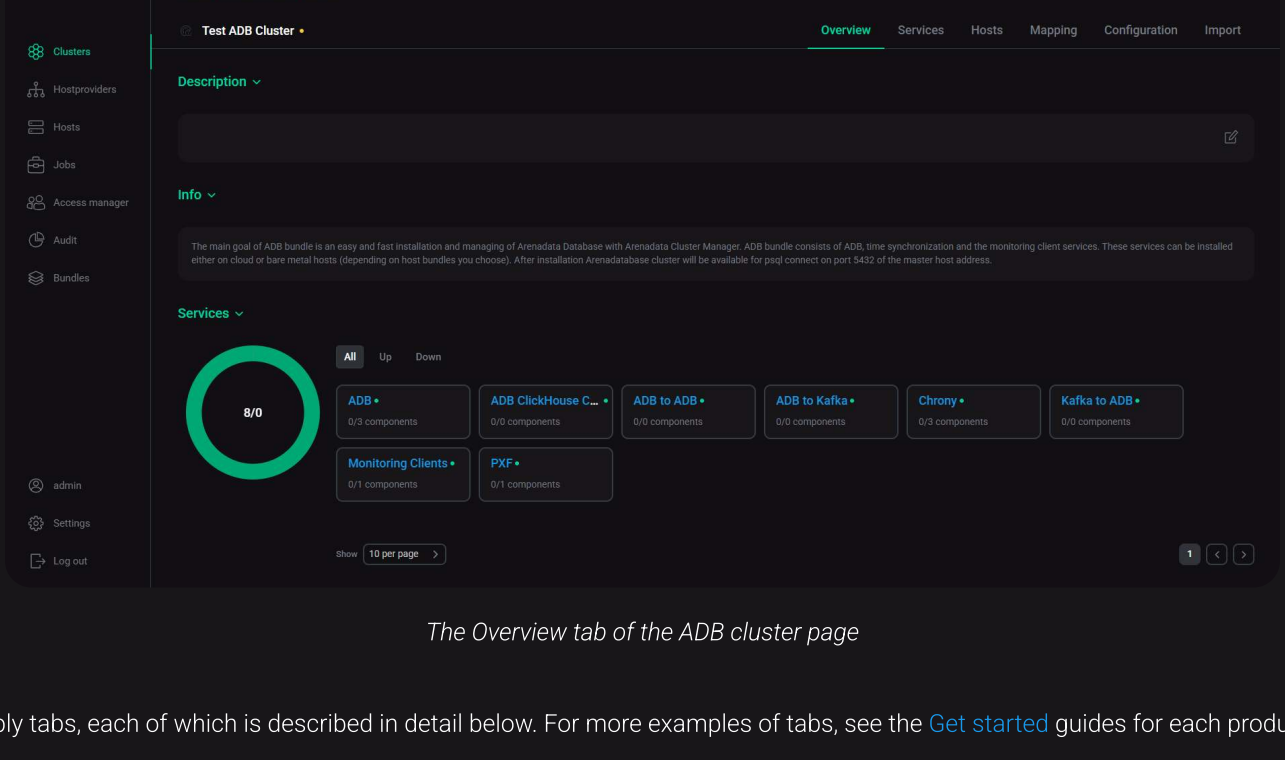
Cancel   Run

Dialog box to run the Install action containing a comment field

For some actions, you should first fill in additional options on a separate tab.

Cluster details

To get detailed information about the cluster, click on the cluster name in the ***Name*** column of the table located on the ***Clusters*** tab.



The Overview tab of the ADB cluster page

The cluster page includes multiply tabs, each of which is described in detail below. For more examples of tabs, see the Get started guides for each product.

Overview

The ***Overview*** tab on a cluster page displays detailed information on the selected cluster. The tab consists of three sections:

- ***Description***
- ***Info***
- ***Services***
- ***Hosts***

To collapse the section content, click the ⌄ icon.

The ***Description*** section contains a description of the cluster. If necessary, it is possible to change the cluster description.

NOTE

The ***Description*** section is available starting with ADCM 2.8.0.

The ***Info*** section displays general information on the purpose and composition of the cluster bundle.

The ***Services*** section displays information on the cluster services and includes the circle diagram and the following tabs:

- ***All*** — all services added to the cluster;
- ***Up*** — operational cluster services;
- ***Down*** — inactive cluster services.

NOTE

The health of an object in ADCM is determined by its state.

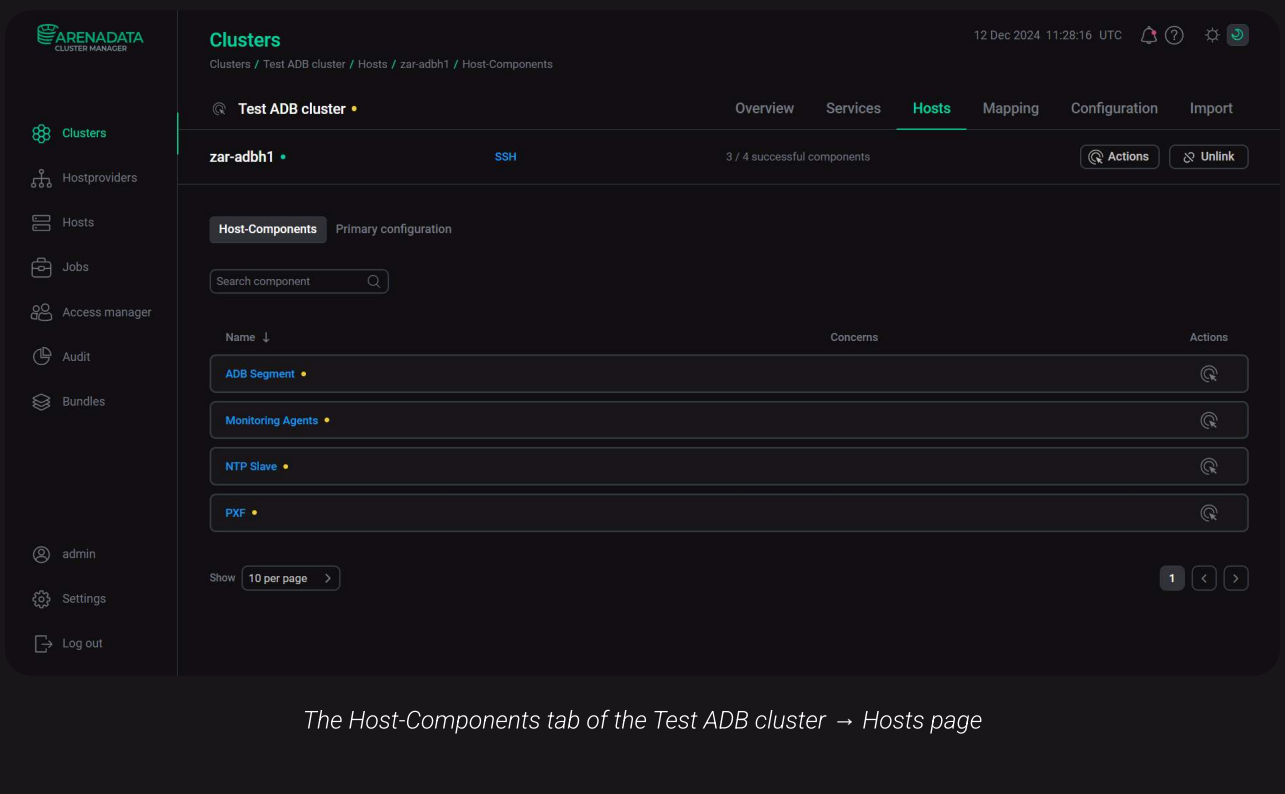
For example, to view services that can also work properly, but their status is not transferred from hosts to ADCM, switch to the ***Down*** tab. To open the ***Components*** tab of the ***<Cluster name> → Services*** page, click the service name.

The circle diagram shows the ratio between services in the `Up` status and services in the `Down` status.

The ***Hosts*** section displays information on the cluster hosts and includes the circle diagram with the following tabs:

- ***All*** — all hosts added to the cluster;
- ***Up*** — hosts that regularly send heartbeats;
- ***Down*** — hosts that do not send heartbeats.

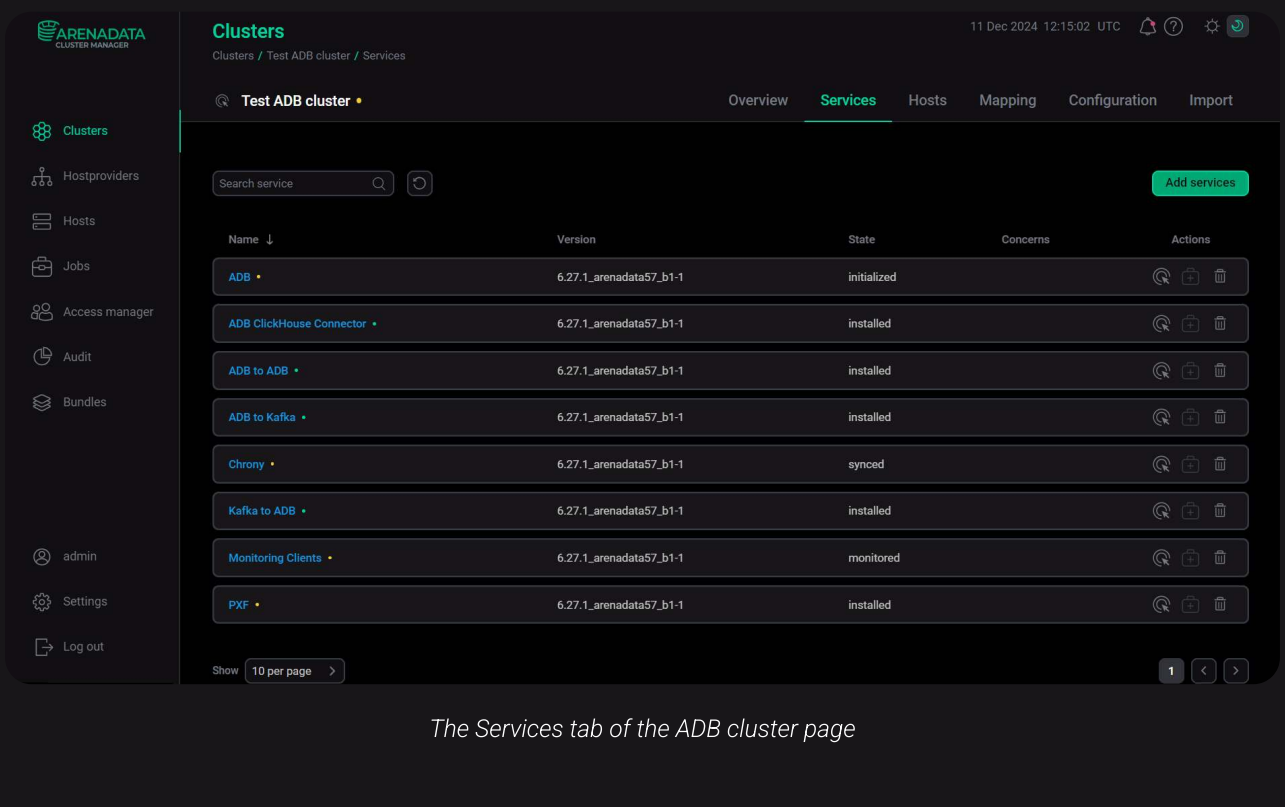
The ***Host-Components*** tab of the ***<Cluster name> → Hosts*** page opens by clicking the host name.



The Host-Components tab of the Test ADB cluster → Hosts page

Services

The ***Services*** tab on a cluster page contains information on services added to a cluster.



The Services tab of the ADB cluster page

Fields of the Services tab

Field	Description
Name	Name of the service
Version	Version of the cluster bundle used to install or upgrade the cluster
State	Current state of the service (for example, `created` or `installed`)
Concerns	One of the following icons: ⓘ — indicates the presence of blocking concerns. ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	Icons for managing the service: ⚙ — opens the drop-down list with actions to manage the service. ⊞ — turns on the maintenance mode on the service if supported by the product in use. If maintenance mode is not supported by the product, the icon image is changed to ⊞. After the action is executed, the service changes its state and the icon is changed to the ⊞ icon. In this state, the service can be transferred to normal mode. 🗑 — deletes the information about the service from ADCM. The service removal is possible after removing its components from hosts on the ***Mapping*** tab described below.

For more information on the service actions, refer to the following links, depending on the product in use:

- ADB
- ADH
- ADQM
- ADQM Control
- ADS
- ADS Control
- ADPG
- ADPG Control
- ADPS

To sort the services alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

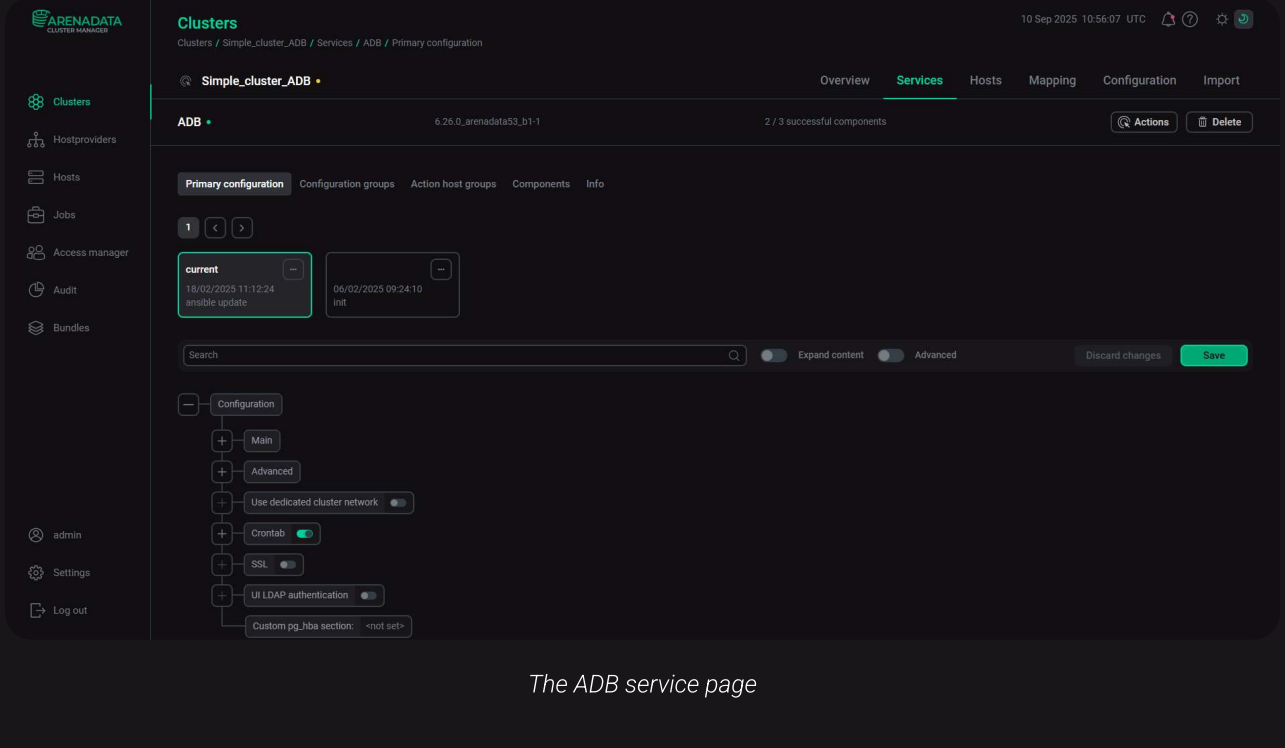
To search for a service in the list, enter its name in the ***Search service*** field.

To reset your filters, click the ↻ icon.

To add a service to a cluster, follow these steps:

1. On the cluster page, open the ***Services*** tab.
2. On the page that opens, click ***Add services***.
3. For further action, see the ***Add services*** article of the corresponding ***Get started*** guide mentioned above.

To get detailed information about the service, click on the service name in the ***Name*** column of the table located on the ***Services*** tab of a cluster page.



The ADB service page

At the top of the service page, the following is displayed:

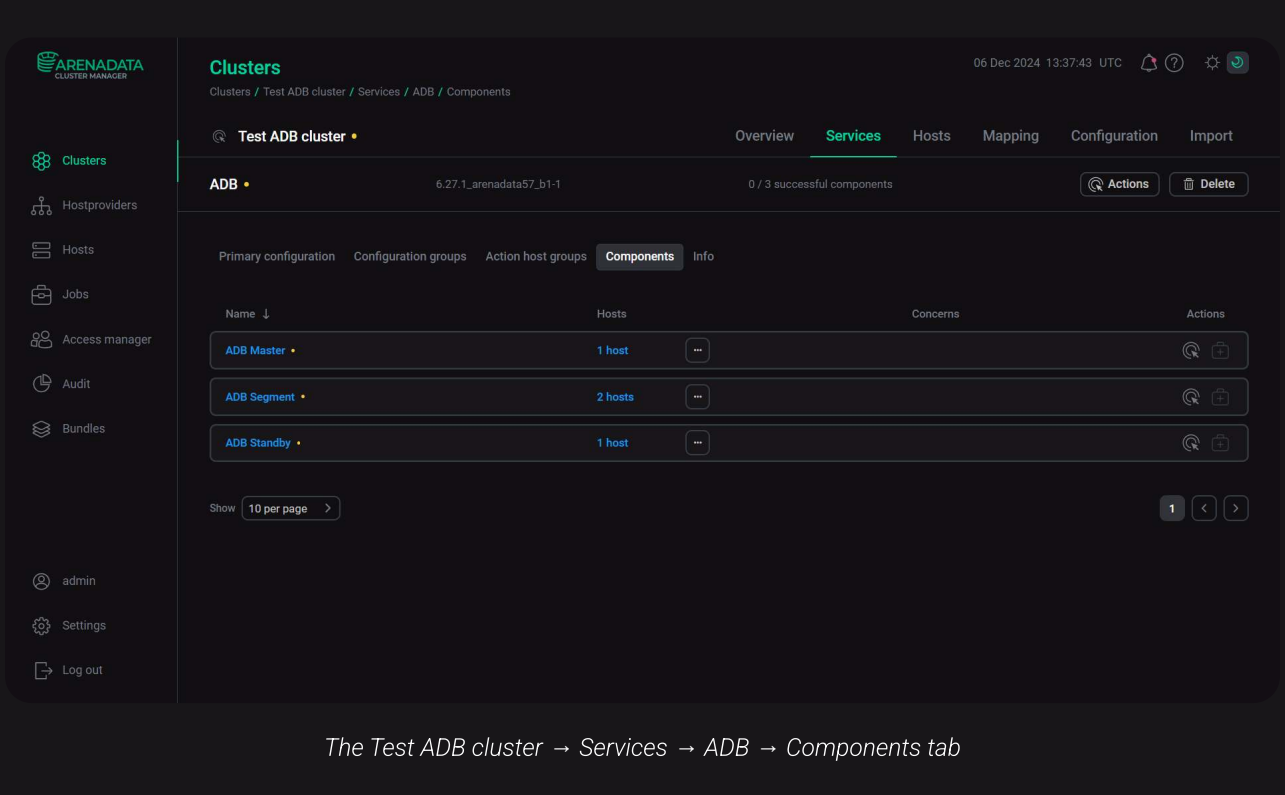
- service name with the service status;
- service version;
- ratio of components in the `Up` status to the total number of the service components;
- buttons ***Actions*** and ***Delete***.

Also, this page includes the following tabs:

- ***Primary configuration*** — allows you to configure the general service settings. The tab description and information about working with configuration parameters are given in the Configuration parameters in the ADCM UI article.
- ***Configuration groups*** — allows you to set up service configuration groups. For more information, refer to the article Set up configuration groups.
- ***Action host groups*** — enables you to create and manage host groups to run actions allowed for the service separately for each group of hosts. For more information, refer to the article Set up host groups for actions.
- ***Components*** — allows you to work with the service components. The tab description is given below. The example of the ADS components configuration is described in the ***Configure services*** article of the Get started guide.
- ***Info*** — displays general information on the purpose of the selected service.

Components

The ***Components*** tab on the ***<Cluster name> → Services → <Service name>*** page contains information on the service components.



The Test ADB cluster → Services → ADB → Components tab

Fields of the <Cluster name> → Services → <Service name> → Components tab

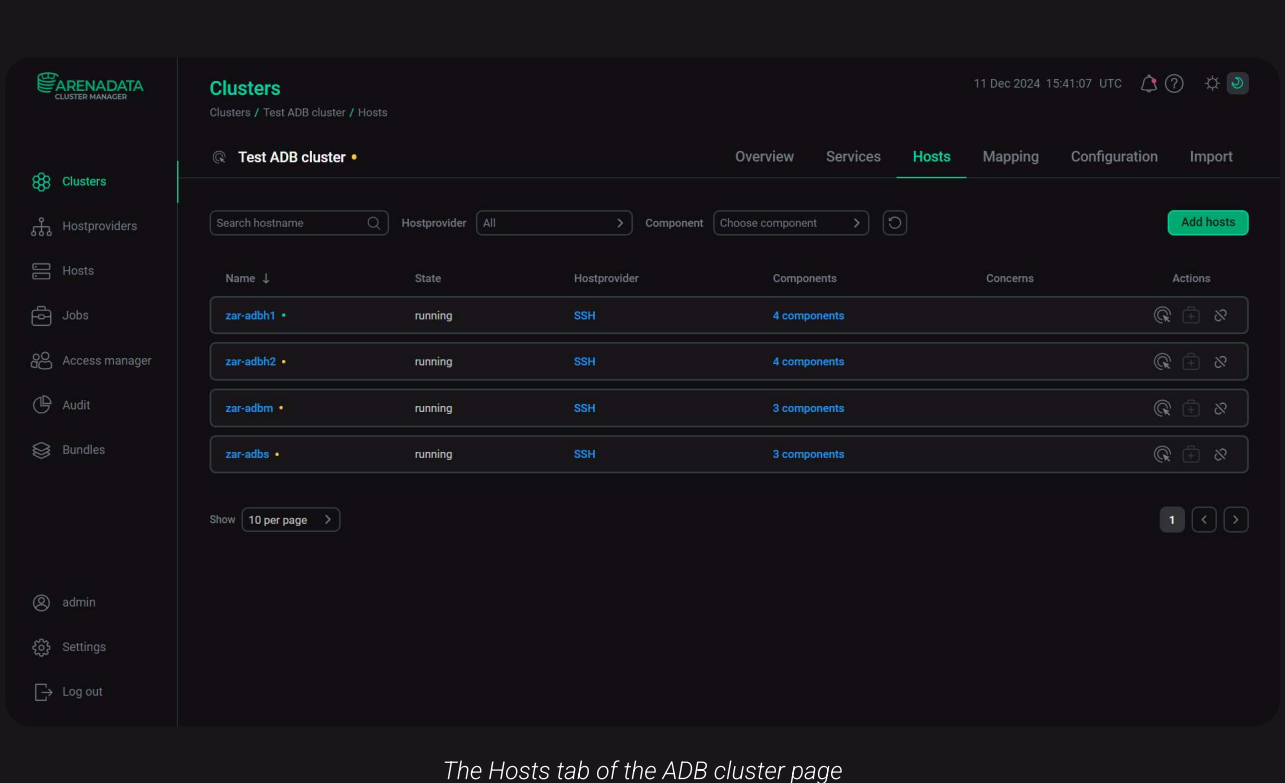
Field	Description
Name	Name of the component that contains a link to the ***Primary configuration*** tab of the ***<Cluster name> → Services → <Service name> → Components*** page
Hosts	Number of hosts assigned for the component that contains a link to the Hosts tab
Concerns	One of the following icons: ⓘ — indicates the presence of blocking concerns. ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	Icons for managing the component: ⚙ — opens the drop-down list with actions to manage the component. ⊞ — turns on the maintenance mode on the component if supported by the product in use. If maintenance mode is not supported by the product, the icon image is changed to ⊞. After the action is executed, the component changes its state and the icon is changed to the ⊞ icon. In this state, the component can be transferred to normal mode.

To sort components alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

To expand or collapse the list of hosts assigned for the component under the component row, click the ⋯ or ⋯ icon respectively.

Hosts

The ***Hosts*** tab on the cluster page contains information on hosts added to a cluster.



The Hosts tab of the ADB cluster page

Fields of the Hosts tab

Field	Description
Name	Name of the host that contains a link to the ***Host-Components*** tab of the <Cluster name> → Hosts → <Host name> page
State	Current state of the host (for example, `running`)
Hostprovider	Name of the hostprovider used to create the host that contains a link to the ***Primary configuration*** tab of the hostprovider page
Components	Number of components added to the host that contains a link to the ***Host-Components*** tab of the <Cluster name> → Hosts → <Host name> page
Concerns	One of the following icons: ⓘ — indicates the presence of blocking concerns. ⓘ — indicates the presence of non-blocking concerns. A pop-up window with the error description appears when hovering over the ⓘ or ⓘ icon
Actions	Icons for managing the host: ⚙ — opens the drop-down list with actions to manage the host. ⊞ — turns on the maintenance mode on the host if supported by the product in use. If maintenance mode is not supported by the product, the icon image is changed to ⊞. After the action is executed, the host changes its state and the icon is changed to the ⊞ icon. In this state, the host can be transferred to normal mode. 🔗 — adds the host to a cluster. After the action is executed, the icon is changed to the 🔗 icon which performs the opposite action.

For more information on actions for hosts based on different hostprovider types, refer to the Operations with host section.

To sort the hosts alphabetically, click the ↓ (from Z to A) or ↑ (from A to Z) icon in the ***Name*** column.

To search for a host in the list, enter its name in the ***Search hostname*** field.

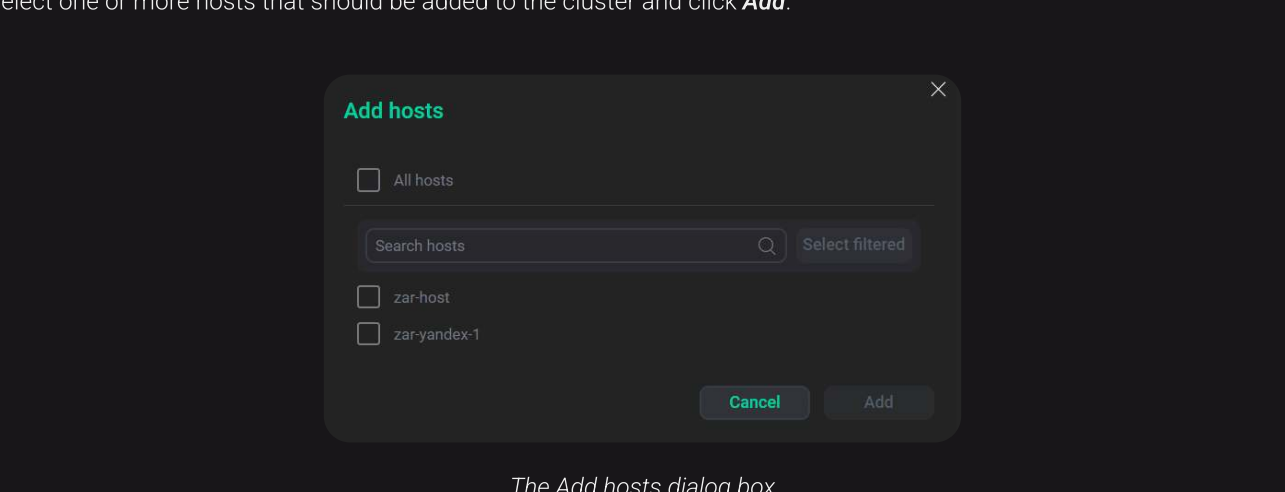
To filter hosts:

- by hostprovider name — select the required value from the drop-down list in the field next to ***Hostprovider***;
- by component name — select the required value from the drop-down list in the field next to ***Component***.

To reset your filters, click the ↻ icon.

To add one or more hosts to a cluster, follow these steps:

1. On the cluster page, open the ***Hosts*** tab.
2. On the page that opens, click ***Add hosts***.
3. In the opened dialog box, select one or more hosts that should be added to the cluster and click ***Add***.



Add hosts

☐ All hosts

Search hosts   Select filtered

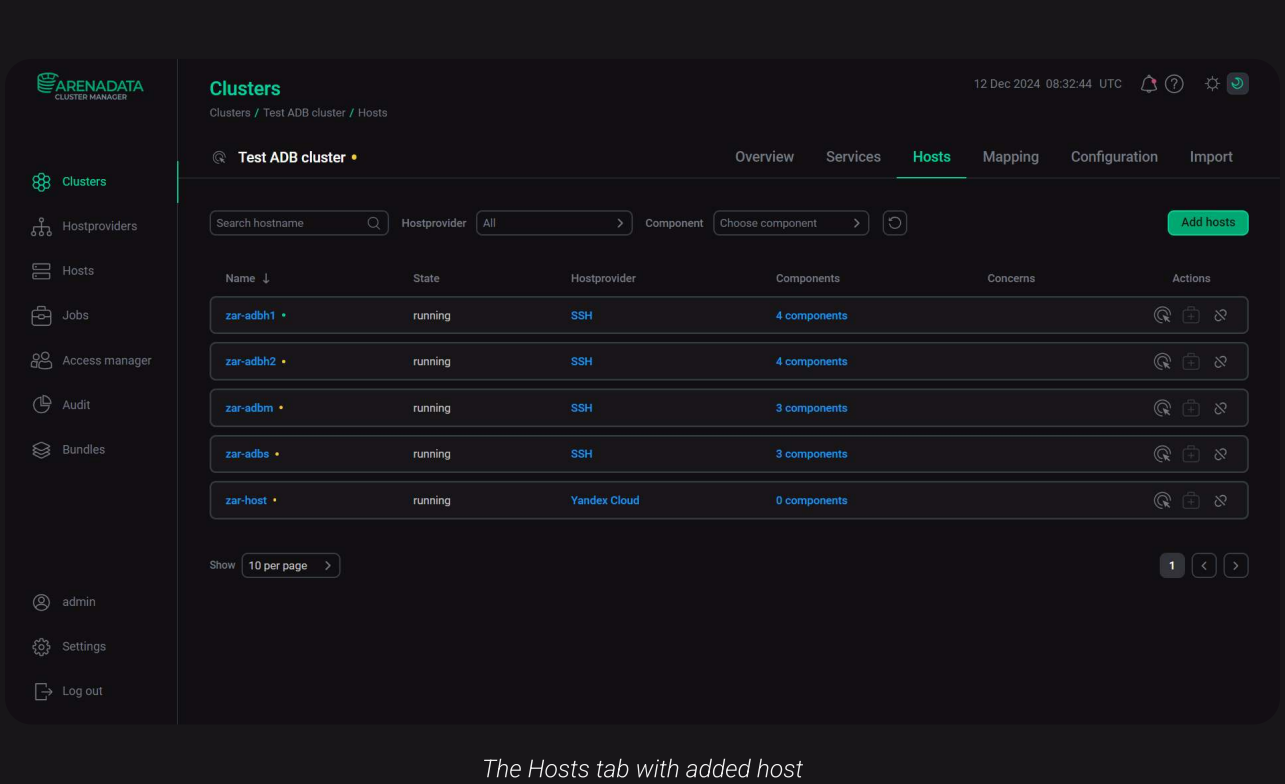
☐ zar-host

☐ zar-yandex-1

Cancel   Add

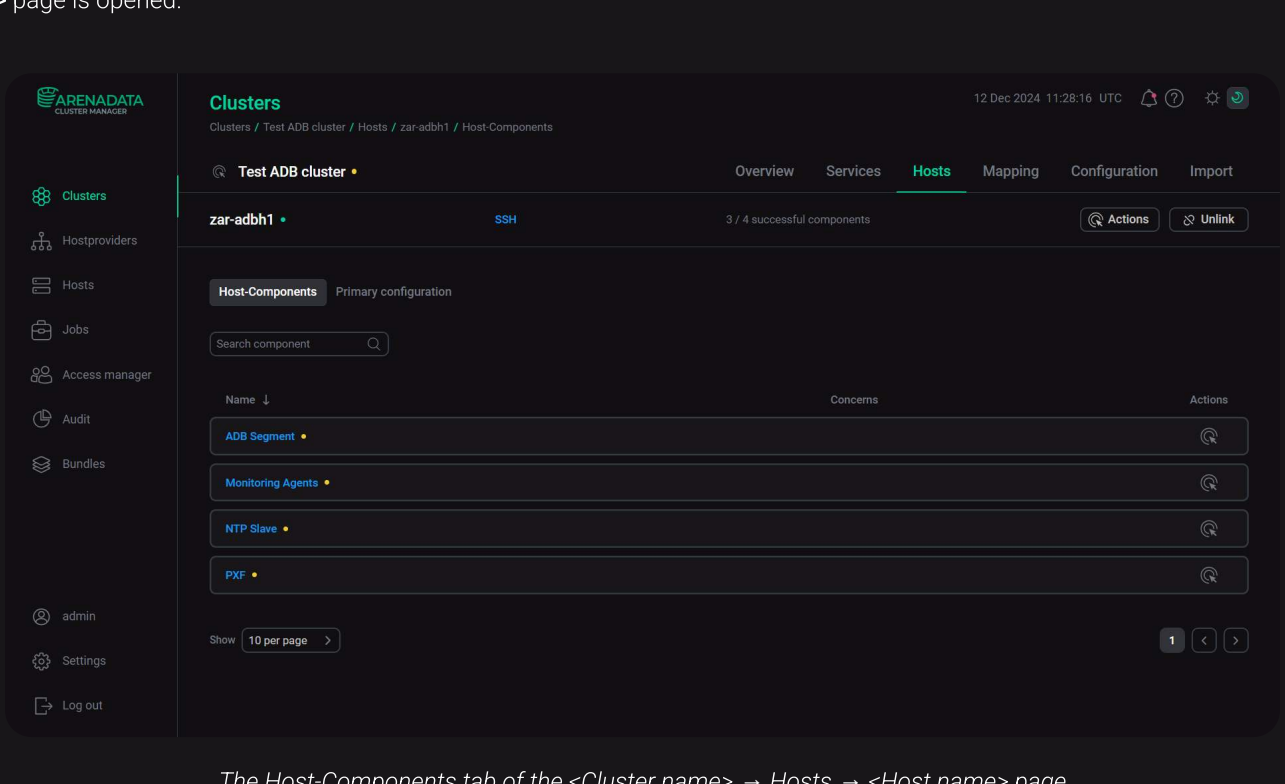
The Add hosts dialog box

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



The Hosts tab with added host

To get detailed information about the host, click on the host name in the ***Name*** column of the table located on the ***Hosts*** tab of a cluster page. The ***Host-Components*** tab of the ***<Cluster name> → Hosts → <Host name>*** page is opened.

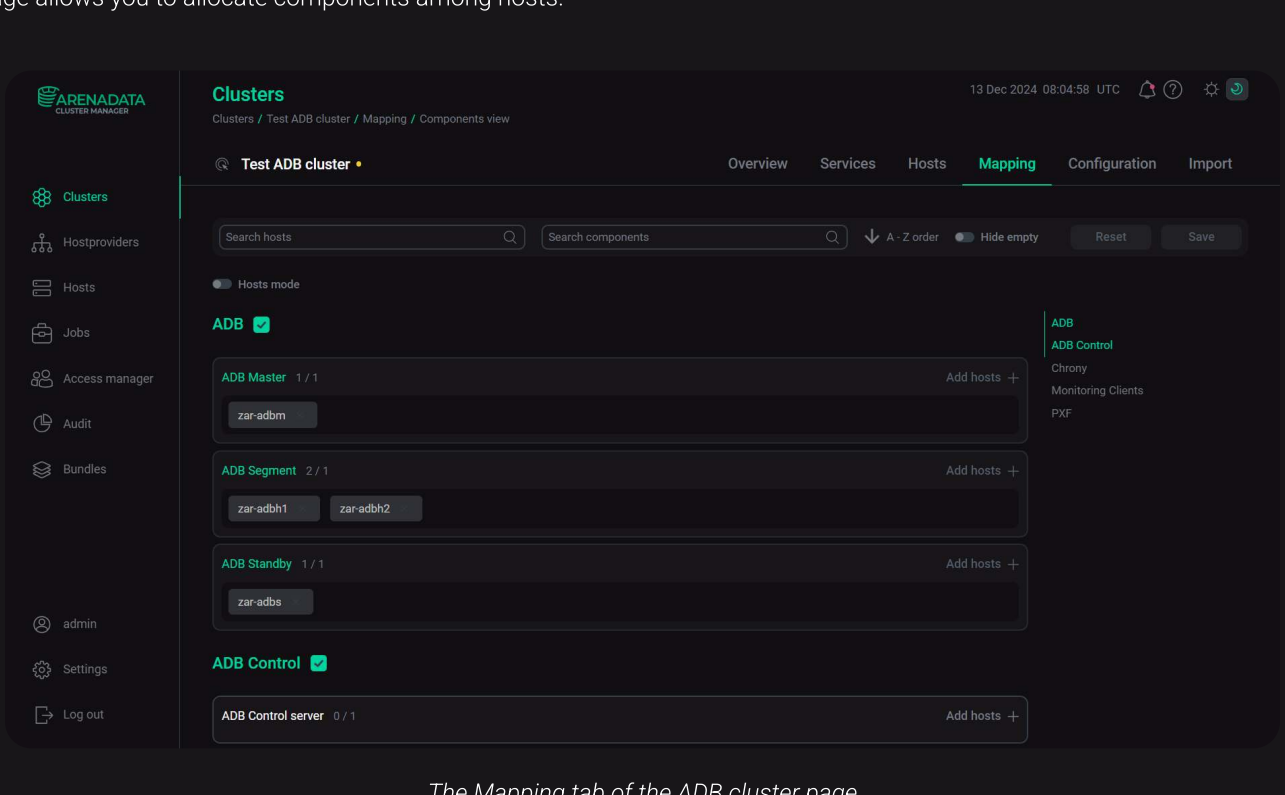


The Host-Components tab of the <Cluster name> → Hosts → <Host name> page

For more information about the host page, refer to the Host details section.

Mapping

The ***Mapping*** tab on a cluster page allows you to allocate components among hosts.



The Mapping tab of the ADB cluster page

For more information on how to allocate components among hosts, see the ***Add components*** article of the corresponding ***Get started*** guide mentioned above.

To search for a host or component in the list, enter the host or component name in the ***Search hosts*** or the ***Search components*** field respectively.

To sort components or hosts (depending on the allocation mode selected in the ***Hosts mode*** field):

- in ascending (from A to Z) alphabetical order, click the ↑ icon next to ***A-Z order***.
- in descending (from Z to A) alphabetical order, click the ↓ icon next to ***A-Z order***.

To hide components without assigned hosts or hosts that do not contain components, switch on the ***Hide empty*** toggle.

To save changes in allocating components among hosts, click ***Save***. To revert changes, click ***Reset***.

To select the other way of allocating components among hosts, switch on the ***Hosts mode*** toggle.

Configuration

The ***Configuration*** tab on a cluster page is designed to configure the cluster after its creation.

This page includes the following tabs:

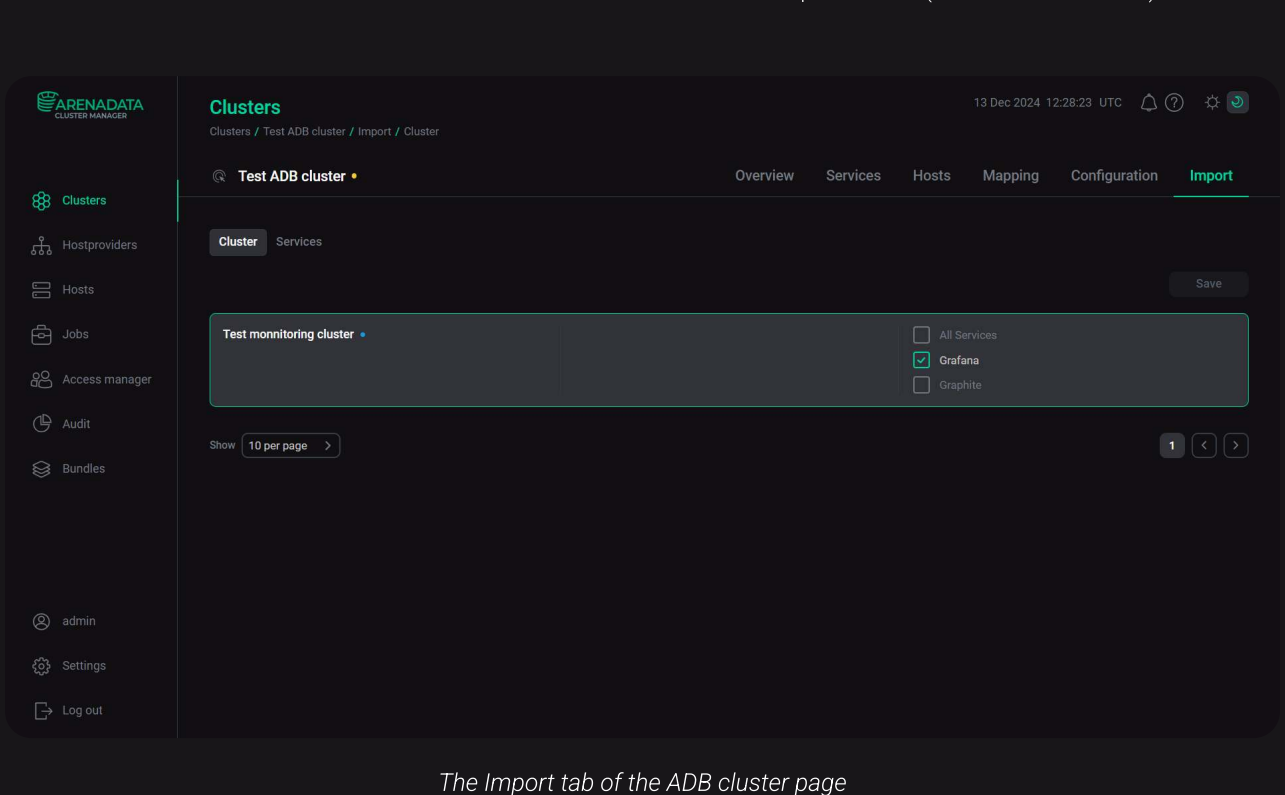
- ***Primary configuration*** — allows you to configure the general cluster settings. For more information on configuration parameters, refer to the ***Configure a cluster*** article of the Get started guide, depending on the product in use.
- ***Configuration groups*** — allows you to set up cluster configuration groups.
- ***Ansible settings*** — allows you to set up Ansible configuration options at the cluster level. The tab is available starting with ADCM 2.2.0. The tab content controls are the same as for the ***Primary configuration*** tab. For detailed information on how to work with configuration parameters, see Configuration parameters in the ADCM UI.
- ***Action host groups*** — enables you to set up host groups to run actions allowed for the cluster separately for each group of hosts.

The ***Primary configuration***, ***Configuration groups***, and ***Action host groups*** are completely identical to tabs of a service page described above.

Import

The ***Import*** tab on a cluster page allows you to import the following settings:

- settings of other clusters or services from other clusters to the current cluster (the ***Cluster*** section);
- settings of other clusters or services from other clusters to a service of the current cluster from the drop-down list (the ***Services*** section).



The Import tab of the ADB cluster page

For an example of how to import settings of the monitoring cluster services to the ADS cluster, see Import/export of services.

To save import changes, click ***Save***. The following message will be displayed: `Imported objects have been updated`.

Import/export of services

Mikhail Serov

Contents

To Table of Contents

Import tab

Mandatory imports

Example of cluster imports

Example of service imports

ADCM offers the import and export of services. For a service, **import** is essentially a transfer of the specific service settings. The service settings being imported are listed in the product bundle, precisely, in the `import` section of the service description.

The settings that are available for import are called **exported settings**. The exported settings for a service are listed in the `export` section of the service description.

Import defines what settings a service is able to receive from other services. Hence, no service is imported or exported physically. Only the settings are being transferred.

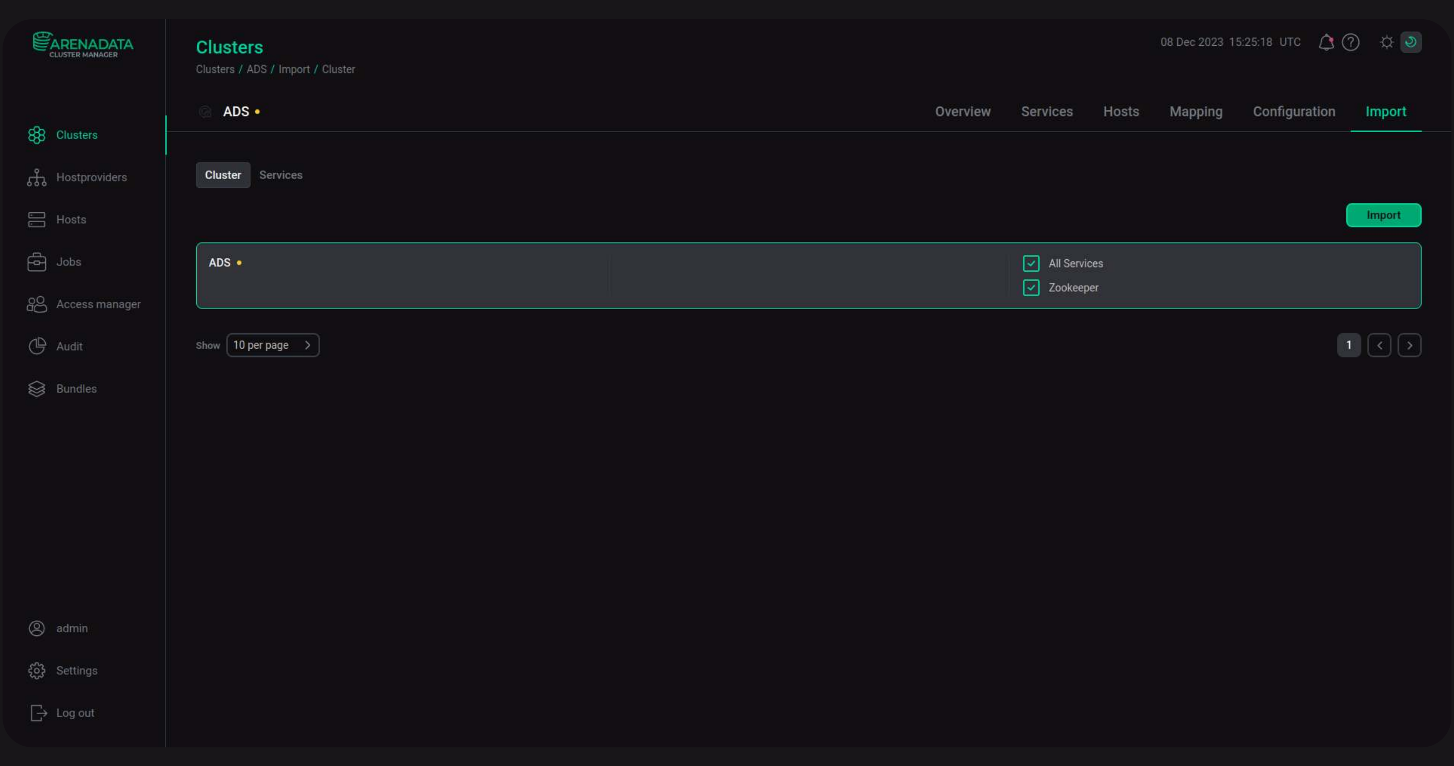
Consider a `hadoop` cluster that contains a `ranger` service able to be imported from another cluster. Then the `ranger` service will contain the exported settings in the `export` section of the `ranger` service description. Those settings become available for the `ranger` service when it is imported into other clusters and can now be accessed at configuration level.

> **IMPORTANT**
>
> Imports are product-specific and are not included into a product by default. Imports are present in a product only if imports are explicitly declared in the product bundle.

Everything that concerns the import and export of services is also true for clusters.

Import tab

You can view the cluster imports on the ***Import*** tab.



The Import tab

To change the set of service settings that are being imported, click a checkbox in the relevant service. To save the changes, click ***Import***.

> **NOTE**
>
> The export of services is not represented in the ADCM interface. You can view the service's exported settings in the service description. The description contains sections and configurations that are being exported.

Mandatory imports

If a service requires some imports, those imports are called the **mandatory imports**.

Suppose a cluster is created and the `monitoring` service is added to this cluster. The monitoring client inside a cluster is useless by itself — the client is able to accumulate data, yet it does not know where to send it. To send data, the monitoring client needs the information from an external service, which, essentially, what an import is.

The main idea behind the import of services is to allow the re-use of service settings. ADCM might contain a lot of clusters and for them not to contain the similar services physically, the import of just their settings is allowed.

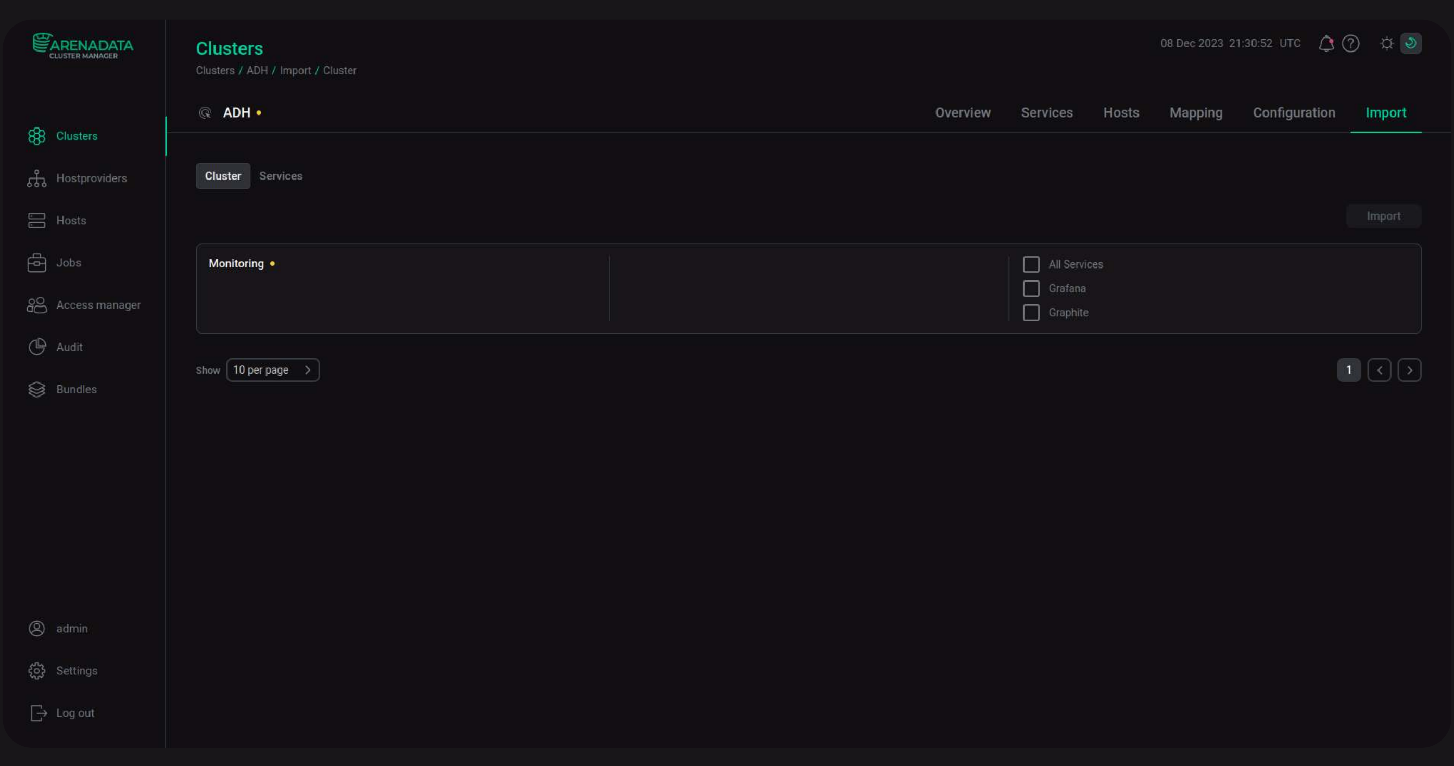
For example, there are some common services, such as the `monitoring` service or the `ranger` service, that are frequently needed. Those services become candidates for re-use, so a standalone `monitoring` cluster is created for all other clusters to access. This allows you to avoid the clusters containing the services they do not directly need.

Example of cluster imports

Suppose there is an `ADH` cluster that contains various services. It is advisable that the `ADH` cluster receives configurations from the `Monitoring` cluster — that might be useful for migration tasks or to integrate with some external services. This requires that the `Graphite` or `Grafana` settings should be imported.

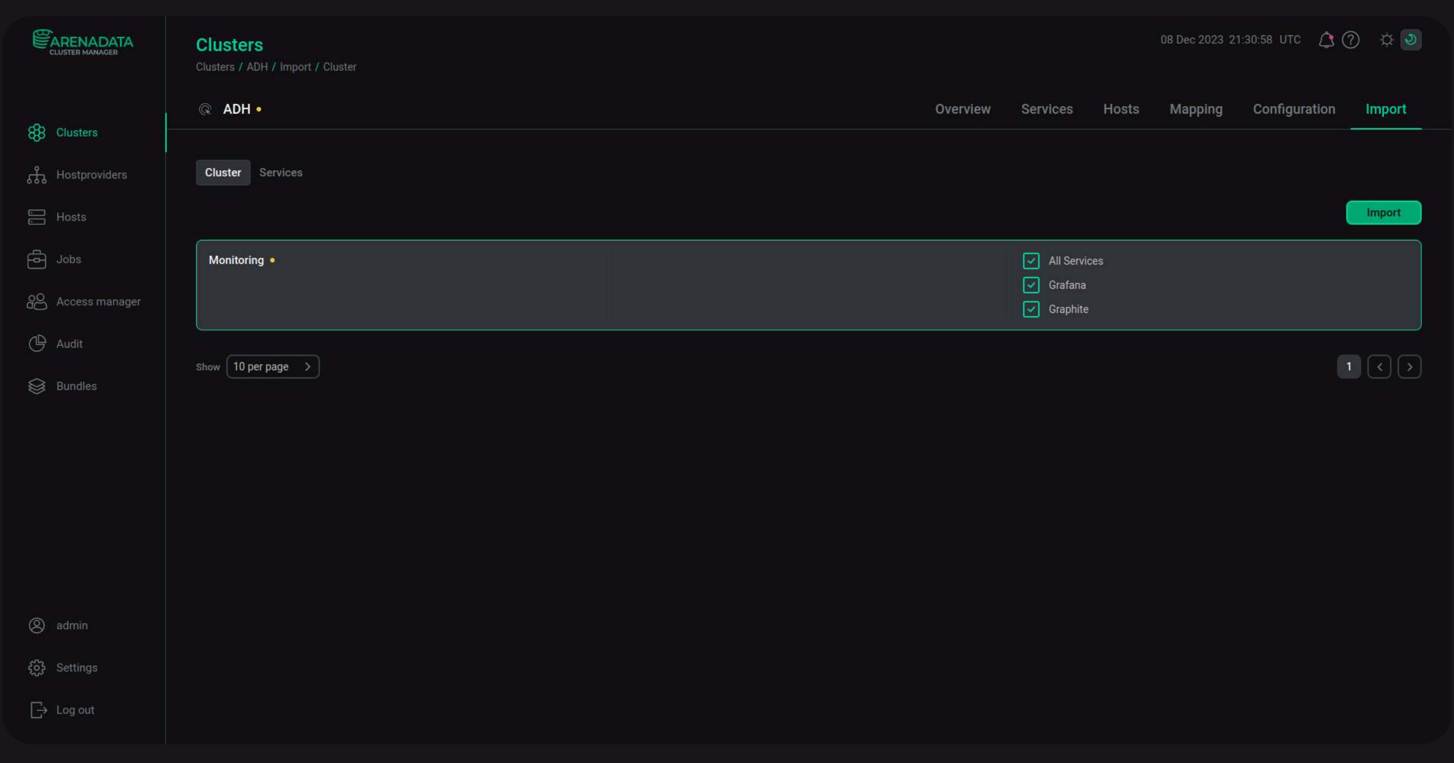
To change the settings that are being imported, do the following:

1. Navigate to the ***Import*** tab of the `ADH` cluster.



The first step of changing imports

2. The ***Import*** tab shows the settings that are available for import. You can change what is being imported by clicking a checkbox. To save the changes, click ***Import***.



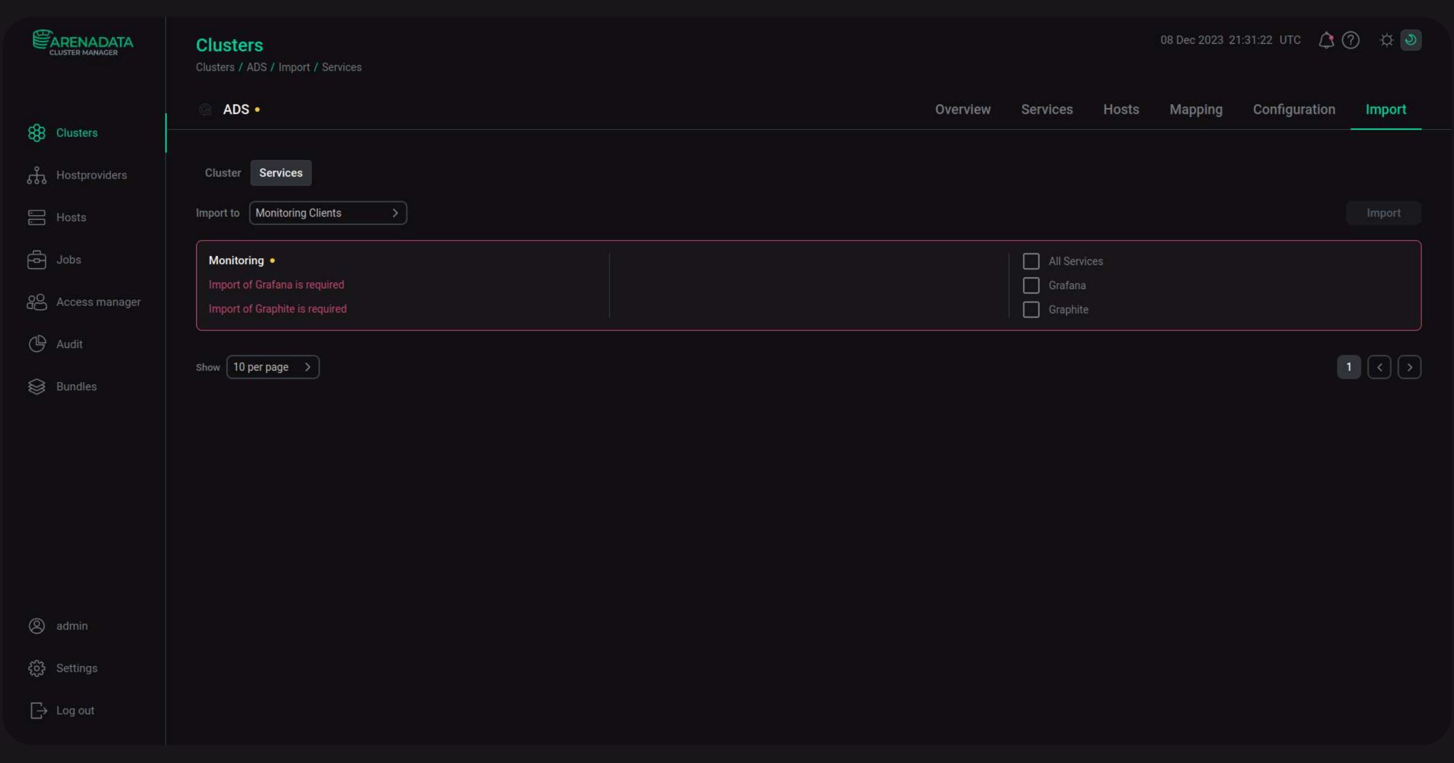
The second step of changing imports

Example of service imports

Suppose there is an `ADS` cluster that contains various services. It is advisable that the `Monitoring Clients` service of the `ADS` cluster receives configurations from the `Monitoring` cluster. This requires that the `Graphite` or `Grafana` settings should be imported. Those settings will be mandatory imports.

To change the set of settings that is being imported, do the following:

1. Navigate to the ***Import*** tab of the `ADS` cluster. Select the ***Services*** section, then select `Monitoring Clients` in the ***Import to*** field.

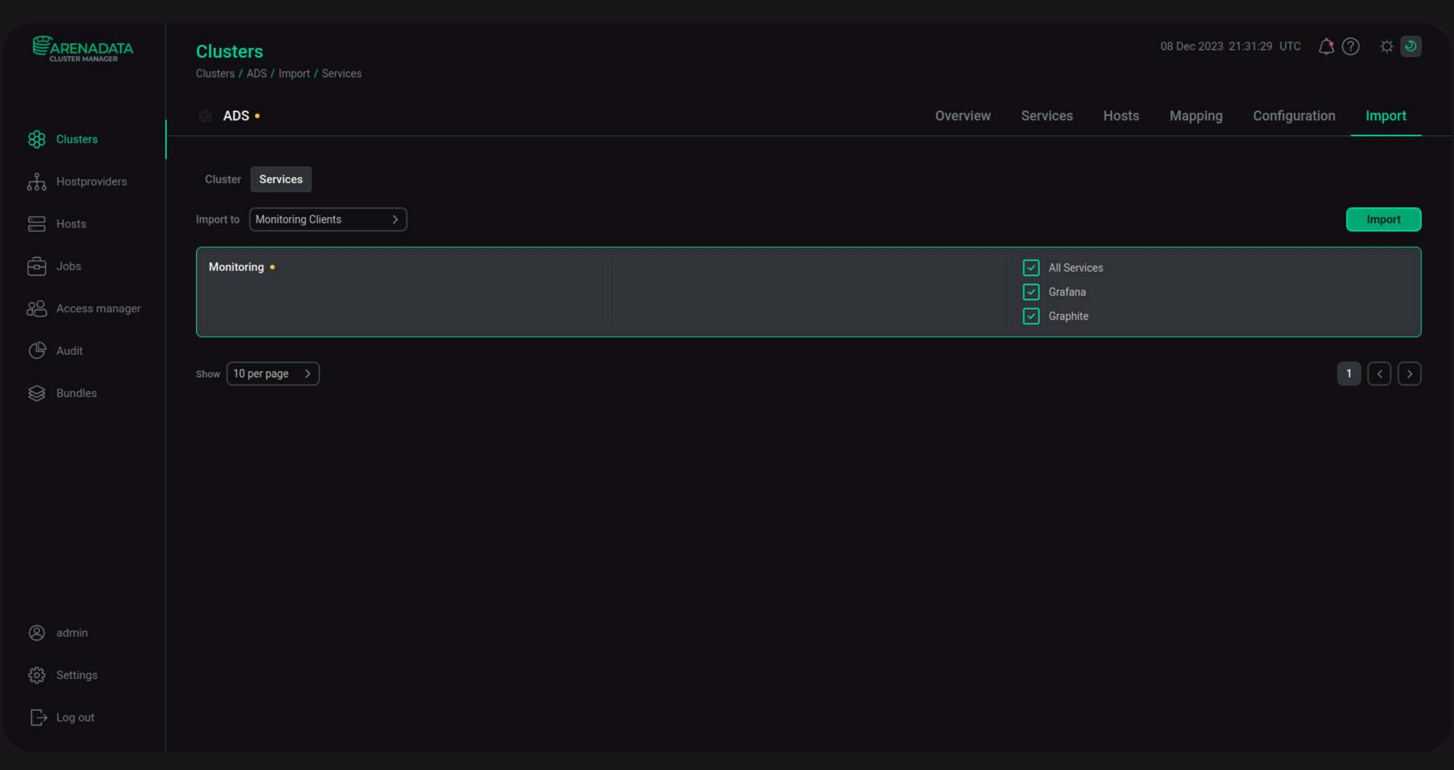


Import of Grafana is required

Import of Graphite is required

The first step of changing imports

2. In this case, the settings are mandatory for import. You can change what is being imported by clicking a checkbox. To save the changes, click ***Import***.



The second step of changing imports

Set up configuration groups

Mikhail Serov, Adelya Zvyagintseva

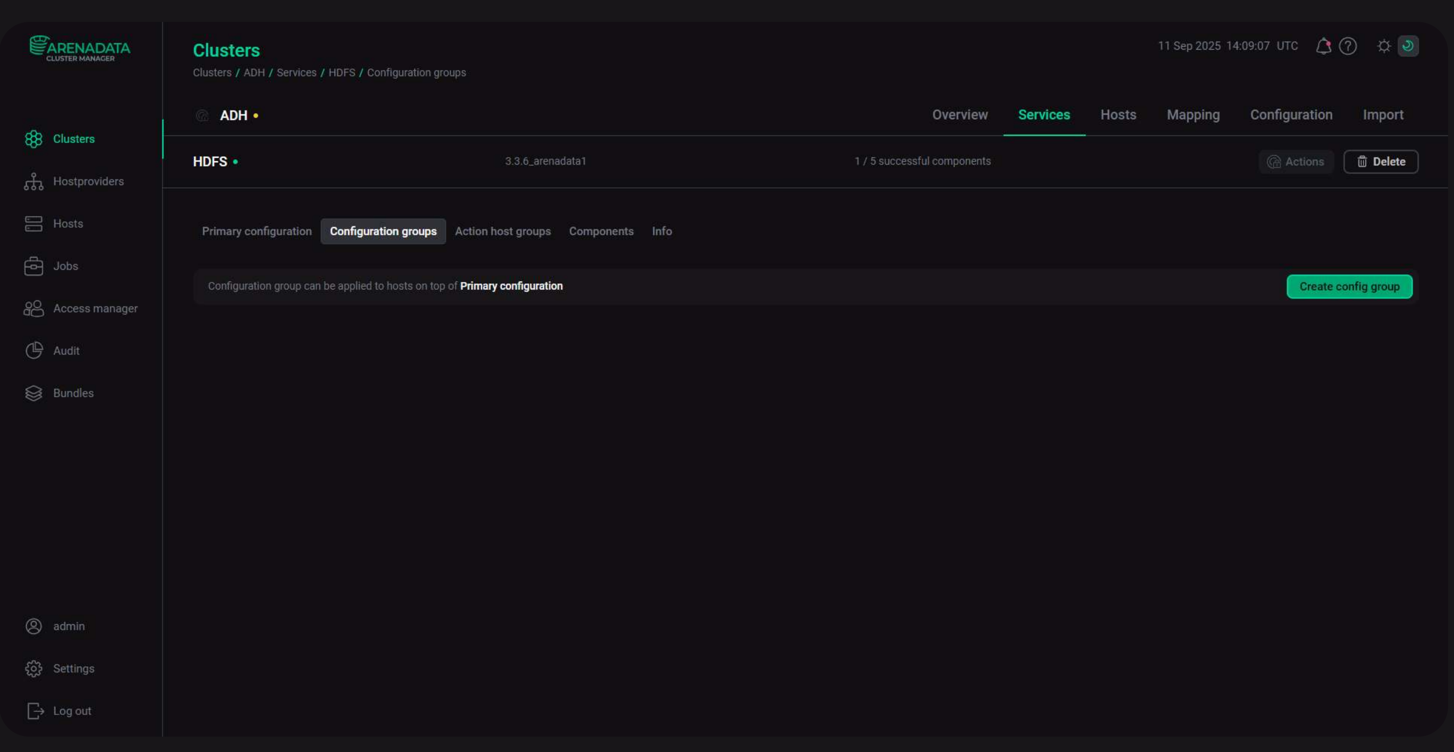
To Table of Contents

Object (cluster, service, component) general configuration is applicable to all hosts related to that object. The purpose of an object's configuration group is to set a parameter for a particular host (or a group of hosts) that is different from the parameters of object's general configuration.

Configuration (or config) group is created via ADCM interface. After a config group is created, it is populated by hosts. Configuration parameters obtain checkboxes that provide an ability to configure a parameter at a host (or a group of hosts). As a result, for a host that belongs to a config group, all parameters are identical to their general configuration versions except those parameters that are changed via a config group.

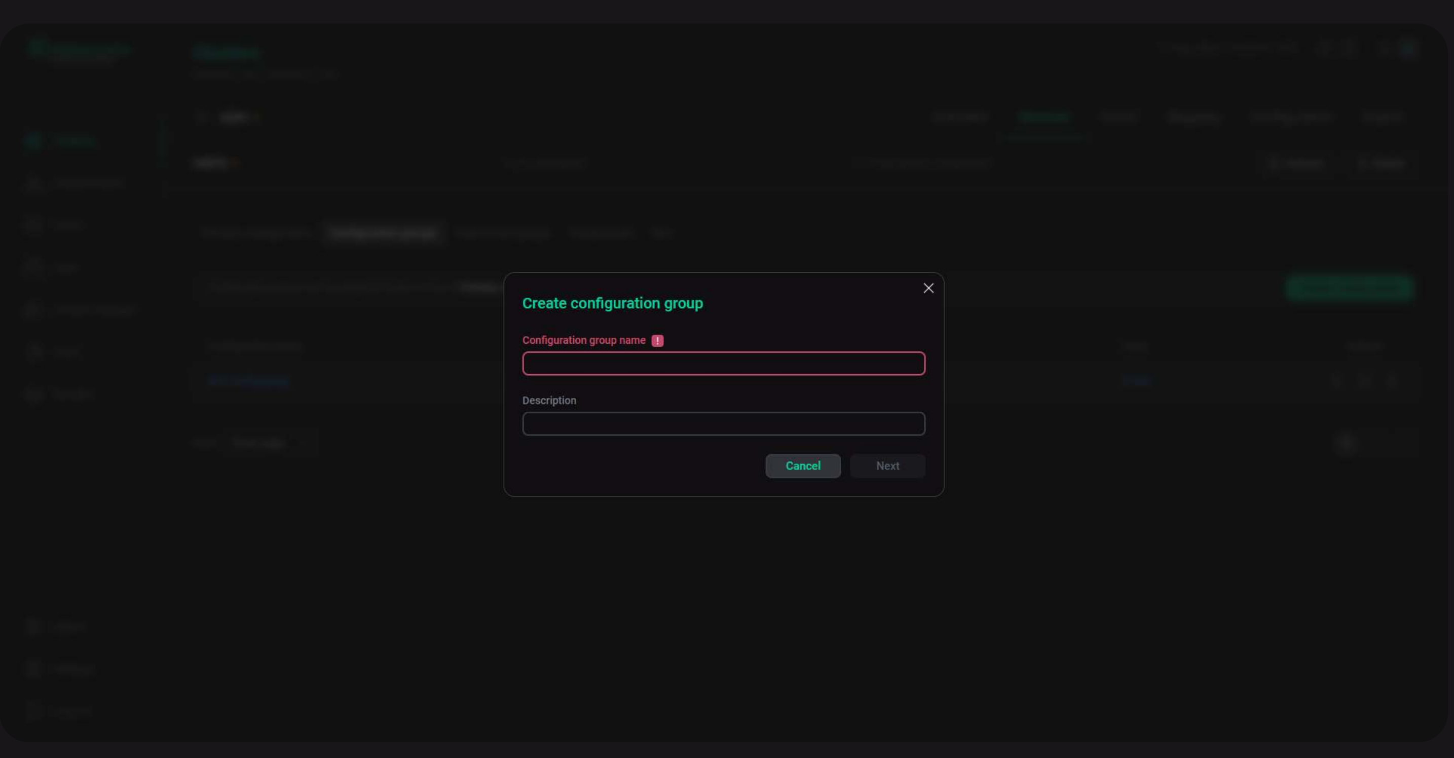
To create a config group for a service, perform the following actions:

1. Navigate to the subsection of the service (for example, HDFS) that is installed in the selected cluster, and select the ***Configuration groups*** tab.



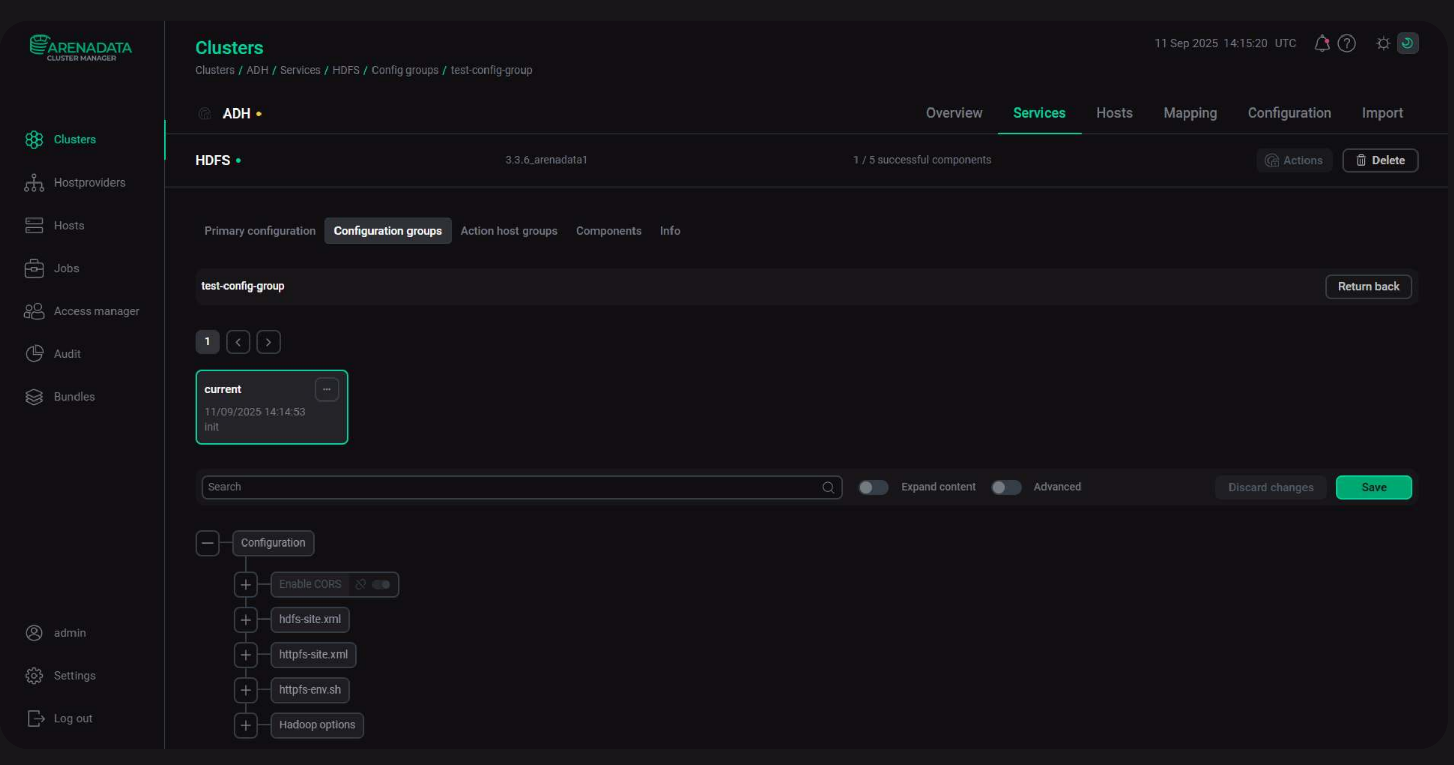
Empty configuration group list in the service subsection

2. Click ***Create config group***. The ***Create configuration group*** window opens.



The Create configuration group window

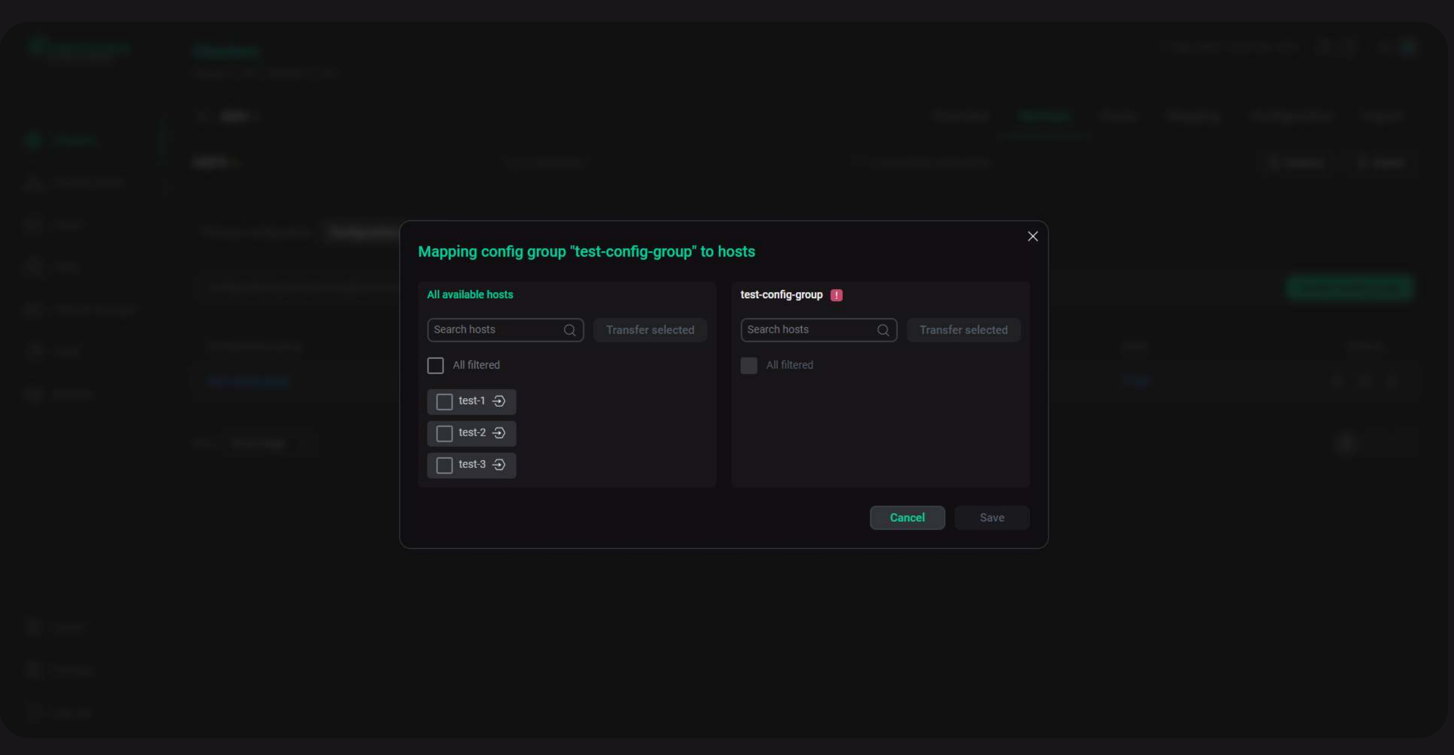
3. Enter the name of a new config group in the ***Configuration group name*** field. The ***Description*** field is optional. Click ***Next***. The page that contains configuration parameters of a new config group opens.



Configuration parameters of a new config group

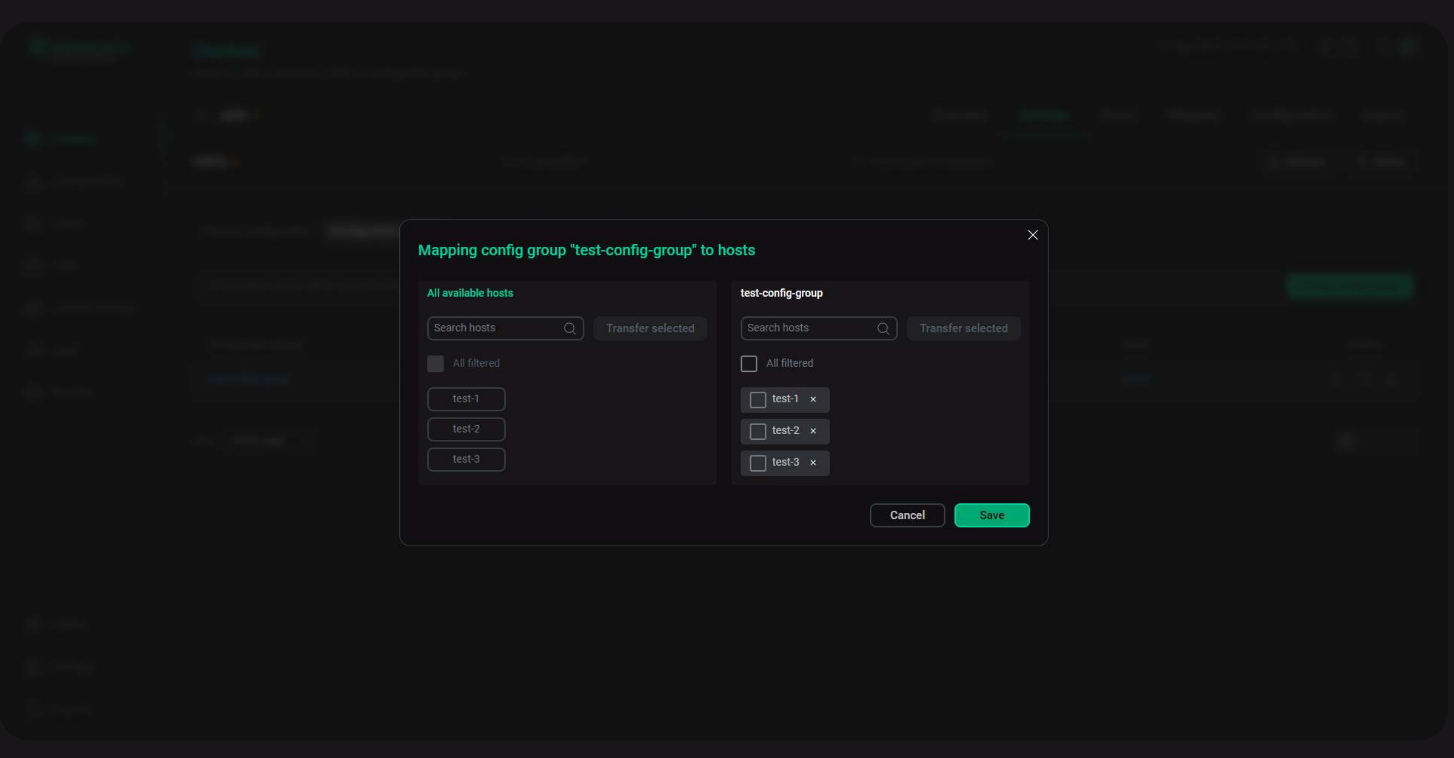
To add hosts to a new config group, perform the following actions:

1. Click ***Return back*** on the page with config group configuration parameters.
2. In the config group row, click the ◎ icon. The window for mapping config group to hosts opens.



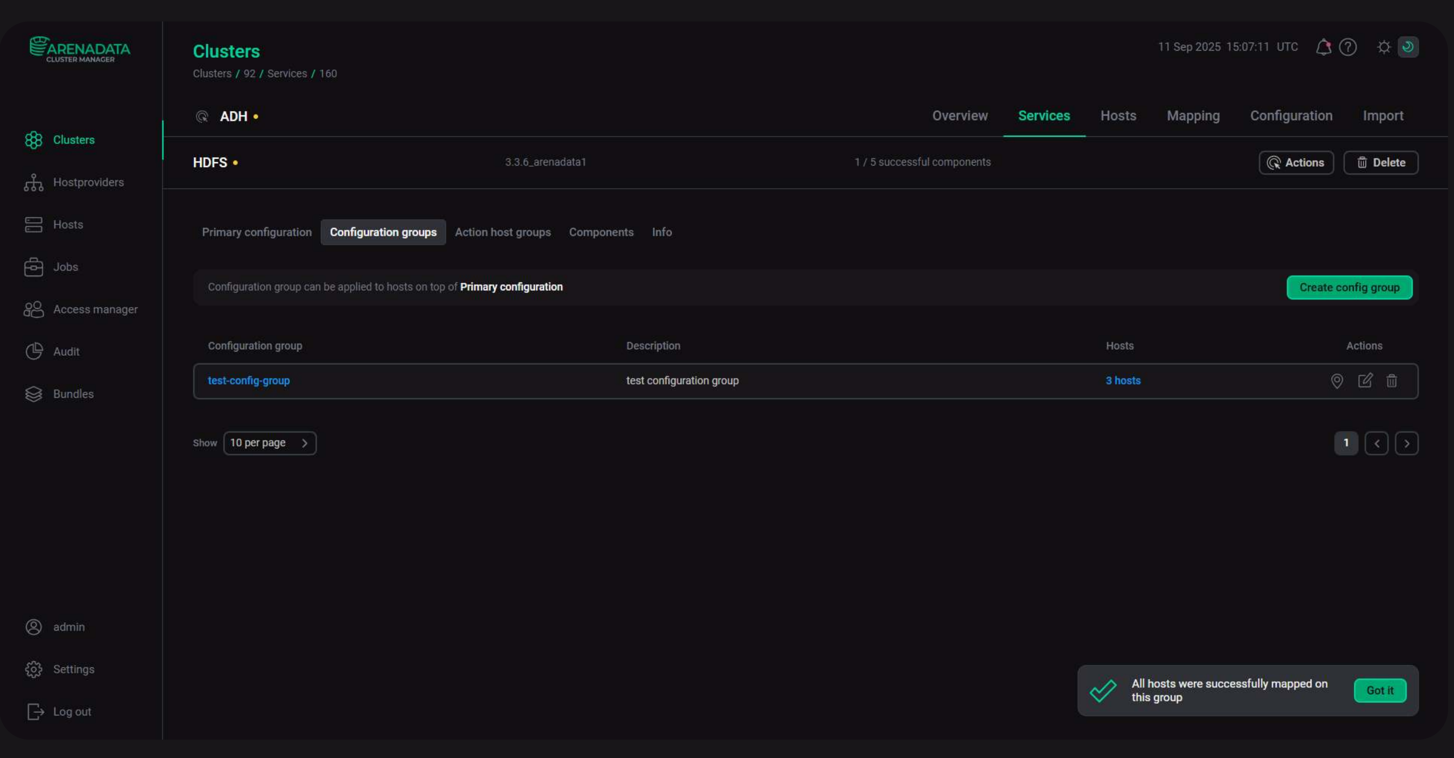
Window for mapping config group to hosts

3. Select the hosts and click ***Transfer selected***.



Config group with added hosts

4. Click ***Save*** to save the mapping. The config group now contains hosts.



Configuration group list in the service subsection

To configure parameters of a new config group, perform the following actions:

1. Click a new config group name.
2. Select the parameters whose values you want to change.
3. Edit values and click ***Save***.

You can add all hosts where the service is installed to a new config group, yet then the configuration parameters also will be changed for all hosts. This can be done without creating a new config group using the general service configuration instead.

When creating and working with config groups, consider the following:

- Config group updates are synchronized with the general service configuration. The changes in the general service configuration are inherited in the config group (except those parameters that are changed).
- It is not possible to add a single host into different config groups. Config groups of different objects do not overlap. A single host may belong to different config groups when it relates to different objects.
- Not all parameters can be changed via config groups. In such cases, the corresponding checkbox appears turned off.

IMPORTANT

You can create config groups for various ADCM objects. The functionality of config groups is available for such objects as hostprovider, cluster, service, and component. For example, the components of the service that has a config group might be in their own config groups.

Set up host groups for actions

Elena Dvoryadkina

Contents

To Table of Contents

Create a host group

Manage host groups

Run an action for a host group

In the settings of a cluster, service, or component, you can combine hosts into groups to be able to run actions allowed for a given object separately for each group of hosts. This can simplify cluster management and various automation tasks (for example, you can sequentially upgrade ClickHouse on ADQM hosts without a long cluster shutdown — see Rolling upgrade for details).

IMPORTANT

Host groups for actions are only available if the product in use supports this functionality.

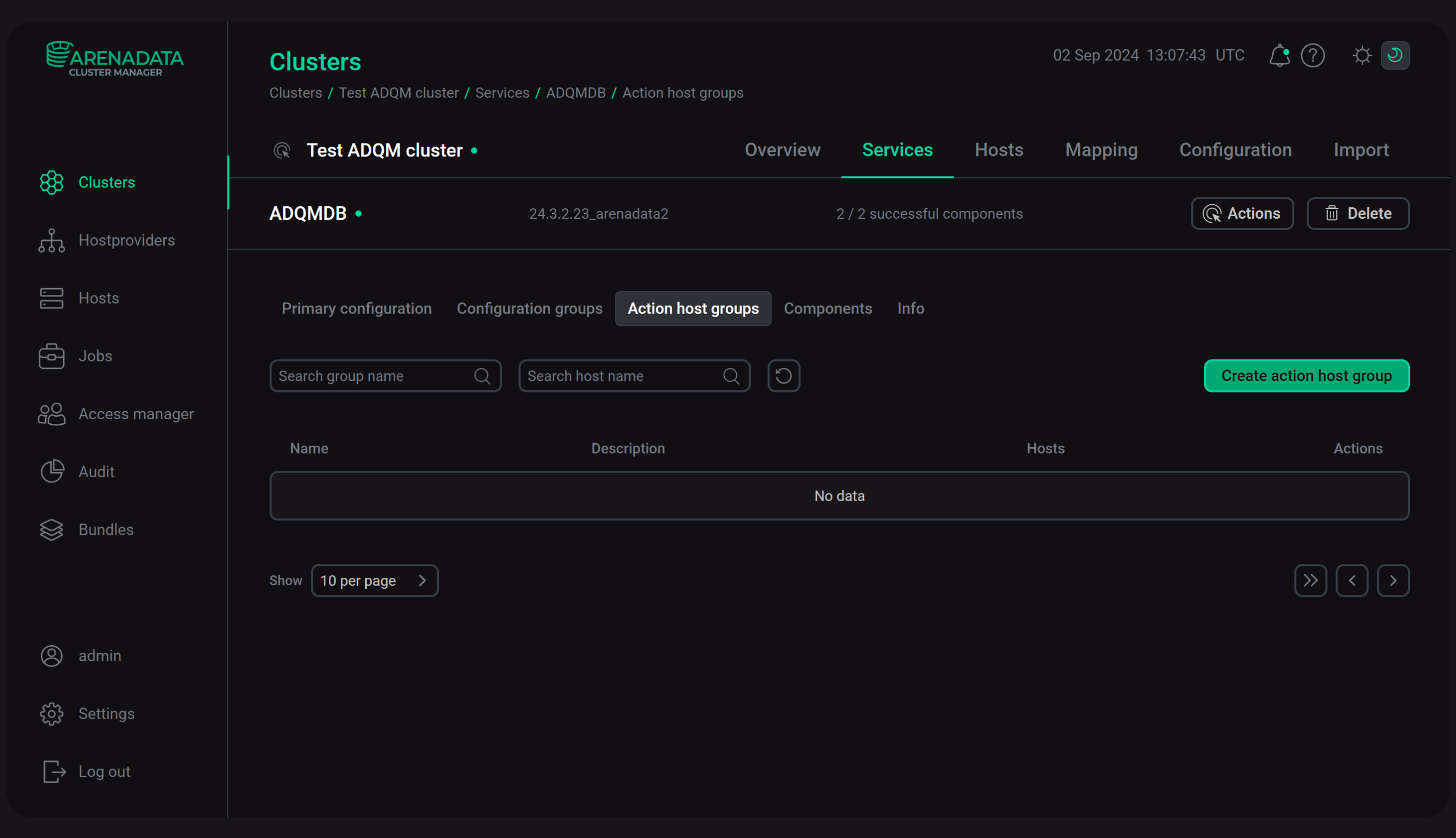
You can manage host groups for actions on the ***Action host groups*** tab, which is located on the configuration page of the corresponding object in the ADCM interface:

- for a cluster: ***Clusters*** → ***<cluster>*** → ***Configuration***;
- for a service: ***Clusters*** → ***<cluster>*** → ***Services*** → ***<service>***;
- for a component: ***Clusters*** → ***<cluster>*** → ***Services*** → ***<service>*** → ***Components*** → ***<component>***.

This article describes how to create and use host groups on the example of an ADQM cluster's ADQMDB service.

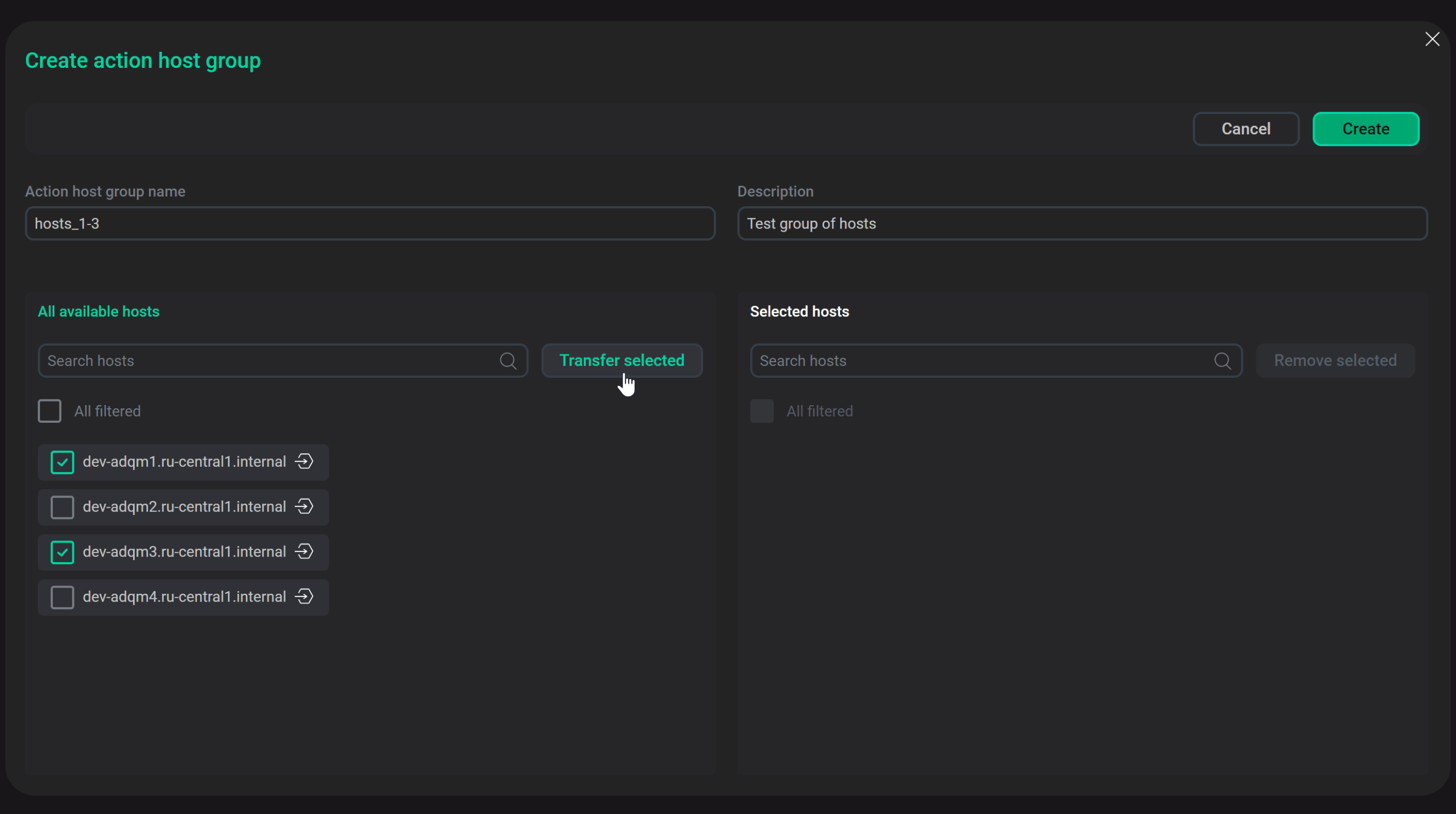
Create a host group

1. On the page of the service installed in the selected cluster, open the ***Action host groups*** tab.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Action host groups
02 Sep 2024 13:07:43 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
24.3.2.23_arenadata2
2 / 2 successful components
Actions
Delete
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
Primary configuration
Configuration groups
Action host groups
Components
Info
Search group name
Search host name
Create action host group
Name
Description
Hosts
Actions
No data
Show
10 per page
admin
Settings
Log out

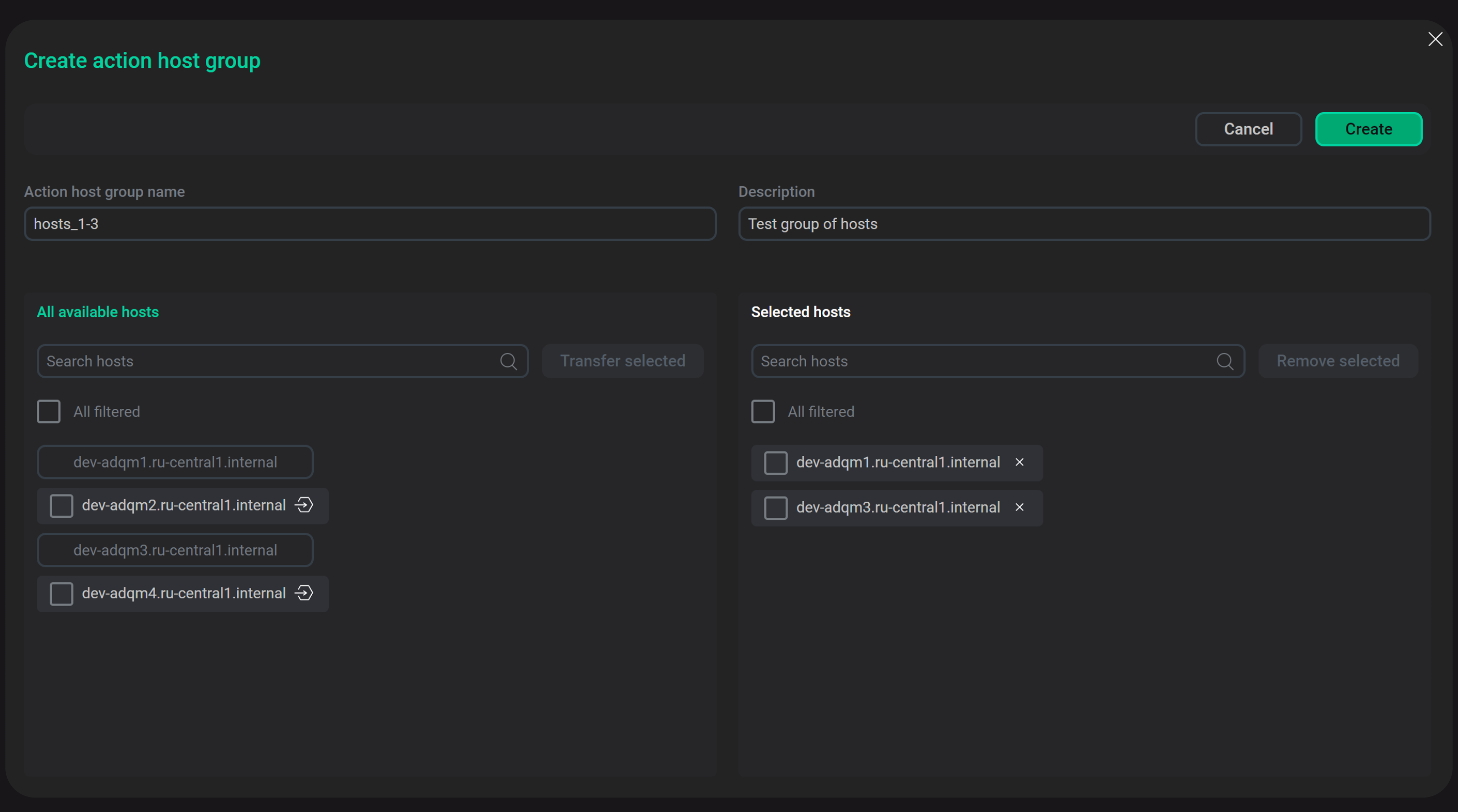
Service page for managing host groups

2. Click ***Create action host group***.
3. In the window that opens, enter a name of a new host group in the ***Action host group name*** field and the group description in the ***Description*** field (optional).
4. Select the desired hosts in the ***All available hosts*** list, and click ***Transfer selected*** to add these hosts to the group.


Create action host group
Cancel
Create
Action host group name
hosts_1-3
Description
Test group of hosts
All available hosts
Search hosts
Transfer selected
All filtered
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Selected hosts
Search hosts
Remove selected
All filtered

Adding multiple hosts to a group

You can also add and remove hosts one by one using the icons ⇥ and × in the ***All available hosts*** and ***Selected hosts*** lists, respectively.


Create action host group
Cancel
Create
Action host group name
hosts_1-3
Description
Test group of hosts
All available hosts
Search hosts
Transfer selected
All filtered
dev-adqm1.ru-central1.internal
dev-adqm2.ru-central1.internal
dev-adqm3.ru-central1.internal
dev-adqm4.ru-central1.internal
Selected hosts
Search hosts
Remove selected
All filtered
dev-adqm1.ru-central1.internal
dev-adqm3.ru-central1.internal

Hosts added to the group

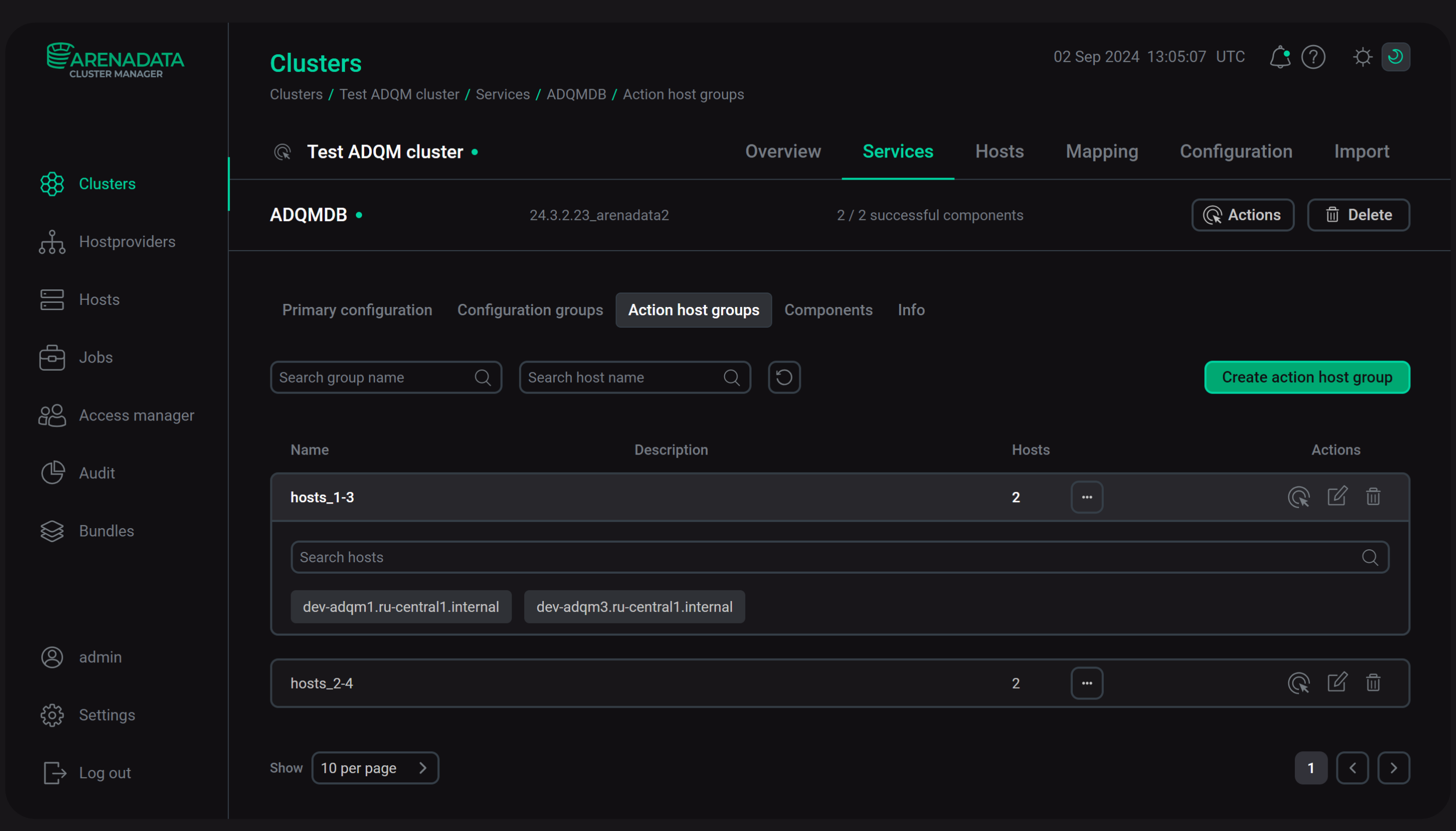
When adding hosts to groups for actions, consider the following:

- The same host can be added to different groups of the same object and different objects within the same cluster.
- A host group for actions at the component level can include only hosts to which the component is mapped. The same is true for services.

5. Click ***Create*** to create a group from the selected hosts.

Manage host groups

Created host groups are listed on the ***Actions host groups*** tab. Above this list, there are filters that you can use to search for the desired groups by name or find all groups that include a host with the specified name. The icon ⟳ resets all filters.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Action host groups
02 Sep 2024 13:05:07 UTC
Test ADQM cluster
Overview
Services
Hosts
Mapping
Configuration
Import
ADQMDB
24.3.2.23_arenadata2
2 / 2 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
Search group name
Search host name
Create action host group
Name
Description
Hosts
Actions
hosts_1-3
2
Search hosts
dev-adqm1.ru-central1.internal
dev-adqm3.ru-central1.internal
hosts_2-4
2
Show
10 per page
1

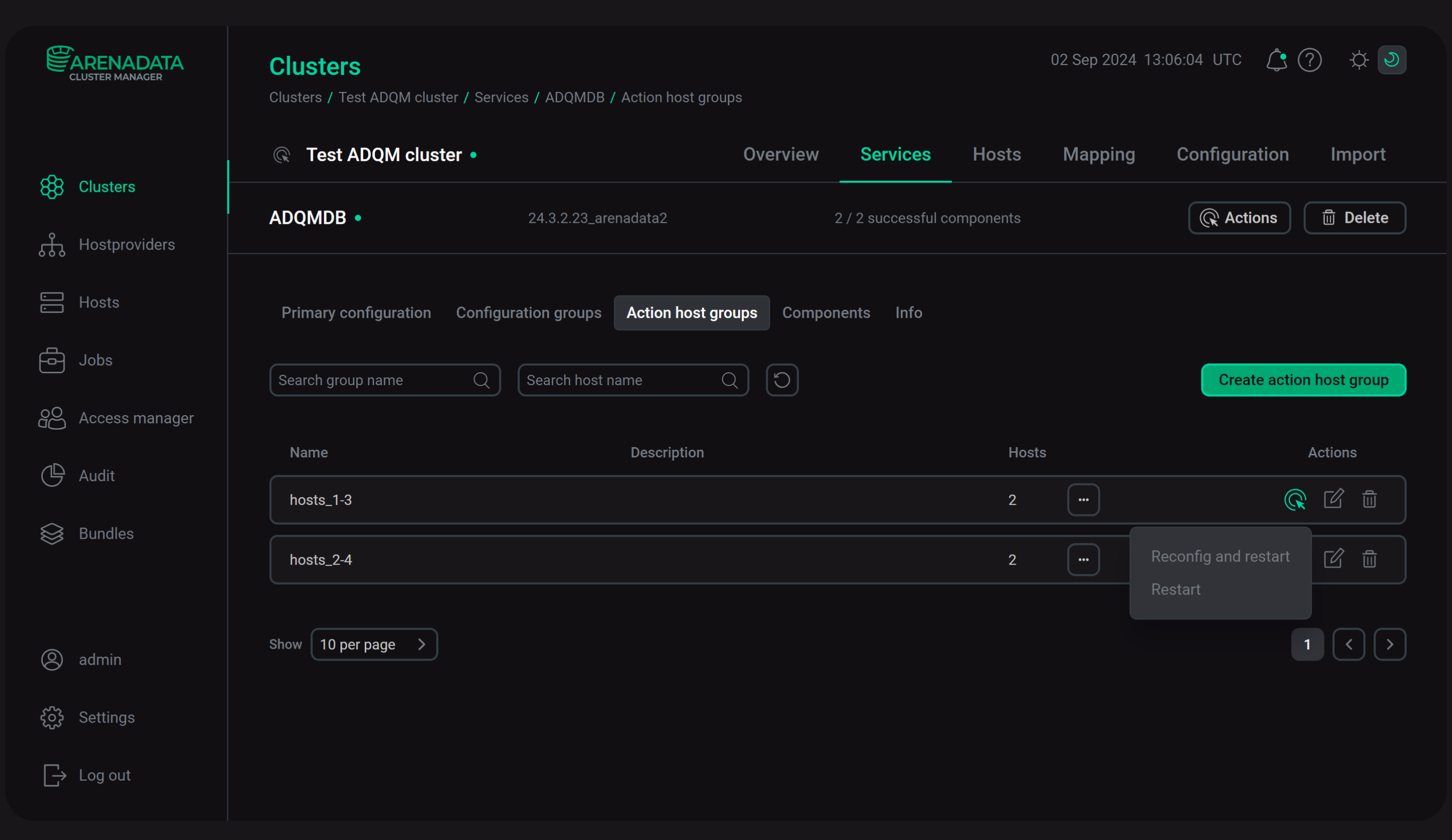
List of host groups

To manage host groups, use the icons:

- ⋯ — expands/hides the list of hosts under a group row in the list;
- ✎ — opens the ***Edit action host group*** window, where you can change the list of hosts included in a group;
- 🗑 — deletes a host group.

Run an action for a host group

In a host group row, click the icon ⟳ — in the list that opens, select an action to perform on the hosts of this group.


ARENADATA CLUSTER MANAGER
Clusters
Clusters / Test ADQM cluster / Services / ADQMDB / Action host groups
02 Sep 2024 13:06:04 UTC
Test ADQM cluster
ADQMDB
24.3.2.23_arenadata2
2 / 2 successful components
Action host groups
Create action host group
Name
Description
Hosts
Actions
hosts_1-3
2
hosts_2-4
2
Reconfig and restart
Restart
Show
10 per page
1

Actions allowed to be performed on hosts in the group

A set of actions available for execution on hosts depends on the object for which the host group is created — product, object type (cluster, service, or component), state of this object. For example, the image above shows the list of actions for the ADQMDB service of the ADQM cluster in the `installed` state. If a product or object does not support performing any actions on groups of hosts, the action list displays `No results found`.

Overview of the Jobs tab in the ADCM UI

Adelya Zvyagintseva

Contents

To Table of Contents

List of jobs

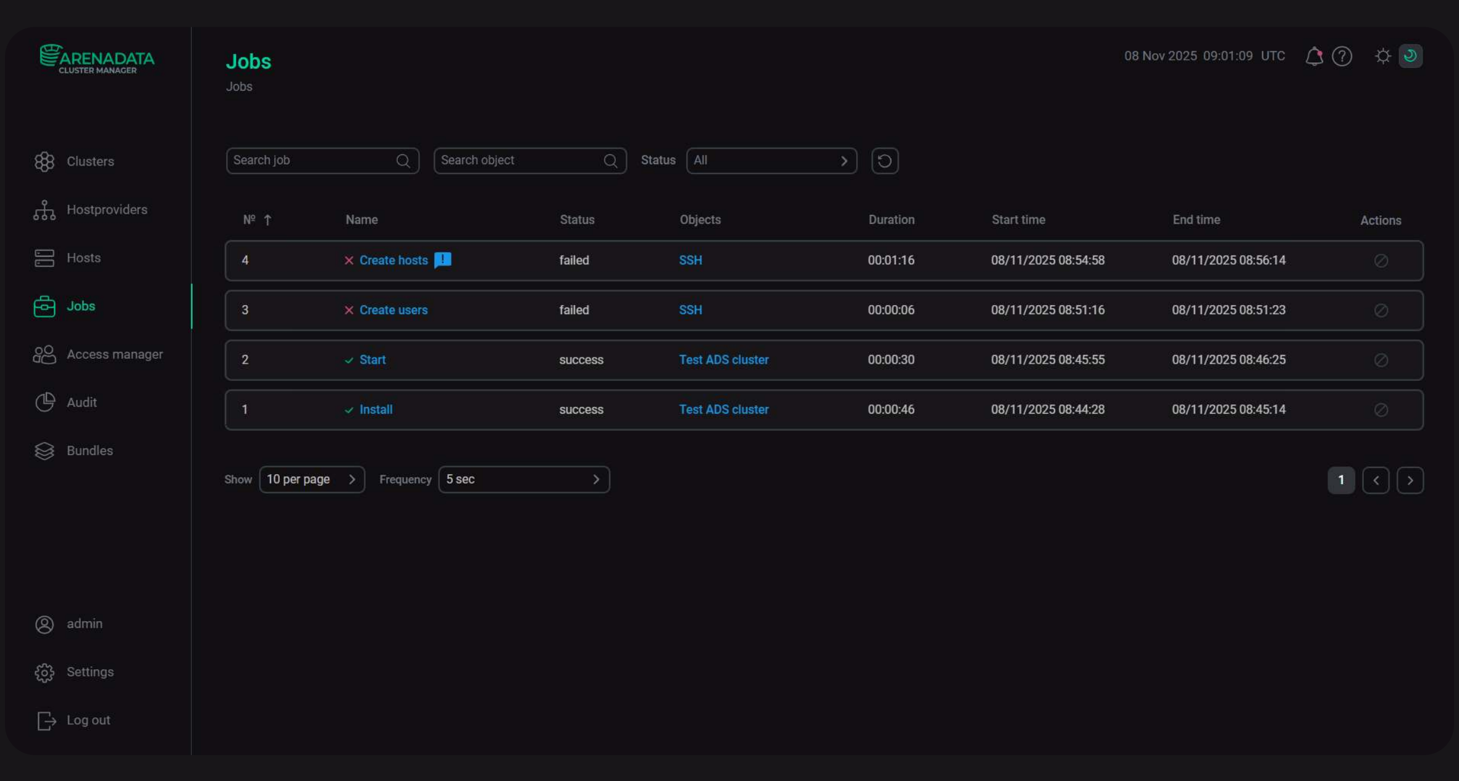
Operations with a job

Job details

The ***Jobs*** tab in the ADCM web interface allows you to view additional information on the execution process and results of actions for the ADCM objects.

List of jobs

On the ***Jobs*** tab, you can see the list of jobs related to actions in tabular form which contains information described below.



Status: All

№	Name	Status	Objects	Duration	Start time	End time	Actions
4	Create hosts	failed	SSH	00:01:16	08/11/2025 08:54:58	08/11/2025 08:56:14	
3	Create users	failed	SSH	00:00:06	08/11/2025 08:51:16	08/11/2025 08:51:23	
2	Start	success	Test ADS cluster	00:00:30	08/11/2025 08:45:55	08/11/2025 08:46:25	
1	Install	success	Test ADS cluster	00:00:46	08/11/2025 08:44:28	08/11/2025 08:45:14	

Show: 10 per page
Frequency: 5 sec

The Jobs tab

Fields of the Jobs tab

Field	Description
№	Order number of the job
Name	Name of the job corresponding to the chosen action and containing a link to the job page
Status	Status of the job. Possible values: ▪ `created` — an action was created, but is not running yet; ▪ `success` — an action was successfully executed; ▪ `failed` — an action was executed with an error; ▪ `running` — an action is being executed at the current time; ▪ `aborted` — the job corresponding to the chosen action was aborted by user; ▪ `broken` — a programming error occurred during the action execution.
Objects	Name of the ADCM object that contains a link to the object page
Duration	Duration of the job execution in the `HH:mm:ss` format
Start time	Job execution start time in the `dd/MM/yyyy HH:mm:ss` format
End time	Job execution end time in the `dd/MM/yyyy HH:mm:ss` format
Actions	The ⊘ icon that stops the job execution. If a job related to an action cannot be stopped (`"isTerminatable": false`), the ⊘ icon will be displayed in the column

To sort the job list:

- in ascending order of the job order numbers, click the ↑ icon in the ***№*** column;
- in descending order of the job order numbers, click the ↓ icon in the ***№*** column.

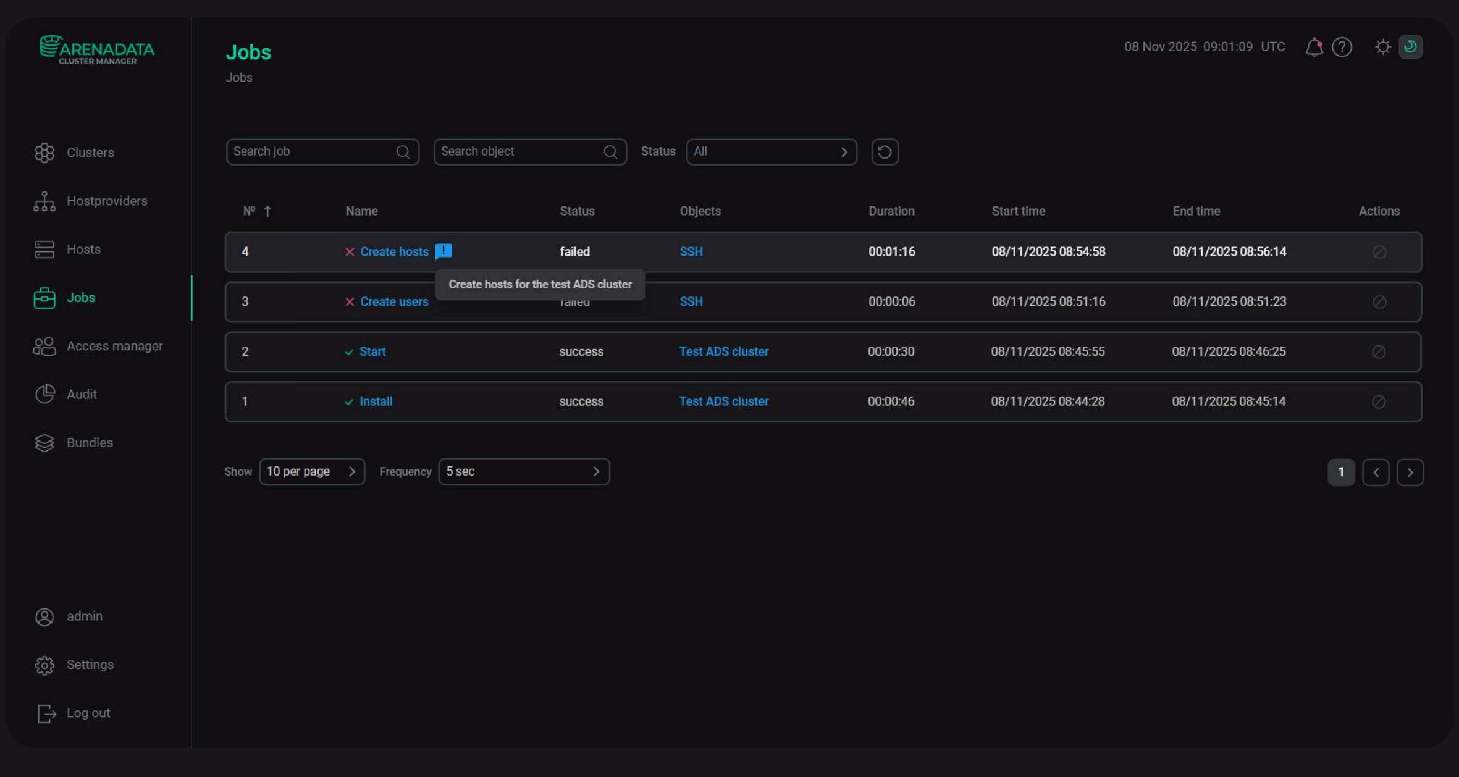
To search for a job in the list:

- by action name — enter its name in the ***Search job*** field;
- by object name — enter its name in the ***Search object*** field.

To filter jobs by status, select the required value from the drop-down list in the field next to ***Status***.

To reset your filters, click the ↻ icon.

To view a comment added in the dialog box when you run an action, hover over the ! icon.



Status: All

№	Name	Status	Objects	Duration	Start time	End time	Actions
4	Create hosts	failed	SSH	00:01:16	08/11/2025 08:54:58	08/11/2025 08:56:14	
3	Create users	failed	SSH	00:00:06	08/11/2025 08:51:16	08/11/2025 08:51:23	
2	Start	success	Test ADS cluster	00:00:30	08/11/2025 08:45:55	08/11/2025 08:46:25	
1	Install	success	Test ADS cluster	00:00:46	08/11/2025 08:44:28	08/11/2025 08:45:14	

Create hosts for the test ADS cluster

Show: 10 per page
Frequency: 5 sec

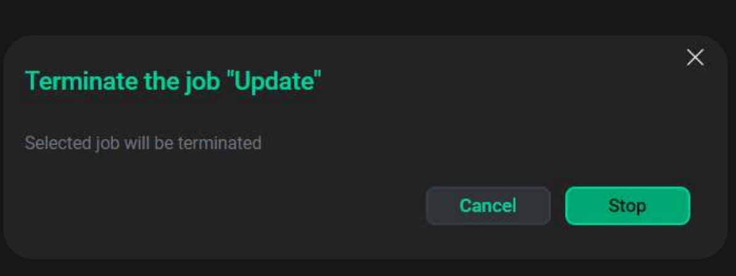
The Jobs tab with the expanded comment on the action

Operations with a job

ADCM allows you to stop the execution of jobs related to actions.

To stop the job execution, follow these steps:

1. Open the ***Jobs*** tab.
2. On the page that opens, click the ⊘ icon in the row of the job the execution of which should be stopped.
3. In the dialog box, confirm your action by clicking ***Stop***. As a result, the `aborted` status is automatically assigned to the job.


Terminate the job "Update"

Selected job will be terminated

The operation confirmation dialog box

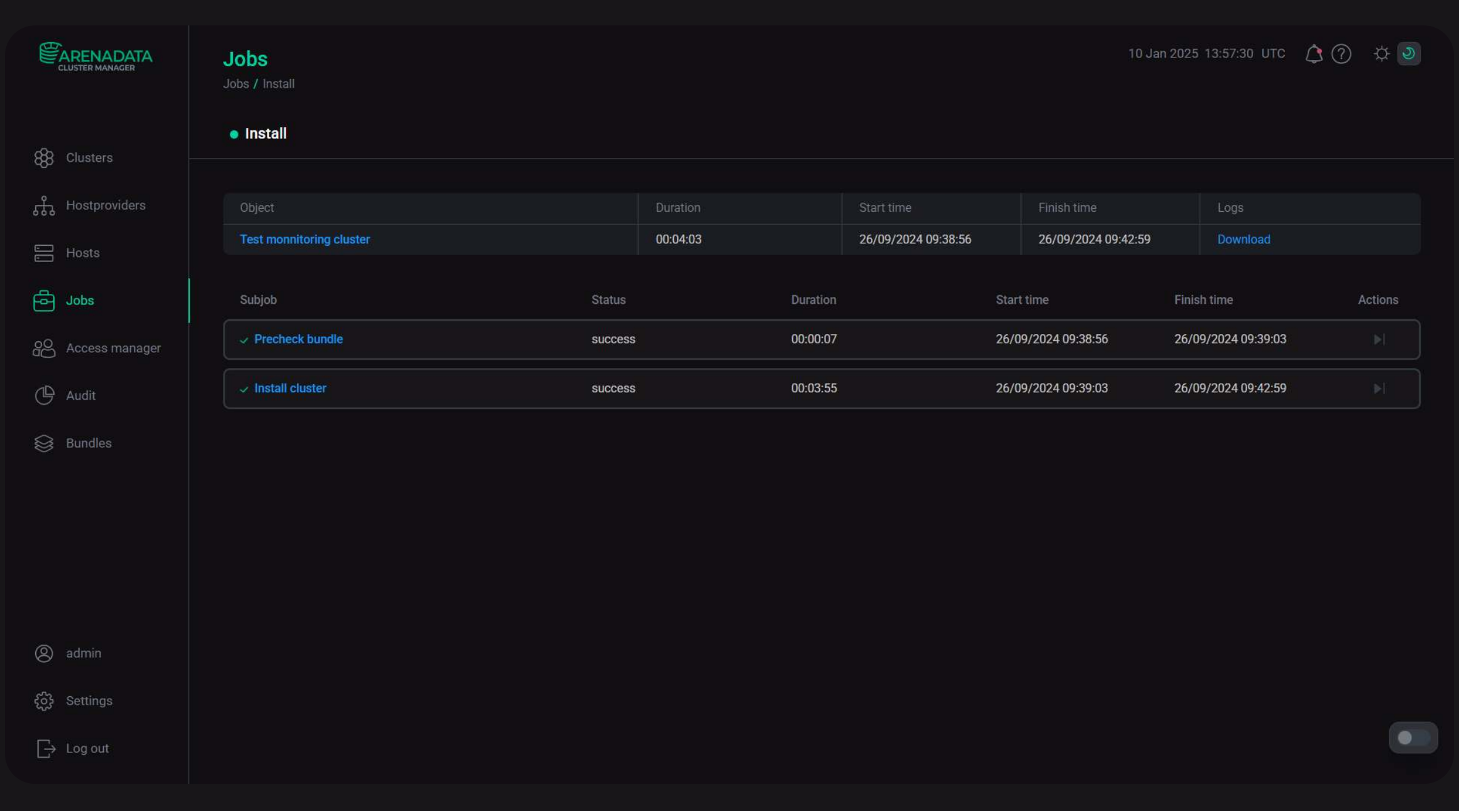
NOTE

You can stop only those tasks for which such possibility is provided (`"isTerminatable":true`).

Job details

To get detailed information about the job, click on the job name in the ***Name*** column of the table located on the ***Jobs*** tab. This page includes the following tables, each of which is described in detail below:

- Table with the job details
- Table with a list of subjobs of which the job consists



Object	Duration	Start time	Finish time	Logs
Test monnitoring cluster	00:04:03	26/09/2024 09:38:56	26/09/2024 09:42:59	Download

Subjob	Status	Duration	Start time	Finish time	Actions
Precheck bundle	success	00:00:07	26/09/2024 09:38:56	26/09/2024 09:39:03	
Install cluster	success	00:03:55	26/09/2024 09:39:03	26/09/2024 09:42:59	

The Install job page

Job details

Field	Description
Object	Name of the ADCM object that contains a link to the object page
Duration	Duration of the job execution in the `HH:mm:ss` format
Start time	Job execution start time in the `dd/MM/yyyy HH:mm:ss` format
Finish time	Job execution finish time in the `dd/MM/yyyy HH:mm:ss` format
Logs	Link to the archive that includes the following files: ▪ *ansible.cfg* — the Ansible config file; ▪ *ansible-stdout.txt* and *ansible-stderr.txt* — log files containing information about the job execution; ▪ *config.json* — the ADCM config file; ▪ *inventory.json* — inventory file of a cluster or infrastructure bundle.

Subjobs

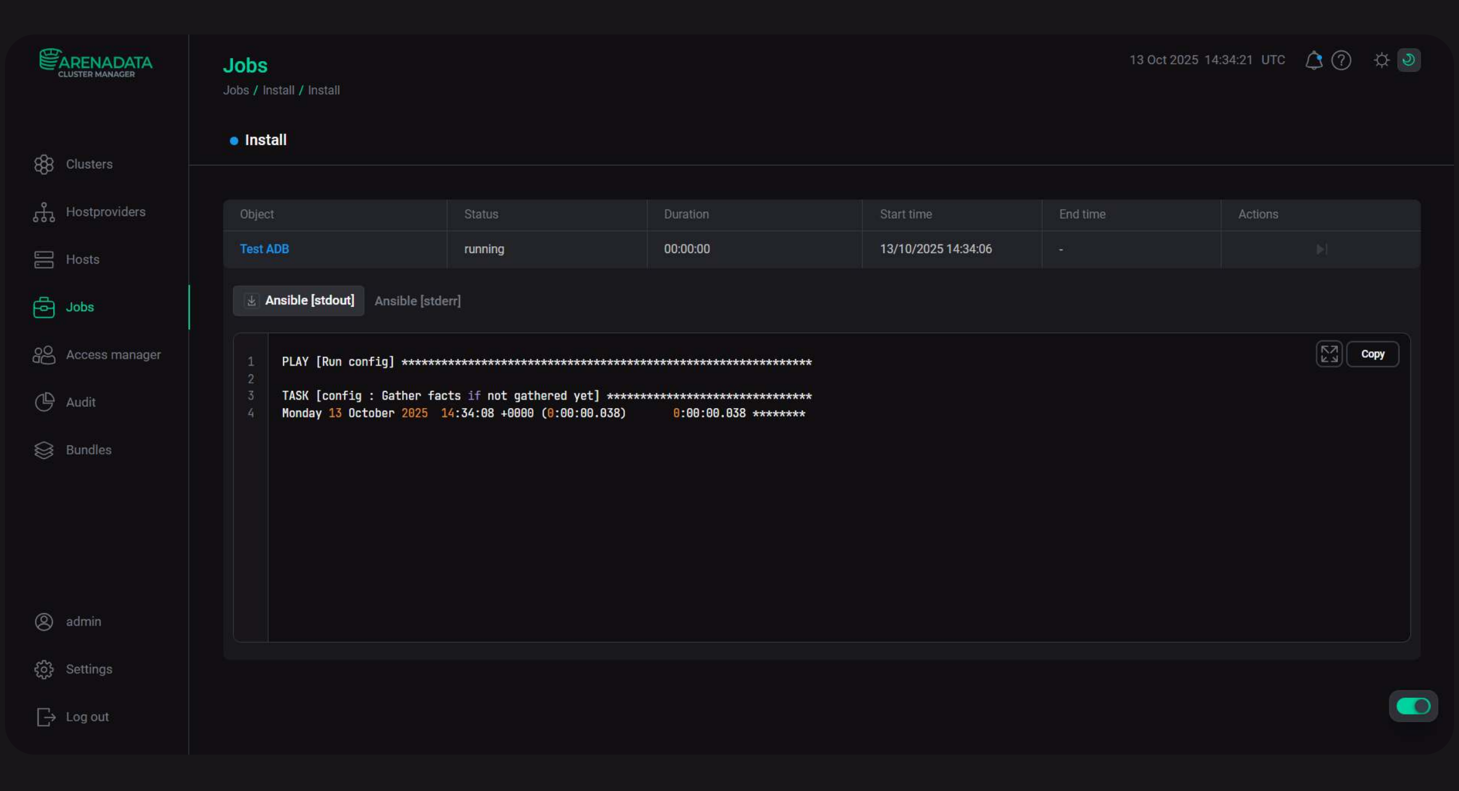
Field	Description
Subjob	Name of the subjob containing a link to the subjob page
Status	Status of the subjob. Possible values are identical to those described in the table Fields of the Jobs tab
Duration	Duration of the subjob execution in the `HH:mm:ss` format
Start time	Subjob execution start time in the `dd/MM/yyyy HH:mm:ss` format
End time	Subjob execution end time in the `dd/MM/yyyy HH:mm:ss` format
Actions	The ▶| icon that stops the execution of the current subjob and transfers the action management to the next subjob. If a stop of the subjob execution is not available (`"isTerminatable": false`), the ▶| icon will be displayed in the column

To stop the subjob execution, click the ▶| icon in the row of the subjob the execution of which should be stopped.

NOTE

The ▶| icon is also available on the subjob page.

To get detailed information on the subjob, click on the subjob name in the ***Subjob*** column of the table.



Object	Status	Duration	Start time	End time	Actions
Test ADB	running	00:00:00	13/10/2025 14:34:06	-	

```
PLAY [Run config] ****************************************************************

TASK [config : Gather facts if not gathered yet] ******************************
Monday 13 October 2025  14:34:08 +0000 (0:00:00.038)       0:00:00.038 ********
```

The Install ADB subjob page

This page displays the same information on the subjob provided in tabular form and described in the Subjobs table.

Also, this page includes the following tabs:

- ***Ansible [stdout]*** — the standard output (stdout) of the command to run the Ansible playbook;
- ***Ansible [stderr]*** — error output (stderr) of the command to run the Ansible playbook;
- ***Ansible [check]*** (optional) — log in a structured format for specific Ansible tasks.

To copy the logs, select the necessary tab, hover over the log area and click ***Copy*** at the top-right of the log area.

To download the *ansible-stdout.txt*, *ansible-stderr.txt* log files, and the *<subjob identifier>-ansible.json* file, select the corresponding tab and click the ⤓ icon.

The toggle in the bottom right corner of the page allows you to disable automatic scrolling to the bottom of the log while the subjob is in progress. By default, the toggle is switched on.

To expand the log window to full screen, hover over the log area and click the ⛶ icon at the top-right of the log area. To exit the full-screen mode, click the ✕ icon.

NOTE

The full-screen viewing mode contains the same tabs and icons that are available in windowed mode.

Configuration parameters in the ADCM UI

Adelya Zvyagintseva

Contents

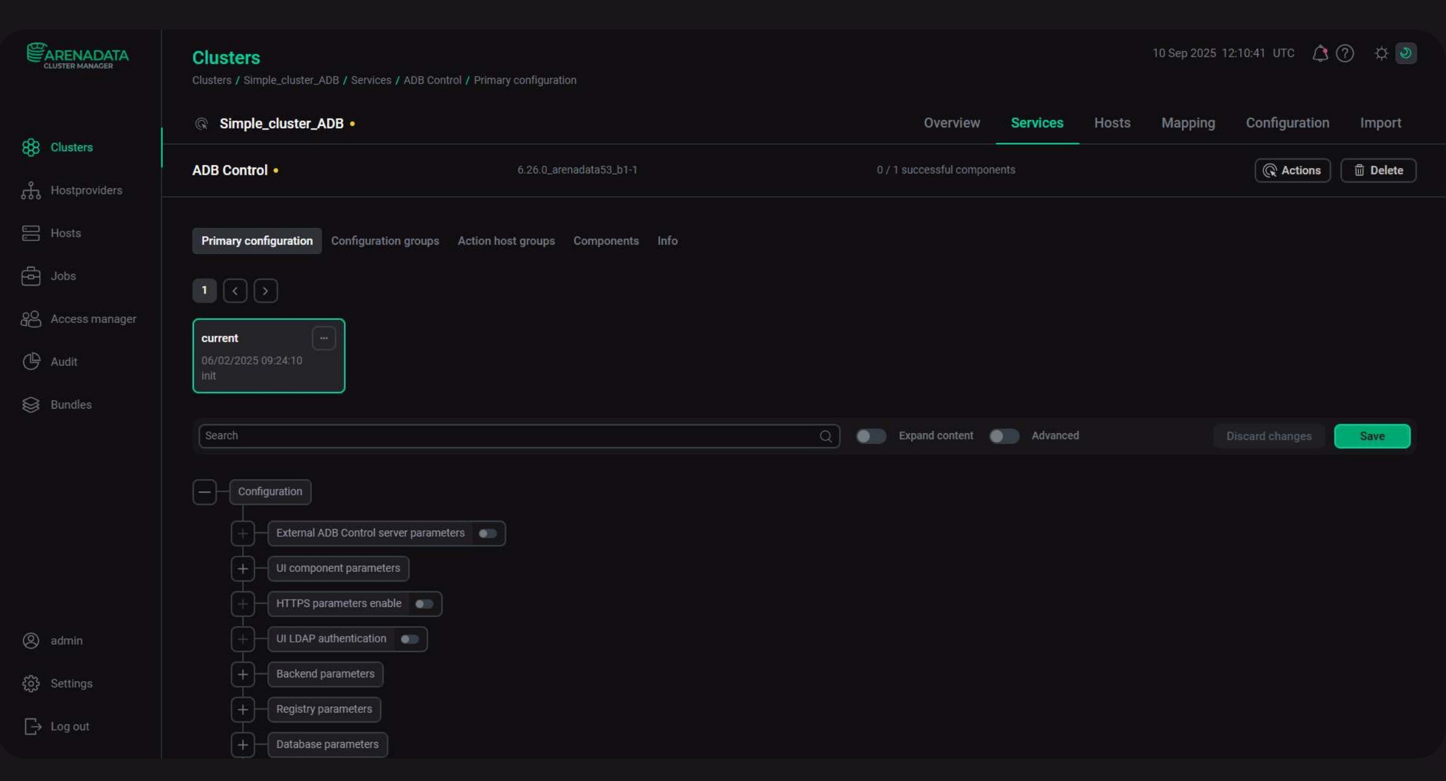
To Table of Contents

- Manage configuration parameters
- Parameter types
  - Drop-down list
  - Single line text field
  - Sensitive data input field
  - Multi-line text input field
  - Numeric input field
  - Arbitrary configuration parameter
  - Array of composite objects or strings
- Compare configurations

This article explains how to work with the object configuration parameters on the following tabs located on all main web interface pages:

- ***Primary configuration*** — contains parameters with which you can configure the ADCM objects.
- ***Ansible settings*** — contains settings used in creating the *ansible.cfg* file on every launch of an action on an object in a cluster.
- ***Configuration group*** — contains configuration parameters of a config group.

Below is an example of how to fill out the ***Primary configuration*** tab for the ADB Control service. The presented option of filling in configuration parameters is correct for other objects.



ARENADATA
CLUSTER MANAGER
Clusters
Clusters / Simple_cluster_ADB / Services / ADB Control / Primary configuration
10 Sep 2025 12:10:41 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Simple_cluster_ADB
Overview
Services
Hosts
Mapping
Configuration
Import
ADB Control
6.26.0_arenadata53_b1-1
0 / 1 successful components
Actions
Delete
Primary configuration
Configuration groups
Action host groups
Components
Info
1
current
06/02/2025 09:24:10
init
Search
Expand content
Advanced
Discard changes
Save
Configuration
External ADB Control server parameters
UI component parameters
HTTPS parameters enable
UI LDAP authentication
Backend parameters
Registry parameters
Database parameters

The Primary configuration tab for the ADB Control service

The ***Primary configuration*** tab contains the following switches:

- ***Advanced*** — makes visible some of the fine-tuning configuration parameters.
- ***Expand content*** — expands and collapses the configuration parameter tree.

After you fill in configuration parameters and save the configuration by clicking ***Save***, a new configuration version appears including service information about configuration. If you have a previously saved configuration, you can view it and change configuration parameters by clicking on a tab with the appropriate name.

Manage configuration parameters

The ADCM web interface provides the following methods for managing object configuration parameters:

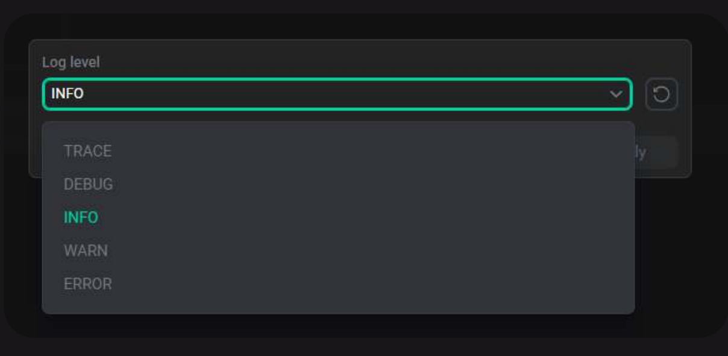
- To reset unsaved configuration parameter changes, click ***Discard changes***.
- To search for a configuration parameter, enter its name in the ***Search*** field.
- To activate or deactivate a section in the configuration tree, click the ○ or ● icon respectively.
- To expand a node in the tree of configuration parameters, for example, the ***Main*** node on the ***Primary configuration*** tab for the ADB service, click the + icon. After you click this icon, it is changed to the − icon, which is used to collapse a node in the tree of configuration parameters.
- To set the default value for a selected configuration parameter, hover over the parameter and click the ⟲ icon.
- To delete one of multiple values added by selecting ***+ Add property***, hover over the parameter value and click the 🗑 icon next to the desired value.
- To clear a parameter value, i.e. set the value to NULL, hover over the parameter value and click the ◇ icon. The icon is displayed in the following situations:
  - Default value was set by the bundle.
  - Parameter value was changed by a user.
- To copy a parameter value, hover over the parameter value and click the ⧉ icon. Note that this icon is unavailable for the sensitive data input fields.
- To get a detailed description of a parameter, hover over the parameter value and click the ? icon. A tooltip with information on filling in the corresponding parameter will appear.

Parameter types

The ADCM web interface supports options for configuration parameter determination described below.

Drop-down list

There are two types of drop-down lists: strict and non-strict. In case of strict drop-down lists, you have to select one option from the list of configuration parameter values. In case of non-strict drop-down lists, you can also specify a value that is not presented in the drop-down list.

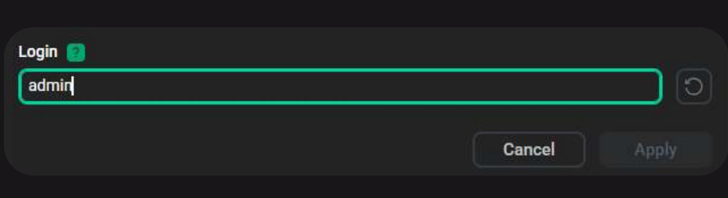


Log level
INFO
TRACE
DEBUG
INFO
WARN
ERROR

The Log level dialog

Single line text field

This field is used to input strings of textual data (letters, numbers, and symbols), such as database name or username.

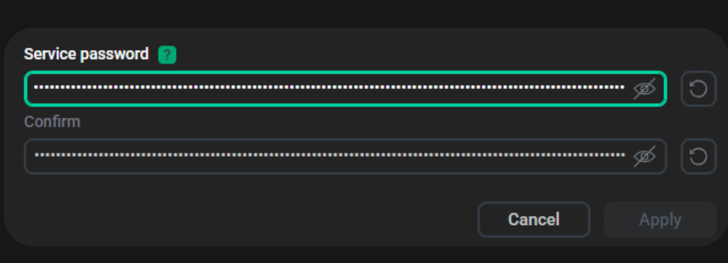


Login
admin
Cancel
Apply

The Login dialog

Sensitive data input field

This field is used to protect sensitive information due to regulatory and privacy requirements. After saving a value, it is available only in encrypted form.



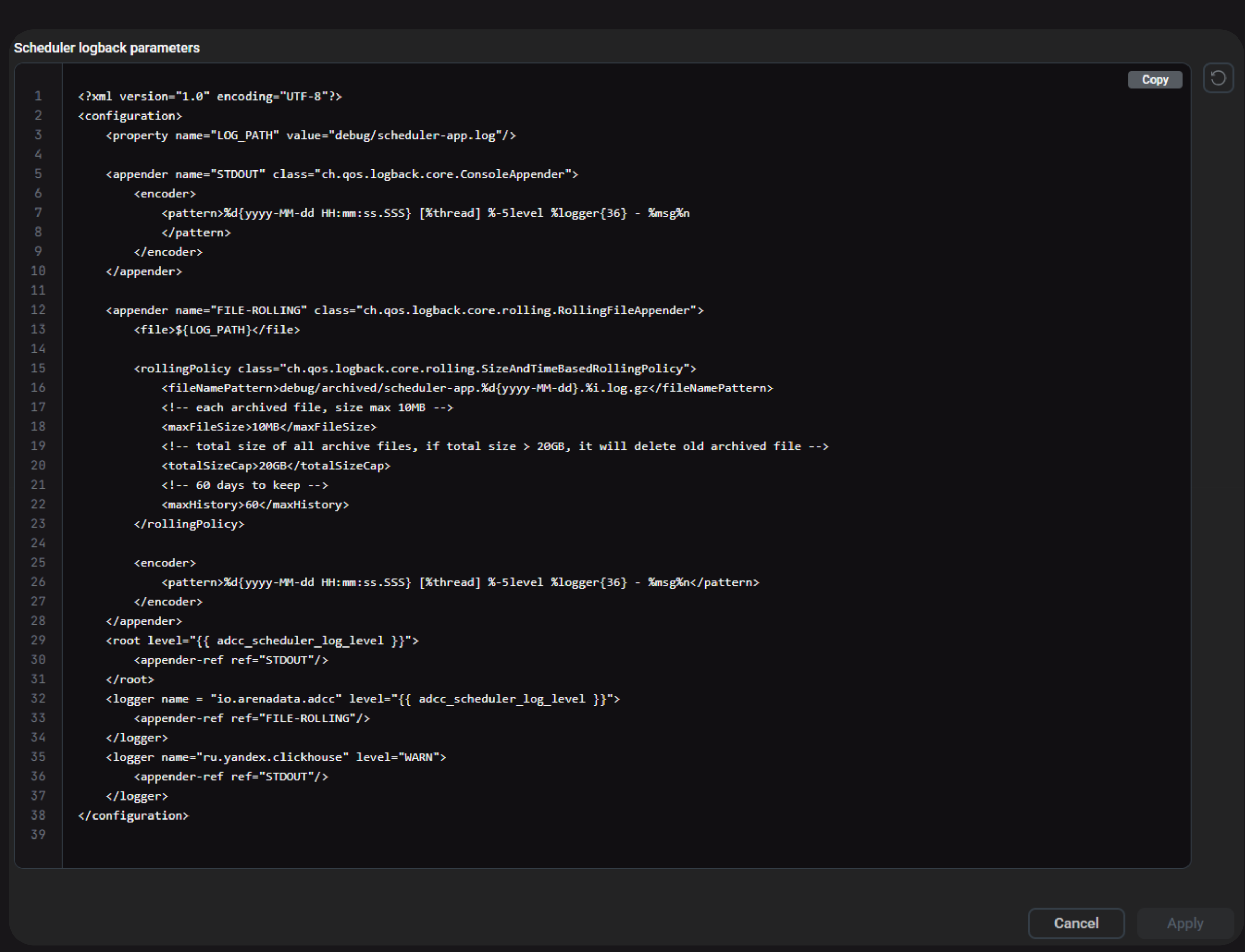
Service password
Confirm
Cancel
Apply

The Service password dialog

To enable the display of entered parameter value, click the ⊘ icon.

Multi-line text input field

This field allows you to enter multi-line text.



Scheduler logback parameters
Copy
Cancel
Apply

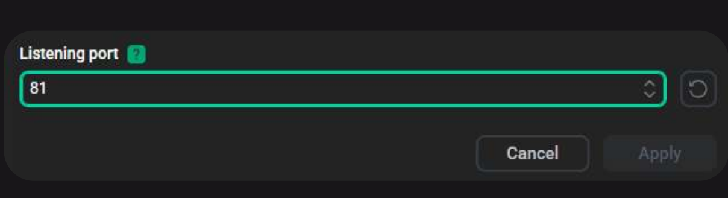
```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <property name="LOG_PATH" value="debug/scheduler-app.log"/>

    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n
            </pattern>
        </encoder>
    </appender>

    <appender name="FILE-ROLLING" class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_PATH}</file>

        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>debug/archived/scheduler-app.%d{yyyy-MM-dd}.%i.log.gz</fileNamePattern>
            <!-- each archived file, size max 10MB -->
            <maxFileSize>10MB</maxFileSize>
            <!-- total size of all archive files, if total size > 20GB, it will delete old archived file -->
            <totalSizeCap>20GB</totalSizeCap>
            <!-- 60 days to keep -->
            <maxHistory>60</maxHistory>
        </rollingPolicy>

        <encoder>
            <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>
    <root level="{{ adcc_scheduler_log_level }}">
        <appender-ref ref="STDOUT"/>
    </root>
    <logger name = "io.arenadata.adcc" level="{{ adcc_scheduler_log_level }}">
        <appender-ref ref="FILE-ROLLING"/>
    </logger>
    <logger name="ru.yandex.clickhouse" level="WARN">
        <appender-ref ref="STDOUT"/>
    </logger>
</configuration>

```

The Scheduler logback parameters dialog

To copy a multi-line text, hover over the working area and click ***Copy*** at the top-right of the working area.

Numeric input field

This field is used to enter only numeric values of configuration parameters (for example, listening port) by using keyboard or increment and decrease values using up and down arrows.

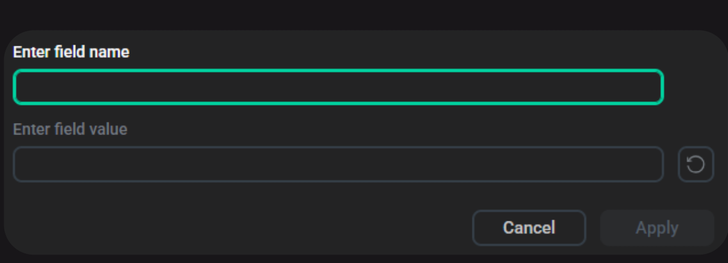


Listening port
81
Cancel
Apply

The Listening port dialog

Arbitrary configuration parameter

The ***+ Add property*** nested element is used to specify a configuration parameter with arbitrary name and value. Use this element in case of absence of the required configuration parameter among the parameters predefined in the ADCM web interface.



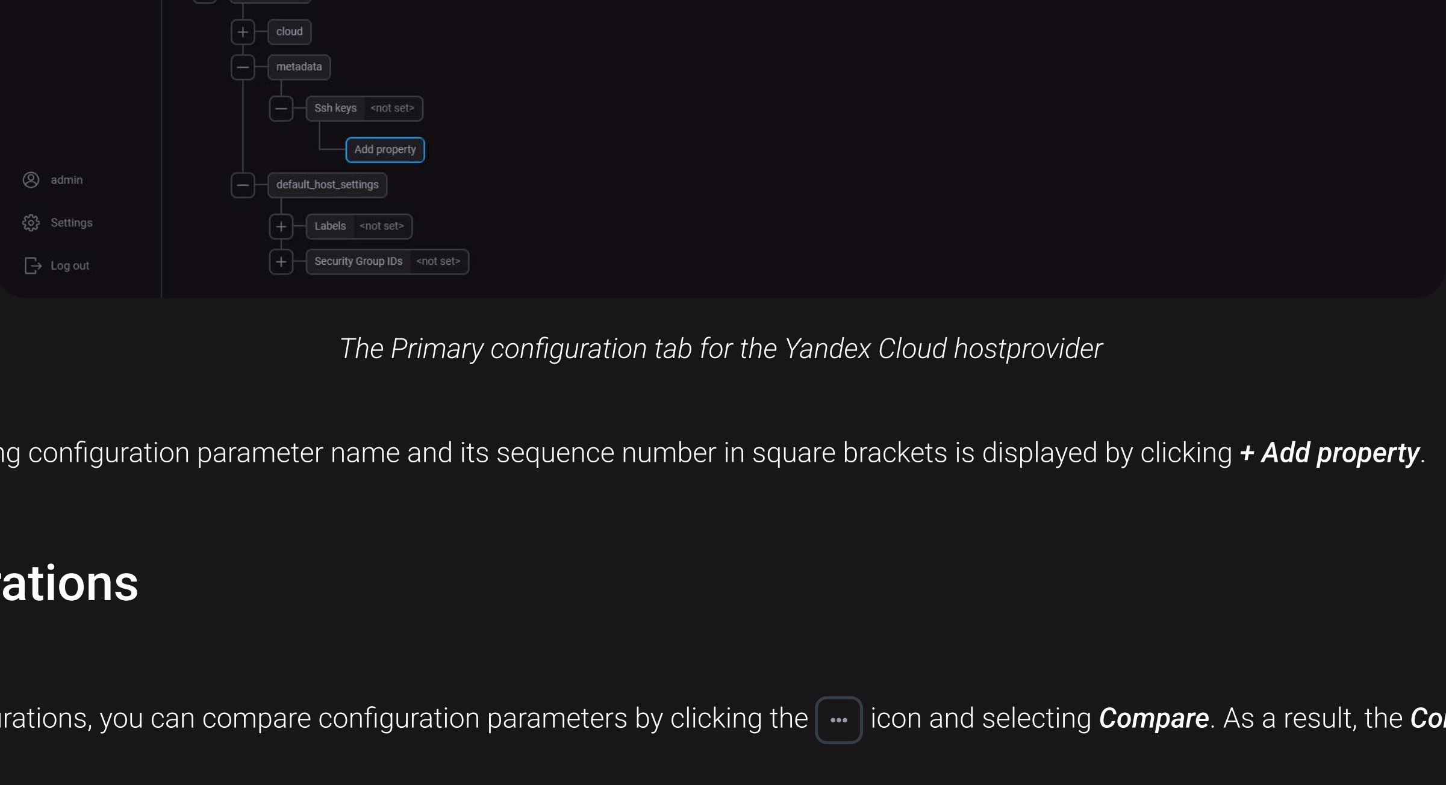
Enter field name
Enter field value
Cancel
Apply

The dialog that appears by clicking Add property

To add a new parameter, enter the parameter name in the ***Enter field name*** field and its value in the ***Enter field value*** field and click ***Apply*** to apply changes.

Array of composite objects or strings

In addition to specifying a configuration parameter with arbitrary name and value, the ***+ Add property*** nested element also allows you to add a new element to the list. The element can be either an element of the string type, such as an array of SSH keys, or a more complex composite object.



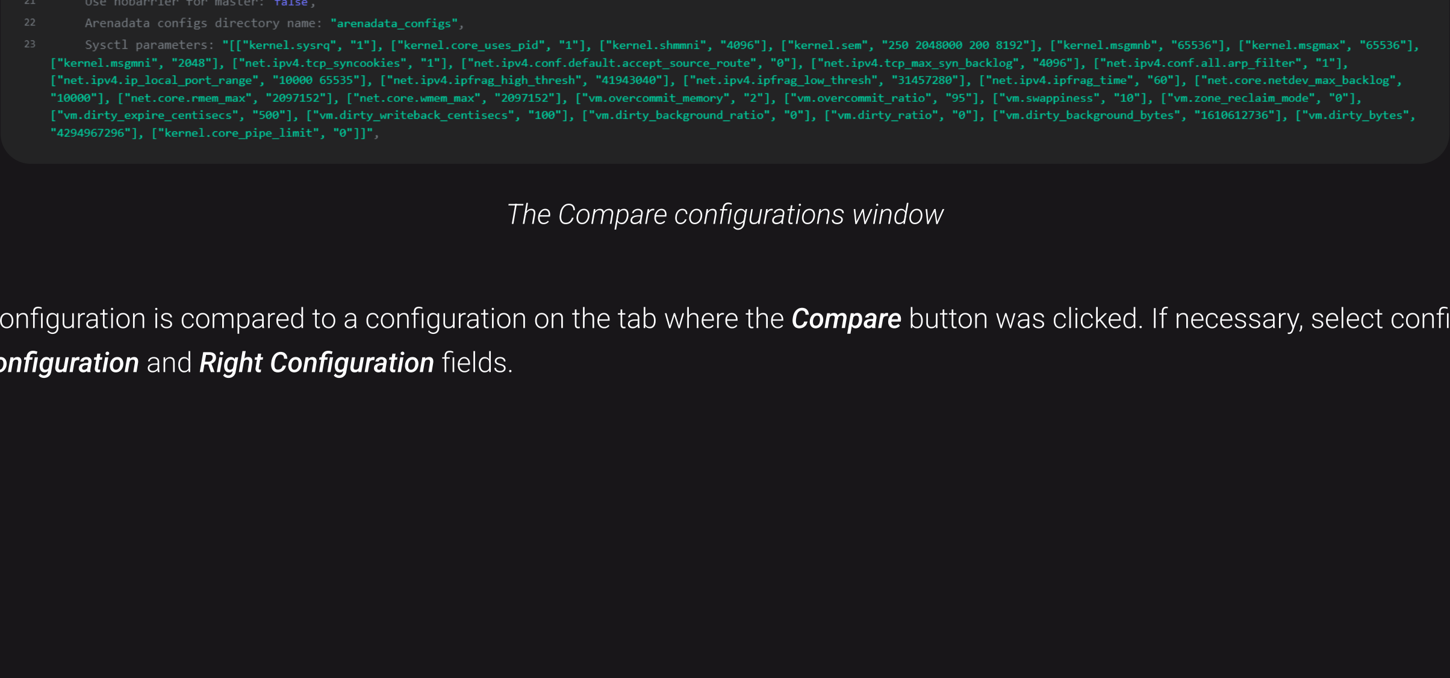
ARENADATA
CLUSTER MANAGER
Hostproviders
Hostproviders / Yandex Cloud / Primary configuration
15 Sep 2025 10:41:07 UTC
Clusters
Hostproviders
Hosts
Jobs
Access manager
Audit
Bundles
admin
Settings
Log out
Yandex Cloud
Version 3.9-1
0 hosts
Actions
Delete
Primary configuration
Configuration groups
1
current
15/09/2025 10:40:02
15/09/2025 10:33:33
init
Search
Expand content
Advanced
Discard changes
Save
Configuration
cloud
metadata
Ssh keys
<not set>
Add property
default_host_settings
Labels
<not set>
Security Group IDs
<not set>

The Primary configuration tab for the Yandex Cloud hostprovider

The field with the name containing configuration parameter name and its sequence number in square brackets is displayed by clicking ***+ Add property***.

Compare configurations

If there are several saved configurations, you can compare configuration parameters by clicking the … icon and selecting ***Compare***. As a result, the ***Compare configurations*** window is displayed.



Compare configurations
Left Configuration
27/01/2025 11:28:10 - ansible update
Right Configuration
27/01/2025 08:49:30 - init

```
Main: {
  Name of default database: "adb",
  Use segment mirroring: true,
  ADB MADlib Extension: false,
  ADB PostGIS Extension: false,
  ADB GPperfmon Extension: true,
  ADB Auto Core Dump Service: false,
  Diskquota Extension: false,
  PL/Python3u Extension: false
},
Advanced: {
-  Number of segments per host: 4,
+  Number of segments per host: null,
  Segment host data directories: {
    /data1: ""
  },
  Use nobarrier for segments: false,
  Masters host data directory: "/data1",
  Mount master storage device for data: false,
  Master server storage device: null,
  Use nobarrier for master: false,
  Arenadata configs directory name: "arenadata_configs",
  Sysctl parameters: "[["kernel.sysrq", "1"], ["kernel.core_uses_pid", "1"], ["kernel.shmmni", "4096"], ["kernel.sem", "250 2048000 200 8192"], ["kernel.msgmnb", "65536"], ["kernel.msgmax", "65536"], ["kernel.msgmni", "2048"], ["net.ipv4.tcp_syncookies", "1"], ["net.ipv4.conf.default.accept_source_route", "0"], ["net.ipv4.tcp_max_syn_backlog", "4096"], ["net.ipv4.conf.all.arp_filter", "1"], ["net.ipv4.ip_local_port_range", "10000 65535"], ["net.ipv4.ipfrag_high_thresh", "41943040"], ["net.ipv4.ipfrag_low_thresh", "31457280"], ["net.ipv4.ipfrag_time", "60"], ["net.core.netdev_max_backlog", "10000"], ["net.core.rmem_max", "2097152"], ["net.core.wmem_max", "2097152"], ["vm.overcommit_memory", "2"], ["vm.overcommit_ratio", "95"], ["vm.swappiness", "10"], ["vm.zone_reclaim_mode", "0"], ["vm.dirty_expire_centisecs", "500"], ["vm.dirty_writeback_centisecs", "100"], ["vm.dirty_background_ratio", "0"], ["vm.dirty_ratio", "0"], ["vm.dirty_background_bytes", "1610612736"], ["vm.dirty_bytes", "4294967296"], ["kernel.core_pipe_limit", "0"]]",
```

The Compare configurations window

By default, a currently selected configuration is compared to a configuration on the tab where the ***Compare*** button was clicked. If necessary, select configurations for comparison from the drop-down lists in the ***Left Configuration*** and ***Right Configuration*** fields.

Ansible ecosystem

Mikhail Serov

To Table of Contents

If the bundle contains the *ansible.cfg* file, use that file when you prepare the action environment. In other case, the default configuration file is used.

Ansible modules and plugins

Mikhail Serov

Contents

To Table of Contents

- adcm_multi_state_set
  - Examples
  - Return values
- adcm_multi_state_unset
  - Examples
  - Return values
- adcm_state
  - Examples
  - Return values
- adcm_check
  - Examples
- adcm_custom_log
  - Examples
- adcm_config
  - Examples
  - Return values
- adcm_manage_revision
  - Examples
- adcm_add_host
  - Examples
  - Return values
- adcm_delete_host
  - Examples
  - Return values
- adcm_add_host_to_cluster
  - Examples
  - Return values
- adcm_remove_host_from_cluster
  - Examples
  - Return values
- adcm_hc
  - Examples
  - Return values
- adcm_delete_service
  - Examples
  - Return values
- adcm_change_maintenance_mode
  - Examples
  - Return values
- adcm_change_flag
  - Examples
  - Return values

You can use additional modules and plugins to change ADCM internal metadata during the execution of playbooks. All modules and plugins are described in detail in the sections below.

adcm_multi_state_set

This is a special ADCM-only module that is useful to add new multistates for various objects. Multistates are supported for clusters, services, hosts, components, and hostproviders.

Using `service_name` gives you the ability to switch between services.

Using `component_name` gives you the ability to switch between components.

Options

Attribute	Description	Required
type	Object type. One of: `cluster`, `service`, `component`, `host`, or `hostprovider`	Yes
state	State value	Yes
service_name	Service name	No
component_name	Component name	No
host_id	Host identifier that is used to set host multistates from a hostprovider	No

> **NOTE**
> You can change the state of one service from another one, yet there is a risk of running concurrent jobs that change the same state. Be careful.

Examples

```
- adcm_multi_state_set:
    type: "cluster"
    state: "state value"
```

```
- adcm_multi_state_set:
    type: "service"
    service_name: "First"
    state: "state value"
```

```
- adcm_multi_state_set:
    type: "component"
    component_name: "another_component"
    state: "state value"
```

```
- adcm_multi_state_set:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    state: "state value"
```

```
- adcm_multi_state_set:
    type: "host"
    host_id: {{ host_id }}
    state: "state value"
```

Return values

```
state:
  returned: success
  type: str
  example: "operational"
```

adcm_multi_state_unset

This is a special ADCM-only module that is useful to change multistates for various objects. Changes are supported for cluster, service, host, component, and hostprovider multistates.

Using `service_name` gives you the ability to switch between services.

Using `component_name` gives you the ability to switch between components.

Options

Attribute	Description	Required
type	Object type. One of: `cluster`, `service`, `component`, `host`, or `hostprovider`	Yes
state	State value	Yes
service_name	Service name	No
component_name	Component name	No
missing_ok	Boolean. If `true`, then no exception should be rised in case of the multistate being absent from an object. The default value is `false`	No
host_id	Host identifier that is used to unset host multistates from a hostprovider	No

> **NOTE**
> You can change the state of one service from another one, yet there is a risk of running concurrent jobs that change the same state. Be careful.

Examples

```
- adcm_multi_state_unset:
    type: "cluster"
    state: "state value"
```

```
- adcm_multi_state_unset:
    type: "service"
    service_name: "First"
    state: "state value"
```

```
- adcm_multi_state_unset:
    type: "component"
    component_name: "another_component"
    state: "state value"
```

```
- adcm_multi_state_unset:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    missing_ok: true
    state: "state value"
```

```
- adcm_multi_state_unset:
    type: "host"
    host_id: {{ host_id }}
    state: "state value"
```

Return values

```
state:
  returned: success
  type: str
  example: "operational"
```

adcm_state

This is a special ADCM-only module that is useful to set states for various objects. Cluster, service, component, host, and hostprovider states are supported.

Using `service_name` gives you the ability to switch between services.

Using `component_name` gives you the ability to switch between components.

Options

Attribute	Description	Required
type	Object type. One of: `cluster`, `service`, `component`, `host`, or `hostprovider`	Yes
state	State value	Yes
service_name	Service name	No
component_name	Component name	No
host_id	Host identifier that is used to unset host multistates from a hostprovider	No

> **NOTE**
> You can change the state of one service from another one, yet there is a risk of running concurrent jobs that change the same state. Be careful.

Examples

```
- adcm_state:
    type: "cluster"
    state: "state value"
register: out
```

```
- adcm_state:
    type: "service"
    service_name: "First"
    state: "state value"
```

```
- adcm_state:
    type: "component"
    component_name: "another_component"
    state: "state value"
```

```
- adcm_state:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    missing_ok: true
    state: "state value"
```

```
- adcm_state:
    type: "host"
    host_id: {{ host_id }}
    state: "state value"
```

Return values

```
state:
  returned: success
  type: str
  example: "operational"
```

adcm_check

This plugin is designed to log results of JSON log storage checks. The log containing those results is also displayed in the ADCM web interface.

Every call of the `adcm_check` plugin adds an entry to the JSON log storage. You can invoke `adcm_check` with a single job identifier any number of times per playbook.

Options

Attribute	Description	Required
group_title	Group check name	No
group_success_msg	Group check result success description	No
group_fail_msg	Group check result fail description	No
title	Check name	Yes
result	Check result	Yes
msg	Check results description	Yes, if no `success_msg` and `fail_msg` fields
success_msg	Check result success description	Yes, if no `msg` field
fail_msg	Check result fail description	Yes, if no `msg` field

Examples

```
- name: ADCM Check
  adcm_check:
    title: "Check"
    msg: "This is message"
    result: yes
```

```
- name: ADCM Check
  adcm_check:
    title: "Check"
    success_msg: "This is success message"
    fail_msg: "This is fail message"
    result: yes
```

```
- name: ADCM check
  adcm_check:
    group_title: "Group 1"
    group_success_msg: "This is success message"
    group_fail_msg: "This is fail message"
    title: "Check"
    msg: "This is message"
    result: yes
```

adcm_custom_log

This plugin is designed to log results of JSON or TXT log storage checks. The log containing those results is also displayed in ADCM.

Every call of the `adcm_custom_log` plugin adds an entry to the JSON or TXT log storage. You can invoke `adcm_custom_log` with a single job identifier any number of times per playbook.

Options

Attribute	Description	Required
name	Log name	Yes
format	Format: `json` or `txt`	Yes
path	File path	Yes, if no `content` field
content	Text	Yes, if no `path` field

Examples

```
- name: custom log
  adcm_custom_log:
    name: "Custom log json"
    format: "json"
    path: "/home/user/log.json"
```

```
- name: custom log
  adcm_custom_log:
    name: "Custom log txt"
    format: "txt"
    content: "This is log"
```

adcm_config

This is a special ADCM-only module that is useful to set the specified config keys for various objects. Cluster, service, component, host, and hostprovider configs are supported.

Using `service_name` gives you the ability to switch between services.

Using `component_name` gives you the ability to switch between components.

Options

Attribute	Description	Required
type	Object type. One of: `cluster`, `service`, `component`, `host`, or `hostprovider`	Yes
parameters	The option for declaring a set of key/value pairs	No
key	Key name	Yes, if no `parameters`
value	Key value. Can contain several key/value pairs (`key1: value1`, `key2: value2`, and so on)	No
active	The option for changing the state of an activatable group	No
service_name	Service name	Yes, if `type = service`
component_name	Component name	Yes, if `type = component`
host_id	Host identifier that is used to set host config from a hostprovider	Yes, if `type = host`

> **NOTE**
> You can change the config keys of a service from another service, yet there is a risk of running concurrent jobs that change the same object. Be careful.

Examples

```
- adcm_config:
    type: "service"
    service_name: "First"
    key: "some_int"
    value: 111
register: out
```

```
- adcm_config:
    type: "service"
    service_name: "my service"
    key: "some_activatable_group"
    active: true
```

```
- adcm_config:
    type: "cluster"
    key: "some_map"
    value:
        key1: value1
        key2: value2
```

```
- adcm_config:
    type: "component"
    component_name: "another_component"
    key: "some_map"
    value: value
```

```
- adcm_config:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    key: "key"
    value: value
```

```
- adcm_config:
    type: "host"
    parameters:
      - key: "some_group/some_string"
        value: "string"
      - key: "some_map"
        value:
          key1: value1
          key2: value2
      - key: "some_string"
        value: "string"
```

```
- adcm_config:
    type: "host"
    host_id: {{ host_id }}
    key: "{{ config.key }}"
    value: "{{ config.value }}"
```

Return values

```
value:
  returned: success
  type: complex
```

adcm_manage_revision

This plugin is designed to allow a bundle to check for changes in the configuration parameter values when applying the configuration of an object (such as cluster, service, or component) and create a configuration revision, for example, if a certain configuration parameter is changed.

Options

Attribute	Description	Required
operation	Operation executed by the plugin. Possible values: - `get_primary_diff` — receives changes between parameter values of the current primary configuration version and the latest configuration with the commited revision; - `set_primary_revision` — marks the configuration of the object as applied, in other words, creates the revision.	Yes
objects	List of objects to the configurations of which the specified operation will be applied	Yes
type	Object type	Yes
service_name	Service name	No
component_name	Component name	No

Examples

```
- name: Getting difference between current/previous
  adcm_manage_revision:
    operation: get_primary_diff
    objects:
      - type: component
        service_name: "kafka"
        component_name: "broker"
  register: diff_result

- name: Set configuration as applied
  adcm_manage_revision:
    operation: set_primary_revision
    objects:
      - type: component
        service_name: "kafka"
        component_name: "broker"
```

adcm_add_host

This module is designed to enable adding new hosts to the ADCM database. The module should be run only in the hostprovider context. The hostprovider identifier is taken from the context.

Options

Attribute	Description	Required
fqdn	Fully qualified domain name of a new host	Yes
description	Optional description of a new host	No

Examples

```
- adcm_add_host:
    fqdn: my.host.org
    description: "my cool host"
```

Return values

```
result:
  host_id: 42
```

adcm_delete_host

This module is designed to enable removing the hosts from the ADCM database. The module should be run only in the host context. The host identifier is taken from the context.

Examples

```
- adcm_delete_host:
```

Return values

None

adcm_add_host_to_cluster

This module is designed to enable adding the existing hosts to a cluster in the ADCM database. The module should be run only in the cluster, service, or component context. The cluster identifier is taken from the context.

Options

Attribute	Description	Required
fqdn	Fully qualified domain name of the new host	You should specify either `fqdn` or `host_id` of the new host
host_id	Host identifier of the new host in the ADCM database	You should specify either `fqdn` or `host_id` of the new host

Examples

```
- adcm_add_host_to_cluster:
    fqdn: my.host.org
```

```
- adcm_add_host_to_cluster:
    host_id: 42
```

Return values

None

adcm_remove_host_from_cluster

This module is designed to enable removing the hosts from a cluster in the ADCM database. The module should be run only in the cluster or service context. The cluster identifier is taken from the context.

Options

Attribute	Description	Required
fqdn	Fully qualified domain name of the removed host	You should specify either `fqdn` or `host_id` of the removed host
host_id	Host identifier of the removed host in the ADCM database	You should specify either `fqdn` or `host_id` of the removed host

Examples

```
- adcm_remove_host_from_cluster:
    fqdn: my.host.org
```

```
- adcm_remove_host_from_cluster:
    host_id: 42
```

Return values

None

adcm_hc

This module is designed to enable changing the host-component map for a cluster. You can break the component constraints during the sequence of operations. All constraints are checked only once during the operation sequence being performed. The module should be run only in the cluster, service, or component context. The cluster identifier is taken from the context.

Examples

```
- name: move master component from host1.company.com to host2.company.com
  adcm_hc:
    operations:
      -
        action: "remove"
        service: "hadoop"
        component: "master"
        host: "host.company.com"
      -
        action: "add"
        service: "hadoop"
        component: "master"
        host: "host2.company.com"
```

Return values

None

adcm_delete_service

This module is designed to enable removing a service from the ADCM database. The module should be run only in the cluster or service context. The service name is taken from the context, if run in the service context. If run in the cluster context, the service name should be given.

Options

Attribute	Description	Required
service	Service name for the cluster context	No

> **NOTE**
> If you run this module in the service context, the call of this module should be placed last in the list of the Ansible jobs. This is due to the removal of a service in the context of which the current playbook is being run.

Examples

In service context:

```
- adcm_delete_service:
```

In cluster context:

```
- adcm_delete_service:
    service: "First"
```

Return values

None

adcm_change_maintenance_mode

This module is designed to switch the maintenance mode of a host, service, or component. If an object is in maintenance mode, it is set to normal mode and vice versa.

Options

Attribute	Description	Required
type	Object type. One of: `host`, `service`, or `component`	Yes
value	Value for switching the maintenance mode. One of: `True` or `False`	Yes

Examples

```
- name: Change host maintenance mode to True
  adcm_change_maintenance_mode:
    type: host
    value: True

- name: Change service maintenance mode to False
  adcm_change_maintenance_mode:
    type: service
    value: False
```

Return values

None

adcm_change_flag

This module is designed to toggle flags on hosts, services, components, clusters, and hostproviders, signaling changes in status.

Options

Attribute	Description	Required
operation	Operation with the flag. One of: `up`, `down`	Yes
name	Internal flag name. If this parameter is not specified, all object's flags will be lowered when the `down` operation is performed	No
msg	Additional flag message for use in specific patterns	No
objects	List of the services or components for which you intend to raise/lower the flag. If this parameter is not specified, the flag will be raised or lowered on the action context object. If you wish to raise or lower the flag on the entire cluster, an action within the cluster context is required	No

Examples

```
- name: "Raise a flag for kafka_broker"
  adcm_change_flag:
    operation: "up"
    name: "adcm_kafka__broker_raise"
    objects:
      - type: component
        service_name: "kafka"
        component_name: "kafka_broker"

- name: "Down an outdated flag"
  adcm_change_flag:
    operation: "down"
    name: "adcm_outdated_config"
    objects:
      - type: service
        service_name: "{{ roles_generic_args.service_name }}"
```

Return values

```
value:
  returned: success
  type: complex
  example:
    failed: false
    changed: true
```

Ansible inventory groups

Mikhail Serov

Contents

To Table of Contents

Basic groups

Groups for operations with the hc_acl atrribute

Group for operations defined on hosts

Groups for maintenance mode

Examples

On this page, we will call the **group** (or **inventory group**) the set of hosts for the Ansible inventory. Inventory groups are needed so that the bundle developer can use them in playbooks. The amount and types of inventory groups depend on the components defined in the prototype.

Inventory file with groups is generated every time when ADCM runs any action, job, or subjob. The generated inventory files are named *inventory.json* and are located in the mount point of your file system (typically */opt/adcm/run/...*). You also can download an inventory file using the icon.

The inventory groups that ADCM can form are described below.

Basic groups

These groups include hosts selected on the basis of common characteristics.

Group	Description	Bundle type
CLUSTER	Group that includes all the hosts in the cluster, excluding hosts that are in maintenance mode	Cluster
PROVIDER	Group that includes all hosts of the hostprovider	Infrastructure
HOST	Group that includes a host on which a current action is run	Infrastructure
<service name>	Group that includes all hosts to which the corresponding service has been added, excluding hosts that are in maintenance mode	Cluster
<service name>.<component name>	Group that includes all hosts to which the corresponding component has been added	Cluster

Groups for operations with the hc_acl atrribute

These groups are generated by launching an action that has the `hc_acl` attribute defined. The groups are named according to the operations.

Group	Description	Bundle type
<service name>.<component name>.add	Group that includes all the hosts that the user has selected to add the corresponding component	Cluster
<service name>.<component name>.remove	Group that includes all the hosts that the user has selected to remove the corresponding component	Cluster

Group for operations defined on hosts

This group includes a host where an action with the `host_action: true` attribute was launched.

Group	Description	Bundle type
target	Group that includes any host running an operation	Cluster

Groups for maintenance mode

These groups include hosts that are in maintenance mode.

Group	Description	Bundle type
<service name>.maintenance_mode	Group that includes all hosts to which the corresponding service has been added, and those hosts are in maintenance mode	Cluster
<service name>.<component name>.maintenance_mode	Group that includes all hosts to which the corresponding component has been added, and those hosts are in maintenance mode	Cluster
CLUSTER.maintenance_mode	Group that includes all cluster hosts that are in maintenance mode	Cluster

Examples

Consider a cluster that contains four hosts. The `server` component is located on the `test-zookeeper-1` host, the `client` component is located on the `test-zookeeper-2` host. The `server` component is also located on the `test-zookeeper-4` host, which has been put into maintenance mode in a separate group.

There is also the ZooKeeper service with `server` and `client` as its components. The service includes all hosts that have components of this service, hence, ZooKeeper includes `test-zookeeper-1` and `test-zookeeper-2`.

If you need an action that expands ZooKeeper to the `test-zookeeper-3` host, then the *inventory.json* file would be as shown below.

```
{
   "all": {
      "children": {
         "CLUSTER": {
            "hosts": {
               "test-zookeeper-1": {
                 ...
               },
               "test-zookeeper-2": {
                 ...
               },
               "test-zookeeper-3": {
                 ...
               }
            }
         },
         "zookeeper.server": {
            "hosts": {
               "test-zookeeper-1": {
                 ...
               }
            }
         },
         "zookeeper": {
            "hosts": {
               "test-zookeeper-1": {
                 ...
               },
               "test-zookeeper-2": {
                 ...
               }
            }
         },
         "zookeeper.client": {
            "hosts": {
               "test-zookeeper-2": {
                 ...
               }
            }
         },
         "zookeeper.server.add": {
            "hosts": {
               "test-zookeeper-3": {
                 ...
               }
            }
         },
         "zookeeper.server.maintenance_mode": {
            "hosts": {
               "test-zookeeper-4": {
                 ...
               }
            }
         }
      }
   }
}
```

Now consider an SSH hostprovider bundle and two hosts (`ssh-1` and `ssh-2`) that were created via this hostprovider.

When running the ***Create users*** action on this hostprovider, the *inventory.json* file will look similar to the one presented below.

```
{
    "all": {
        "children": {
            "PROVIDER": {
                "hosts": {
                    "ssh-1": {
                        ...
                    },
                    "ssh-2": {
                        ...
                    }
                }
            }
        }
    }
}
```

Ansible inventory structure

Mikhail Serov

Contents

To Table of Contents

- Inventory file for a cluster bundle
- Inventory file for an infrastructure bundle
- Inventory file section for hosts

There are two bundle types in ADCM. On this page, the Ansible inventory file structure is described for each type.

There is no `config` section for hosts in an inventory file as well. Everything specified for a host in the prototype `config` section is added directly to host dictionary.

Inventory file for a cluster bundle

The cluster is specified inside *inventory.json* via the attributes given in the table below.

Inventory file cluster attributes

Attribute	Description
id	Internal cluster identifier
state	Current state of a cluster
multi_state	Additional current state of a cluster
before_upgrade	Cluster configuration that existed before an upgrade
config	Current cluster configuration
name	Cluster name defined by the user
version	Cluster version according to prototype description
edition	Cluster edition according to prototype description

Services are specified via the following attributes.

Inventory file service attributes

Attribute	Description
<service name>	Service name according to prototype description
id	Internal service identifier
state	Current state of a service
multi_state	Additional current state of a service
before_upgrade	Service configuration that existed before an upgrade
config	Current service configuration
display_name	Service display name according to prototype description
version	Service version according to prototype description
maintenance_mode	Current maintenance mode state for a service
<component name>	Component name according to prototype description

Inventory file component attributes

Attribute	Description
component_id	Internal component identifier
state	Current state of a component
multi_state	Additional current state of a component
before_upgrade	Component configuration that existed before an upgrade
config	Current component configuration
display_name	Component display name according to prototype description
maintenance_mode	Current maintenance mode state for a component

Example:

```
{
  "all": {
    ...
    "vars": {
      "cluster": {
        "id": 210,
        "state": "installed",
        "multi_state": [],
        "before_upgrade": {
          "state": null
        },
        "config": {
          ...
        },
        "name": "adh",
        "version": "3.2.4_arenadata2_b1-for_autotest",
        "edition": "enterprise"
      },
      "services": {
        "zookeeper": {
          "id": 193,
          "state": "installed",
          "multi_state": [],
          "before_upgrade": {
            "state": null
          },
          "config": {
            ...
          },
          "display_name": "Zookeeper",
          "version": "3.5.10_arenadata1",
          "maintenance_mode": false,
          "SERVER": {
            "component_id": 582,
            "state": "installed",
            "multi_state": [],
            "before_upgrade": {
              "state": null
            },
            "config": {
              ...
            },
            "display_name": "Zookeeper Server",
            "maintenance_mode": false
          }
        }
      }
    }
  }
}
```

Inventory file for an infrastructure bundle

The hostprovider is represented in *inventory.json* via the attributes listed in the table below.

Inventory file hostprovider attributes

Attribute	Description
id	Internal hostprovider identifier
host_prototype_id	Internal host prototype identifier
name	Hostprovider name defined by the user
state	Current state of a hostprovider
multi_state	Additional current state of a hostprovider
before_upgrade	Hostprovider configuration that existed before an upgrade
config	Current hostprovider configuration

Example:

```
{
  "all": {
    ...
    "vars": {
      "provider": {
        "id": 236,
        "host_prototype_id": 1292,
        "name": "Yandex Compute VM Provider 3.5",
        "state": "created",
        "multi_state": [],
        "before_upgrade": {
          "state": null
        },
        "config": {
          ...
        }
      }
    }
  }
}
```

Inventory file section for hosts

The host is represented in *inventory.json* according to the attributes listed in the table below. `<host>` and its description are added to `<INVENTORY GROUP NAME>` according to the rules for forming inventory groups.

Inventory file host attributes

Attribute	Description
adcm_hostid	Internal host identifier
state	Current state of a host
multi_state	Additional current state of a host
ansible_ssh_common_args	Common SSH arguments for Ansible
instance	Host configuration according to prototype description
ansible_host	Name of the host to connect to
ansible_ssh_port	SSH port for Ansible
ansible_ssh_pass	Ansible SSH password
ansible_become	Option for the force escalation of privileges
ansible_become_pass	Password for the force escalation of privileges
ansible_user	Username that is used for SSH connection
ansible_ssh_private_key_file	Ansible SSH private key

Example:

```
{
  "all": {
    "children": {
      <INVENTORY GROUP NAME>:
        "hosts": {
         <host>: {
            "adcm_hostid": 229,
            "state": "running",
            "multi_state": [],
            "ansible_ssh_common_args": "-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null",
            "instance": {
              ...
            },
            "ansible_host": "10.92.40.43",
            "ansible_port": "22",
            "ansible_become": true,
            "ansible_user": "adcm",
            "ansible_ssh_private_key_file": "/adcm/data/file/host.229.ansible_ssh_private_key_file.",
          },
          ...
        }
      ...
    }
  }
}
```

Prototype definition guide

Mikhail Serov, Adelya Zvyagintseva

Contents

To Table of Contents

name

type

version

actions

adcm_min_version

allow_maintenance_mode

components

config

config_group_customization

description

display_name

edition

export

flag_autogeneration

import

license

monitoring

required

requires

upgrade

venv

This article describes the prototypes and their properties that you can use to describe services for integration with ADCM, particularly that are specified in configuration files for creating the ADCM bundles. The format of configuration (config) file should be YAML.

NOTE

The config file should be named *config.yaml* or *config.yml*. Multiple *config.yaml* files in a bundle are allowed. The name of the config file is reserved; other files should not be called by that name.

ADCM objects are created from prototypes. A **prototype** is a description of a single ADCM object.

Every prototype has the mandatory properties described below.

name

`name` is a short name designed for use in Ansible scripts.

type

Every prototype has one of the following types:

- `cluster`
- `service`
- `provider` (stands for hostprovider)

- `host`

version

`version` is a version of the prototype.

So the minimal working prototype should look like this:

```
---
- type: cluster
  name: control
  version: 1
```

A prototype also might have the optional properties described below.

actions

An **action** is an entity that can be used to describe an action on an ADCM object. Such an action is ultimately an Ansible playbook run with certain parameters under certain conditions.

You can describe special actions to bind their invocation to specific endpoints and UI buttons. You can choose from one of the four special actions names:

- `adcm_turn_on_maintenance_mode`
- `adcm_turn_off_maintenance_mode`
- `adcm_host_turn_on_maintenance_mode`
- `adcm_host_turn_off_maintenance_mode`

Special actions are specified as follows:

```
  actions:
    adcm_turn_on_maintenance_mode
```

Special actions should be described in the matching prototypes.

Name	Type of the prototype	Description
adcm_turn_on_maintenance_mode	Service/component	Turns on maintenance mode on an object via the corresponding UI element
adcm_turn_off_maintenance_mode	Service/component	Turns off maintenance mode on an object via the corresponding UI element
adcm_host_turn_on_maintenance_mode	Cluster	Turns on maintenance mode on a cluster host via the corresponding UI element
adcm_host_turn_off_maintenance_mode	Cluster	Turns off maintenance mode on a cluster host via the corresponding UI element
adcm_delete_service	Service	Deletes service via the corresponding UI element

Special actions that are described in a cluster prototype (`adcm_host_turn_on_maintenance_mode`, `adcm_host_turn_off_maintenance_mode`) should be host actions (the value of the `host_action` parameter should be `true`). Special actions are not supported for `config`, `hc_acl`, and `ui_options`.

Example

Every prototype might have `actions`. Actions should be described as follows:

```
---
- type: cluster
  name: control
  version: 1
  description: "Monitoring and Control Software"

  actions:
```

```
    install:
      display_name: "Install Monitor Server"
      description: |
         By click on this button you install monitoring and controlling server ...
      type: job
      script_type: ansible
      script: ansible/site.yaml
      allow_to_terminate: true
      params:
        ansible_tags: install
        jinja2_native: true
      states:
        available:
          - created
        on_success: installed
        on_fail: created
      config:
        quorum:
          type: integer
      log_files:
        - check
```

The `install` action from the example above has a `job` type and its `script` property specifies an Ansible playbook that resides in the *<bundle_root>/ansible/site.yaml* directory.

You can use a special dot syntax to specify a path to a script file. In this case, a script file will be searched in the same directory as *config.yaml*.

```
script: ./site.yaml
```

name

`name` is a short name that is unique for the given prototype.

display_name

`display_name` is a short human-readable name.

type

Action has a `type` statement. It defines the `job` and `task` clauses.

`job` is the type of action that starts a single script. Job is specified via the following statements:

- `script`
- `script_type`
- `params`

`task` is the type of action that is a chain of jobs. The sequence of jobs to be executed is specified via the `scripts` parameter. The job sequence is executed according to how it is written in `scripts`. If any script fails, the execution is stopped and the state of object is set to the value of `on_fail`.

```
actions:
  install:
    type: task
    scripts:
      -
          name: prepare
          display_name: Prepare to install
          script_type: ansible
          script: ansible/prepare.yaml
          on_fail: created
      -
          name: install
          display_name: Actual install
          script_type: ansible
```

```
          script: ansible/install.yaml
          on_fail: prepare
    states:
      available:
        - created
        - prepare
      on_success: installed
```

script

`script` is a parameter that takes it value depending on the `script_type` value.

In case of `script_type = ansible`, the `script` parameter denotes the path to the script file.

In case of `script_type = internal`, the `script` parameter can take the value from one of the three ADCM functions:

- `bundle_revert` — this function returns the state of the cluster as it was before the upgrade. Cluster state includes its configuration, the set of all cluster services, hosts, and components, as well as the host-component mapping.

- `hc_apply` — this function is useful when action includes `hc_acl` and it becomes necessary to commit the host-component mapping changes regardless of the action execution result or after the certain subjob execution. The `hc_apply` function commits the host-component mapping changes in accordance with data received from the user. You can use the optional `params` parameter when it is necessary to commit partial changes. In other words, with `params` you can specify a rule similarly to `hc_acl` (the service and component names, and also the `add` or `remove` operation), that will commit part of the host-component mapping associated with the specified rule.

```
- name: apply hc
  script_type: internal
  script: hc_apply
  params:
    rules:
      - service: redis
        component: server
        action: add
      - service: redis
        component: server
        action: remove
```

- `config_apply` — this function is useful when action includes `wizard_template` and it becomes necessary to save changes in configuration parameter values, for example, according to the data received from a user. You need to specify the rule of committing configuration parameter values and objects using the `params` parameter which describes the object, the service and component names corresponding to the object type, and also the set of key/value pairs.

NOTE

The `config_apply` function is available from version 2.8.0.

```
- name: save
  display name: "Save configuration on target objects"
  script_type: internal
  script: config_apply
  params:
    changes:
      - object:
          type: cluster
        parameters:
          - key: "ssl_default_config/keystore_password"
            value: "my_secrete_value"
      - object:
          type: service
          service_name: "hive"
        parameters:
          - key: "core-site_xml/keystore_password"
            value: "my_secrete_value"
      - object:
          type: component
          service_name: "hive"
          component_name: "server"
        parameters:
          - key: "core-site_xml/keystore_password"
            value: "my_value"
```

```yaml
- name: state_2
  display name: "State 2"
  script_type: internal
  script: config_apply
  params:
    changes:
      - object:
          type: cluster
        parameters:
          - key: "{{ ssl_key }}"
            value: "{{ action.process.stages.manage_ssl_stage.configure_ssl.config.ssl_config }}"
      - object:
          type: service
          service_name: "{{ roles_generic_args.service_name }}"
        parameters:
          - key: "{{ ssl_key }}"
            value: "{{ action.process.stages.manage_ssl_stage.configure_ssl.config.ssl_config }}"
      - object:
          type: component
          service_name: "{{ roles_generic_args.service_name }}"
          component_name: "{{ roles_generic_args.component_name }}"
        parameters:
          - key: "{{ ssl_key }}"
            value: "{{ action.process.stages.manage_ssl_stage.configure_ssl.config.ssl_config }}"
```

- `bundle_switch` — this function upgrades all prototypes to their new versions. The function can be used only for `upgrade`, it cannot be used in the common actions.

`script` is a mandatory parameter if `type = job`.

script_type

`script_type` is a parameter that denotes the type of the script, either `ansible` or `internal`.

`script_type` is a mandatory parameter if `type = job`.

scripts

`scripts` is a parameter that denotes the sequence of jobs to be executed.

`scripts` is a mandatory parameter if `type = task`.

params

You can also pass any additional parameter to the Ansible call. They will be used in `ansible-playbook` calls.

One special case is parameter `ansible_tags`. That parameter turns to be `--tags` parameter of an `ansible-playbook` call.

You can also pass the `jinja2_native` parameter. This parameter will be written to the *ansible.cfg* file. This option preserves variable types during template operations.

states

Action might have a `states` optional statement. It defines three clauses:

- `available` — list of states of an object for which action will be available;
- `on_success` — state in which object will be set in case of actions success;
- `on_fail` — state in which object will be set in case of actions fail.

The `on_success` and `on_fail` states are optional. If they are missed, the state of an object is not changed after actions success or/and fail, respectively.

The `states` statement also has two reserved values:

- `created` — the primary state in which objects are immediately after creation;

- `upgrading` — the state to which the cluster can be put into. In this state, actions such as removing a host or service from the cluster are not available to the user.

masking

`masking` is an alternative way to describe the action availability in UI and API, which allows you to describe a set of basic (`states`) and extended (`multi_state`) states.

Use the `masking` parameter if you need to describe the action availability using extended states specified in the `multi_state` parameter.

IMPORTANT

DSL `masking` is not compatible with the `states` DSL provided above.

```
---
actions:
  install:
  display_name: "Install Monitor Server"
  type: job
  masking:
      # Action will be shown if both condition under state and multi_state are met.
      state:
          available:
              # If the state equal to any of this, then condition is true
              - "state_value1"
              - "state_value2"
          unavailable:
              # If you place unavailable, then no available should be there
              - "state_value1"
              - "state_value2"
      multi_state:
          available:
              # If we have any of this multistate, then condition is true
              - "multi_state_1"
              - "multi_state_2"
          unavailable:
              # If you place unavailable, then no available should be there
              - "multi_state_3"
              - "multi_state_4"
  on_fail:
      state: "new_sate_value"
      multi_state:
          set:
              - "multi_state3"
          unset:
              - "multi_state4"
  on_success:
      state: "new_sate_value"
      multi_state:
          set:
              - "multi_state3"
          unset:
              - "multi_state4"
```

To simplify the rule description, you could use the `any` scalar that selects all possible values.

```
---
  masking:
  state:
      available: "any"
      unavailable: "any"
  multi_state:
      available: "any"
      unavailable: "any"
```

If the `masking` state is omitted, then the action is considered available on every `state` or `multi_state`. Hence, the following variants are equal:

```
---
  masking:
```

```
---
  masking:
      state:
```

```
---
  masking:
      state:
          available: "any"
```

The following variants are equal too:

```
---
  masking:
```

```
---
  masking:
      multi_state:
```

```
---
  masking:
      multi_state:
          available: "any"
```

config

Action might have a `config` optional statement. It can define a list of values as in config parameters. When the action is run via UI, all those values will be prompted by the user before the action starts.

You can later refer those values in the Ansible script:

```
{{ job.config.quorum }}
```

The `config` statement works with static configuration parameters that are displayed the same way every time an action is started. If you want to work with dynamic configuration parameters, use the `config_jinja` statement.

scripts_jinja

An action might have a `scripts_jinja` optional statement. This statement points to the file of the Jinja format (ADCM uses the 2.xx version).

NOTE

The `scripts_jinja` statement is deprecated and will no longer be supported in ADCM in Q4 2026.

The `scripts_jinja` statement is designed for dynamic formation of a set of subjobs based on the context depending on the conditions, in other words, on the cluster topology (for example, whether a particular service was added to the cluster or not). After the action started, a file is rendered in the Jinja format (Jinja2 template) to generate a sequence of subjobs. During the template rendering, the context is formed according to the inventory file, except config group changes, as well as variables and facts that Ansible generates during a playbook execution (including additional action details, and also what was entered and selected by the user when configuring the action).

The `scripts_jinja` statement is only supported for actions of the `task` type and is mutually exclusive with the `scripts` parameter (static subjob generation).

```
---
actions:
  - name: Add/remove components
    type: task
    scripts_jinja: scripts_jinja/manage_component_scripts.j2
```

The *manage_component_scripts.j2* file:

```
{%- if (groups['zookeeper.SERVER.add'] | length) > 0 %}
```

```
- name: zookeeper
  display_name: "Zookeeper"
  script_type: ansible
  script: ansible/playbooks/cluster/zookeeper_install.yaml
  params:
    ansible_tags: repo_add, install, configure, configure_main_info, start, bootstrap
{%- endif %}
{%- if (groups['zookeeper.SERVER.remove'] | length) > 0 %}
- name: zookeeper
  display_name: "Zookeeper"
  script_type: ansible
  script: ansible/playbooks/cluster/zookeeper_remove.yaml
  params:
    ansible_tags: stop, remove
{%- endif %}
- name: zookeeper
  display_name: "Zookeeper"
  script_type: ansible
  script: ansible/playbooks/cluster/zookeeper_install.yaml
  params:
    ansible_tags: configure
```

When specifying a Jinja2 template, you can use a `groups` variable. This variable contains a list of host names without host configurations. An example of Jinja2 template rendering context variables is shown below.

```
"cluster": {
   "config": {
      "reliability_control": {
         "retries": 3153600,
         "delay": 10,
         "timeout": 31536000
       }
   },
   "id": 210,
   "name": "adh bu"
   "state": "installed",
   "multi_state": [],
   "before_upgrade": {…},
   "version": "3.2.4_arenadata2_b1-for_autotest",
   "edition": "enterprise",
   "imports": None  # absent if no imports
},
"services":{
   "zookeeper":{
      "id" :193,
      "state":"installed",
      "multi_state":[],
      "before_upgrade":{"state":null},
      "config": {},
      "display_name": "Zookeeper",
      "version": "1.0",
      "maintenance_mode": False,
      "SERVER":{
         "component_id": 582,
         "state":"installed",
         "multi_state":[],
         "before_upgrade":{...},
         "config":{},
         "display_name": "Zookeeper Server",
         "maintenance_mode": false
      },
   },
},
"groups": {
   "CLUSTER": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
   "zookeeper": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
   "zookeeper.SERVER": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
```

```
},
"task":{
   "config": {},
   "verbose": false
},
"action": {
   "owner_group": "zookeeper",
   "name": "install"
}
```

config_jinja

An action might have a `config_jinja` optional statement. This statement points to the file of the Jinja format (supported in ADCM 2.xx).

NOTE

The `config_jinja` statement is deprecated and will no longer be supported in ADCM in Q4 2026.

The `config_jinja` statement is designed to work with dynamic configuration parameters. Here "dynamism" means the ability to display (or hide) a certain set of parameters depending on the conditions, in other words, on the cluster topology (for example, whether a particular service was added to the cluster or not). When metadata on action is obtained (via the GET request in the API), a file is rendered in the Jinja format (Jinja2 template). During the template rendering, the context is formed according to the inventory file, except config group changes, as well as variables and facts that Ansible generates during a playbook execution.

The `config_jinja` statement is mutually exclusive with the `config` statement which is intended to work with static configuration parameters.

```
---
actions:
  - name: Restart and Reboot
    type: job
    script_type: ansible
    script: ansible/restart.yaml
    config_jinja: config_jinja/restart.j2
```

The *restart.j2* file:

```
- reboot_servers:
  type: boolean
  default: false
  display_name: Reboot servers
{%  if cluster.state == 'created' %}
- configure_etc:
  type: boolean
  default: false
  display_name: Configure /etc/hosts
{% endif %}
```

When specifying a Jinja2 template, you can use a `groups` variable. This variable contains a list of host names without host configurations. An example of Jinja2 template rendering context variables is shown below.

```
"cluster": {
   "config": {
      "reliability_control": {
         "retries": 3153600,
         "delay": 10,
         "timeout": 31536000
       }
   },
   "id": 210,
   "name": "adh bu"
   "state": "installed",
   "multi_state": [],
   "before_upgrade": {…},
   "version": "3.2.4_arenadata2_b1-for_autotest",
```

```
   "edition": "enterprise",
   "imports": None  # absent if no imports
},
"services":{
   "zookeeper":{
      "id" :193,
      "state":"installed",
      "multi_state":[],
      "before_upgrade":{"state":null},
      "config": {},
      "display_name": "Zookeeper",
      "version": "1.0",
      "maintenance_mode": False,
      "SERVER":{
         "component_id": 582,
         "state":"installed",
         "multi_state":[],
         "before_upgrade":{...},
         "config":{},
         "display_name": "Zookeeper Server",
         "maintenance_mode": false
      },
   },
},
"groups": {
   "CLUSTER": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
   "zookeeper": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
   "zookeeper.SERVER": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
},
},
"action": {
   "owner_group": "zookeeper",
   "name": "install"
}
```

log_files

Action might have a `log_files` optional statement. It defines the list of log tags. Currently, a `check` log tag is supported, and you can use the `adcm_check` Ansible module with it.

allow_to_terminate

Action might have an `allow_to_terminate` optional statement. This is a Boolean value. If the value is not explicitly set, then the default value is `false`. If `allow_to_terminate` is `true`, then it is possible to cancel the running Ansible playbook and stop the action, otherwise it is not possible.

The `allow_to_terminate` statement is also applied to all jobs the action consists of. When one of such jobs is stopped, the action continues to run, starting with the job that follows the stopped one. Essentially, `allow_to_terminate` being specified for an action means `allow_to_terminate` being specified for all jobs of that action.

allow_in_maintenance_mode

Action might have an `allow_in_maintenance_mode` optional statement. This is a Boolean value. If the value is not explicitly set, then the default value is `false`. If `allow_in_maintenance_mode` is `true` and `allow_maintenance_mode` in the prototype is also `true`, then it is possible to run an action even in case of object or just one of its hosts being in maintenance mode.

allow_for_action_host_group

Action might have the `allow_for_action_host_group` optional statement. This is a Boolean value. If the value is not explicitly set, then the default value is `false`. If `allow_for_action_host_group` is `true`, then it is possible to launch an action on action host groups.

```
actions:
  upgrade:
    type: job
    script_type: ansible
    script: ansible/site.yaml
    allow_for_action_host_group: true
```

Limitations of using the `allow_for_action_host_group` parameter:

- Composite actions (for example, the ***Upgrade*** action) are not allowed to launch on action host groups.
- The simultaneous use of this parameter and the `host_action` parameter is not supported.

ui_options

This block allows you to tweak how UI displays and performs an action. When you hover the cursor over an action in the UI, the popup message specified in the `disclaimer` field is displayed.

```
actions:
  install:
    type: job
    script_type: ansible
    script: ansible/site.yaml
    ui_options:
       disclaimer: "Enter your disclaimer"
```

host_action

An action might have a `host_action` optional statement which is a Boolean value. The default value is `false`. If the value is `true`, then the action appears in the drop-down list of actions on the ***Clusters** → **<Cluster name>** → **Hosts** → **<Host name>** → **Host-Components** → **<Component name>*** page. In this case, the action is performed on the selected host and a target group that includes the one selected host is created.

It is advisable to declare action (`host_action: true`) only in the context of a component, since such actions allow you to manage specific components on specific hosts.

```
actions:
  install:
    type: job
    script_type: ansible
    script: action.yaml
    host_action: true
```

hc_acl

Action might have a `hc_acl` optional statement. If `hc_acl` is present when an action is run via UI, then a new host-component map will be requested from the user before the action starts.

```
actions:
  expand:
    type: job
    script_type: ansible
    script: ansible/site.yaml
    states:
      available: all
    hc_acl:
      -
        service: hadoop
        component: datanode
        action: add
```

```
    -
      service: hadoop
      component: server
      action: remove
```

`hc_acl` defines the list of three clauses that specify operations to modify the component distribution among hosts:

- `service` — name of a service to work with.
- `component` — name of a component to work with.
- `action` — operation type. Can be `add` or `remove`.

venv

There is a way to specify the Ansible version for the `ansible` script type actions via the `venv` optional statement.

Valid values:

- `default` — launch action in Ansible 2.8 environment;
- `2.9` — launch action in Ansible 2.9 environment;
- `2.16` — launch action in Ansible 2.16 environment (starting with ADCM 2.8.0).

The default value is `default`.

NOTE

With the release of ADCM 2.8.0, Ansible 2.8 is deprecated and will be removed from ADCM in Q3 2026.

Change the default environment for all actions on an object:

```
---
- type: cluster # service component hostprovider host
  name: control
  version: 1

  venv: "2.9"
```

You also have a way to tweak `venv` for any actions:

```
actions:
  install:
    type: job
    script_type: ansible
    script: action.yaml
    venv: "2.9"
```

wizard_template

Action might have the `wizard_template` optional statement which is the template of the flow formed as a result of its rendering.

Unlike the `config` statement, the `wizard_template` statement allows you to do the following:

- divide an action configuration into several logical steps as modal windows;
- pass the template rendering context between steps;
- save the states of the steps that are already provided by the user.

The `wizard_template` statement is mutually exclusive with the following statements:

- `config`, which is intended to work with static configuration parameters;
- `config_jinja`, which is intended to work with dynamic configuration parameters;
- `hc_acl`, which is intended to commit the host-component mapping changes.

The specified statement is displayed in the ADCM web interface as a wizard which is the preparation phase of an action before its launch.

The flow is a sequence of the following elements:

- **Stages** — the containers of logically related steps.
- **Steps** — the minimal indivisible elements of a process performed by the user.

NOTE

The process occurs when a user clicks the action in the drop-down list of actions available to work with an object.

A step can be one of the two types:

- **Configuration** — a configuration filled in by the user. The configuration is dynamically generated during the rendering of the template specified in the `config_template` statement when receiving information about a specific step.
- **Operation** — a job representing a set of subjobs that is launched when executing the step. The set of subjobs is dynamically generated during rendering of the template specified in the `scripts_template` statement when executing the specific step. The example of a step of the operation type is given in the manage_install.j2 file (the `check_kerberos` step of the `manage_kerberos_stage` stage).
  Steps of the operation type provide the following options:
  - validating the action configuration parameter values entered by the user;
  - saving the action configuration parameter values entered by the user in configuration of the target object (cluster, service, or component).

Each stage includes the following parameters:

- `name` — unique stage name;
- `display_name` — name of the stage that is displayed in the UI for the user;
- `steps` — sequence of steps.
  The following attributes are used to describe the step:
  - common attributes:
    - `name` — step name that is unique to this stage;
    - `display_name` — wizard step name which is intended for use as the name of the modal window step;
  - attribute for steps of the configuration type: `config_template` — template for working with dynamic configuration parameters;
  - Attributes for steps of the operation type:
    - `ui_options` — attributes specific to UI, in particular, `button_name` — button name;
    - `scripts_template` — template for dynamic generation of the subjob sequence.

Template description format (the `<wizard|config|scripts>_template` statement) includes the following parameters:

- `file` — file that is used as a template and contains the following attributes:
  - `path` — path to the file with the *.j2* or *.py* extension.
  - `entrypoint` — name of the `callable()` function that should be called (without specifying the package or module). This attribute is used only if `engine` is set to `python`.
- `engine` — file processing engine. Possible values: `jinja2`, `python`.

Based on the received information about an action (if `wizard_template` is used), a process is created.

```
actions:
  install:
    display_name: "Install"
    type: task
    allow_to_terminate: true
    scripts_jinja: scripts_jinja/cluster/manage_install.j2
    wizard_template:
      file:
        path: wizard_jinja/manage_install.j2
      engine:
        type: jinja2
    states:
      available:
        - created
```

```
        - faulty_installed
```

Creating a process, among other things, includes rendering of the file specified in `wizard_template`. This results in a flow. During the template rendering, the context is formed according to the inventory file, except config group changes, as well as variables and facts that Ansible generates during a playbook execution.

The *manage_install.j2* file:

```
- name: manage_ssl_stage
  display_name: "Manage SSL"
  steps:
    - name: configure_ssl
      display_name: Configure SSL
      config_template:
        file:
          path: configs_jinja/manage_ssl.j2
        engine:
          type: jinja2
- name: manage_kerberos_stage
  display_name: "Manage Kerberos"
  steps:
    - name: configure_kerberos
      display_name: "Kerberos configuration"
      config_template:
        file:
          path: configs_jinja/manage_kerberos.j2
        engine:
          type: jinja2
    - name: check_kerberos
      display_name: "Check configuration"
      ui_options:
        button_name: Check
      scripts_template:
        file:
          path: scripts_jinja/cluster/save_config.j2
        engine:
          type: jinja2
{% if vars.services.hdfs is defined and vars.services.hdfs.state == 'created' %}
- name: hdfs_stage
  display_name: "Manage HDFS"
  steps:
    - name: configure_hdfs
      display_name: Configure HDFS
      config_template:
        file:
          path: configs_jinja/manage_hdfs.j2
        engine:
          type: jinja2
{% endif %}
- name: save_stage
  display_name: "Save configuration"
  steps:
    - name: save_configuration
      display_name: "Check configuration"
      ui_options:
        button_name: Check
      scripts_template:
        file:
          path: scripts_python/cluster/check.py
          entrypoint: generate_scripts
        engine:
          type: python
```

The *manage_ssl.j2* file:

```
- name: ssl_config
  display_name: "Enable SSL"
  type: group
{% if vars.cluster.config.ssl_config.enable_ssl %}
  activatable: true
{% else %}
  activatable: false
{% endif %}
  subs:
    - name: keystore_path
```

```
      display_name: "Keystore path"
      description: "Keystore path"
      type: string
      default: "{{ cluster.config.ssl_default_config.keystore_path }}"
    - name: keystore_password
      display_name: "Keystore password"
      description: "Keystore password"
      type: password
```

The *manage_kerberos.j2* file:

```
- name: kerberos_config
  display_name: "Enable Kerberos"
  type: group
{% if vars.cluster.config.kerberos_config.enable_kerberos %}
  activatable: true
{% else %}
  activatable: false
{% endif %}
  subs:
    - name: keystore_password
      display_name: "Keystore password"
      description: "Keystore password"
      type: password
      default: {{ action.process.stages.manage_ssl_stage.configure_ssl.config.ssl_config.keystore_password }}
```

The *save_config.j2* file:

```
{% set ssl_key = "ssl_default_config/keystore_password" %}
- name: validate
  display name: "Precheck"
  script_type: ansible
  script: some/playbook.yaml
  params:
    ansible_tags:
      - check
- name: state_2
  display name: "State 2"
  script_type: internal
  script: config_apply
  params:
    - object:
        type: cluster
      parameters:
{% if action.process.stages.manage_ssl_stage.configure_ssl.config.ssl_config.enable_ssl %}
        - key: "{{ ssl_key }}"
          value: "{{ action.process.stages.manage_ssl_stage.configure_ssl.config.ssl_config }}"
{% endif %}
{% if action.wizard.manage_kerberos_wizard.config.kerberos_config.enable_kerberos %}
        - key: "kerberos_default_config/keystore_password}"
          value: "{{ action.process.stages.manage_kerberos_stage.configure_kerberos.config.kerberos_config.keystore_password }}"
{% endif %}
```

The *manage_hdfs.j2* file:

```
- name: kerberos_config
  display_name: "DFS data dir"
  type: string
  default: {{ vars.services.hdfs.config.data_dir }}
```

The *check.py* file:

```
def create_script_template(display_name, name, params, script, script_type):
    """Template for creating action dictionaries with consistent structure."""
    return {
        'display name': display_name,
        'name': name,
        'params': params,
        'script': script,
        'script_type': script_type
    }
```

```
def generate_scripts(context: dict):
    cluster = context['cluster']
    action = context['action']
    script1 = create_script_template(
        display_name='Check',
        name='validate',
        params={'ansible_tags': ['check']},
        script='some/playbook.yaml',
        script_type='ansible'
    )

    script2_params = [{
        'object': {'type': 'cluster'},
        'parameters': [{
            'key':  'ssl_config',
            'value': action['process']['manage_ssl_stage']['configure_ssl']['config']['ssl_config']
        }]
    }]

    script2 = create_script_template(
        display_name='State 2',
        name='state_2',
        params=script2_params,
        script='config_apply',
        script_type='internal'
    )

    return [script1, script2]

  ### Return
  [{'display name': 'Check',
  'name': 'validate',
  'params': {'ansible_tags': ['check']},
  'script': 'some/playbook.yaml',
  'script_type': 'ansible'},
   {'display name': 'State 2',
  'name': 'state_2',
  'params': [{'object': {'type': 'cluster'},
              'parameters': [{'key': 'ssl_config', 'value': 'some_value'}]}],
  'script': 'config_apply',
  'script_type': 'internal'}]
```

When specifying a template, you can use a `groups` variable. This variable contains a list of host names without host configurations. An example of rendering of a context variable template for the `configure_kerberos` step of the `manage_kerberos_stage` stage is shown below.

NOTE

When forming a context, variables stored in the `action.process` dictionary are added from the previous step.

```
"cluster": {
   "config": {
      "reliability_control": {
         "retries": 3153600,
         "delay": 10,
         "timeout": 31536000
       }
   },
   "id": 210,
   "name": "adh bu"
   "state": "installed",
   "multi_state": [],
   "before_upgrade": {…},
   "version": "3.2.4_arenadata2_b1-for_autotest",
   "edition": "enterprise",
   "imports": None  # absent if no imports
},
"services":{
   "zookeeper":{
      "id" :193,
      "state":"installed",
```

```
      "multi_state":[],
      "before_upgrade":{"state":null},
      "config": {},
      "display_name": "Zookeeper",
      "version": "1.0",
      "maintenance_mode": False,
      "SERVER":{
         "component_id": 582,
         "state":"installed",
         "multi_state":[],
         "before_upgrade":{...},
         "config":{},
         "display_name": "Zookeeper Server",
         "maintenance_mode": false
      },
   },
},
"groups": {
   "CLUSTER": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
   "zookeeper": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
   "zookeeper.SERVER": [
      "ke-test-hadoop-25.ru-central1.internal",
   ]
},
"action": {
    "owner_group": "zookeeper",
    "name": "install",
    "process": {
      "current": {
            "step": "configure_kerberos",
            "stage": "manage_kerberos_stage"
        },
        "stages": {
          "manage_ssl_stage": {
              "configure_ssl": {
                  "config": {
                      "ssl_config": {
                          "keystore_path": "/etc/security/ssl",
                          "keystore_password": "ANSIBLE_VAULT:...."
                      }
                  }
              }
          }
      }
  }
}
```

adcm_min_version

Any cluster or hostprovider prototype might have an optional `adcm_min_version` property. This field defines the minimal version of ADCM that is required for this bundle to work correctly. If the installed ADCM instance version is less than `adcm_min_version` (defined in a bundle), then uploading of such bundle will cause an error.

allow_maintenance_mode

Every cluster prototype might have an optional `allow_maintenance_mode` property. This option enables support of the maintenance mode for all prototypes in a bundle. The default value is `false`.

```
---
- type: cluster
  name: control
  version: 1
  adcm_min_version: 2019.05.01.00
  allow_maintenance_mode: true
```

components

Service prototype might have **components**. Component itself is a prototype, so a component description also can contain `config` or `action` sections.

Example

`components` is a way of placing service components on hosts in a cluster.

```
---
- type: service
  name: server
  version: 1

  components:
    master:
      display_name: "Master Node"
      description: "This node control all data nodes (see below)"
      constraint: [1,2]
    node:
      display_name: "Data Node"
      constraint: [+]
    client:
      monitoring: passive
```

constraint

Every component might have an optional `constraint` property. It describes how many instances of this component should be installed in one cluster:

- `[1]` — exactly one component should be installed.
- `[0,1]` — one or zero components should be installed.
- `[1,2]` — one or two components should be installed.
- `[0,+]` — zero or any more components should be installed (default value).
- `[1,odd]` — one or more components should be installed; the total amount should be odd.
- `[0,odd]` — zero or more components should be installed; if more than zero, the total amount should be odd.
- `[odd]` — same as `[1,odd]`.
- `[1,+]` — one or more components should be installed.
- `[+]` — component should be installed on all hosts of a cluster.

monitoring

Every service/component might have an optional `monitoring` property. It defines if the service/component should be monitored. The default value is `active`. If you want to exclude a service/component from the monitoring system, set it to `passive`.

```
---
- type: service
  name: server
  version: 1

  components:
    client:
```

```
    monitoring: passive
```

requires

Components might have an optional `requires` property that describes dependencies between the components, i.e. which components need other components or services to be installed simultaneously.

Each component might require one or more components in the same service or in another service of this cluster. Also, a component may need one or more services. If a component requires another component that is absent from a host-component map, you cannot add this component to a host-component map as well. In some cases, all required components should be installed simultaneously. Dependencies defined at the component level affect the validation of the host-component map.

Dependencies can be viewed as another type of constraints (or metaconstraints) for a component.

The example below considers the `Hive` service with three components: `HiveServer`, `Metastore`, and `TezUI`. The `HiveServer` component needs the `HDFS` service, while the information about the `HDFS` service components is not required. The `TezUI` component needs the `TimeLineServer` component of the `YARN` service.

```
---
- type: service
  name: Hive
  version: 1.0

  components:
    HiveServer:
      requires:
        - service: HDFS
    Metastore:
      constraint: [0,1]
    TezUI:
      requires:
        - service: YARN
          component: TimeLineServer
      constraint: [0,1]
```

bound_to

IMPORTANT

The `bound_to` property is not supported since the 2.0.0 version.

Components might have an optional `bound_to` property. Those components should be located on the same hosts as the components they are bound to.

For example, the `server` component of the `pxf` service should be installed only on all those hosts where the `segment` components of the `gpdb` service are installed. It means the `server` component requires the `segment` component and cannot be installed separately.

Hence `bound_to` is yet another type of constraints (or metaconstraints) for a component.

```
---
- type: service
  name: GPDB
  version: 1.0

  components:
    Segment:

- type: service
  name: PXF
  version: 2.0

  components:
    Server:
      bound_to:
        service: GPDB
        component: Segment
```

config

Config is a general description of object configuration parameters. Configuration parameters can be combined into groups.

Example

There is a number of user-editable values that are related to every object and can be represented in UI.

```yaml
---
- type: cluster
  name: control
  version: 1
  description: "Monitoring and Control Software"

  config:
    - name: repos
      type: group
      subs:
         - name: ads
           display_name: "ADS repository URL"
           type: string
           default: https://ci.arenadata.io/artifactory/list/ads-centos7-x64-ads/
           required: no
```

In the above example you can see the `ads` parameter in the `repos` group. The `ads` parameter is not required. This means that it is possible for the user to leave it empty.

Group is not mandatory, so you can place your parameter at the highest level:

```yaml
---
- type: cluster
  name: control
  version: 1
  description: "Monitoring and Control Software"

  config:
    - name: ads
      type: string
      default: https://ci.arenadata.io/artifactory/list/ads-centos7-x64-ads/
      required: no
```

Group is just a visual representation in the UI.

Configuration parameters properties

Name	Mandatory	Values	Default	Description
type	Yes	See below	—	Type of configuration parameter. For more information, see the separate ***Types of configuration parameters*** section below
display_name	No	User defined	—	Readable name that is displayed in UI
description	No	User defined	—	Allows you to give the parameter a description
default	No	User defined	—	Allows you to set a default value
required	No	yes/no	Yes	If set to `yes`, then it is required to have a non-empty and valid value for the variable
read_only	No	See below	No	Makes a parameter read-only (protected from change) under certain conditions

writable	No	See below	Yes	Makes a parameter writable (available for changes) under certain conditions
ui_options	No	See below	No	Allows you to adjust the visual representation of a parameter in UI
group_customization	No	yes/no	No	Makes a parameter available to be changed via config groups
pattern	No	User defined	—	Validation rule for the entered configuration parameter value
ansible_options	No	See below	No	Allows you to specify additional settings for a configuration parameter when preparing Ansible inventory

The `ui_options` parameter allows you to tweak how UI depicts config.

The following options are available:

- Invisible
  Add `invisible: true` to your config to hide the config from UI.

  This option does not hide information from API and does not disable the `read_only` / `writable` logic. This is simply a UI tweak.

```
int:
  type: integer
  default: 1
  required: no
  ui_options:
    invisible: true
```

- Advanced
  There is a new checkbox that allows you to hide some parameters via labeling them `advanced`.

  Parameter labeled as `advanced` is not searchable.

```
float:
  type: float
  default: 1.0
  required: no
  ui_options:
    advanced: true
```

The `group_customization` parameter allows you to customize a parameter at the configuration group level.

```
---
 - name: ads
   type: string
   group_customization: true/false
```

The `read_only` parameter allows you to make any config parameter read-only (protected from change by the user or API) for the specified cluster/host/service states. The `writable` parameter removes the write protection.

The `quorum` parameter is read-only if a service state is `installed` or `running`:

```
config:
  quorum:
    type: integer
    read_only: [installed, running]
```

The `quorum` parameter is read-only for all service states except `created`:

```
config:
  quorum:
    type: integer
    writable: [created]
```

The `quorum` parameter is read-only for all service states:

```
config:
  quorum:
    type: integer
    default: 4
    read_only: any
```

With the `pattern` parameter, you can set a validation rule for the configuration parameter value according to a regular expression. Parameter is supported for the config parameters of the `password`, `string`, `text`, and `secrettext` types.

The value of the `pattern` parameter must be a string that is a valid regular expression according to the ECMA 262 regular expression dialect. The rules are outlined below:

- A single Unicode character (other than the special characters below) matches itself.
- `.` — matches any character except line break characters.
- `^` — matches only at the beginning of the string.
- `$` — matches only at the end of the string.
- `(…)` — groups a series of regular expressions into a single regular expression.
- `|` — matches either the regular expression preceding or following the `|` symbol.
- `[abc]` — matches any of the characters inside the square brackets.
- `[a-z]` — matches the range of characters.
- `[^abc]` — matches any character not listed.
- `[^a-z]` — matches any character outside of the range.
- `+` — matches one or more repetitions of the preceding regular expression.
- `*` — matches zero or more repetitions of the preceding regular expression.
- `?` — matches zero or one repetitions of the preceding regular expression.
- `+?`, `*?`, `??` — the `*`, `+`, and `?` qualifiers are all greedy. They match as much text as possible. Sometimes this behavior can be unwanted and you may want to match as few characters as possible.
- `(?!x)`, `(?=x)` — negative and positive lookahead.
- `{x}` — matches exactly `x` occurrences of the preceding regular expression.
- `{x,y}` — matches at least `x` and at most `y` occurrences of the preceding regular expression.
- `{x,}` — matches `x` occurrences or more of the preceding regular expression.
- `{x,y}?`, `{x,}?` — non-greedy versions of the above expressions.

Use only standard escapes like `\n`, `\r`, `\t` and notice that you also need to do JSON escaping.

When the `pattern` parameter is added, consider the upgrade procedure for the current bundle for an existing cluster:

1. Check if the config parameter value matches the `pattern` and fail the upgrade if it does not match.
2. Check if the config parameter value matches the `pattern` and migrate its value if necessary.
3. Check if the config parameter value matches the `pattern` and raise a flag to notify the user to change the config parameter themselves.

Additionally, for config parameters where the `pattern` parameter is present, it is recommended to reword the `description` parameter. You can add examples of characters the value can consist of.

```
config:
- name: example
  type: password
  display_name: "Example password"
  description: "Use uppercase and lowercase letters, as well as numbers"
  default: "qwerty56"
  pattern: "[a-z][A-Z][0-9]*?"
```

The `ansible_options` parameter allows you to specify options for setting a configuration parameter when preparing Ansible inventory.

The supported option is `unsafe` which marks the variable value as unsafe to prevent unsafe character substitution in accordance with the Ansible functionality.

```
---
 - name: adb
   type: string
```

```
    display_name: "Name of default database"
    default: {%adb
    ansible_options:
      unsafe: True
```

Types of configuration parameters

- `boolean` — boolean value: `True` or `False`.
- `file` — the content of this property is stored as a file on a file system. When you reference this variable in an Ansible script, it returns a full path to that file. If the default value is set, it should be the path to the file relative to the bundle root directory.

  User can edit the content of this file via UI. The file is updated every time after config is saved.

  It can be used to store SSH private keys for an Ansible script:

```
- type: host
  name: ssh
  version: 1

  config:
    ansible_private_key_file:
      type: file
      default: host/ssh.key
```

  You can use the special dot syntax to specify a path to the default file. In this case, the default file is expected to be in the same directory as the *config.yaml* file:

```
config:
  ansible_private_key_file:
    type: file
    default: ./ssh.key
```

- `float` — floating-point number. If the default value is not explicitly set, then it will be `0.0`. You can use the optional `min` and `max` parameters to limit the values:

```
config:
  average_load:
    type: float
    default: 1.0
    min: 0.5
    max: 3.5
```

- `integer` — integer number. If the default value is not explicitly set, then it will be `0`. You can use the optional `min` and `max` parameters to limit the values:

```
config:
  memory_size:
    type: integer
    default: 16
    min: 2
    max: 64
```

- `json` — this type stores arbitrary JSON data. The JSON structure is not preserved and can be modified during every config update:

```
config:
  population:
    type: json
    default:
      -
        country: Russia
        cities:
          -
            name: Moscow
            population: 9
      -
        country: USA
        cities:
```

```
          -
             name: LA
             population: 5
```

- `list` — dynamic array of arbitrary length. Array elements should be strings. Any element can be added, deleted, or modified on every config update:

```
config:
  mount_points:
    type: list
    default:
      - /dev/rdisk0s1
      - /dev/rdisk0s2
      - /dev/rdisk0s3
```

- `map` — dynamic key/value dictionary of arbitrary length. Keys and values should be strings. Any key or value can be added, deleted, or modified on every config update:

```
config:
  person:
    type: map
    default:
      name: Joe
      age: "24"
      sex: m
```

- `option` — key/value dictionary type. The keys are showed in UI while the corresponding values are stored in DB:

```
config:
  protocol:
    type: option
    option: {http: 80, https: 443}
    default: 80
```

- `password` — same as the `string` type, yet displayed as a password field in UI.
- `secretfile` — obfuscates parameter and its value in UI.
- `secretmap` — obfuscates the value that parameter takes in UI.
- `string` — arbitrary string. The size is not limited. If the default value is not explicitly set, then it will be an empty string `""`.
- `structure` — this type is intended for describing multi-level configurations. The number of levels is unlimited. Configurations are described according to the specification outlined below. Upon each configuration update, new data is compared against the specification. If the input data does not match the specification, ADCM issues an error. The root configuration element, which should be the first one described in the specification, is named `root`. For further attribute description, the `match` element is used, which can be equated to a data type. Possible values for `match` at this level include `dict` and `list`. Depending on the `match` value, attributes of the next level are added — `items` (for `dict`) or `item` (for `list`). Then, the composition of elements included in `item` or `items` is listed.

  You can see available attributes in the table below.

Name	Type	Required	Description
match	string	Yes	Rule according to which every other rule performs its checks
item	string	Yes, for `list` type	Rule according to which every object is checked in case of `list` type
items	key/value pairs	Yes, for `dict` type	Rule according to which every object is checked in case of `dict` type
required_items	list	No	List of mandatory object elements. Only for `dict` type
invisible_items	list	No	Contains elements that can be used for operational purposes and cannot be seen by the user
default_item	string	No	Rule according to which the user-created element in the object is additionally checked. Only for `dict` type

You can view the available `match` values (element types) in the table below.

Type	Description
string	Simple type. Designed for strings
boolean	Simple type. Designed for boolean operations
integer	Simple type. Designed for integers
float	Simple type. Designed for floats
list	Recursive type. Checks itself first according to `item`, then applies rules to all its elements
dict	Recursive type. Checks itself first, then applies rules to its values according to `default_item` and `required_items`

Below is a configuration example.

```
config:
  - name: country_codes:
    type: structure
    yspec: codes/schema.yaml
    default:
      - country: Greece
        code: 30
      - country: France
        code: 33
      - country: Spain
        code: 34
        regions:
          - Andalusia
          - Andalusia
          - Cantabria
          - Catalonia
          - Valencia
```

The *schema.yaml* specification file:

```
---
root:
  match: list
  item: country_code

country_code:
  match: dict
  items:
    country: some_string
    code: some_integer
    regions: list_of_string
    comment: some_string
    uid: some_string
  required_items:
    - code
    - country
  invisible_items:
    - uid

list_of_string:
  match: list
  item: some_string

some_string:
  match: string

some_integer:
  match: int
```

You can use the dot syntax to specify a path to the specification file. In this case, the specification file is expected to be in the same directory as the *config.yaml* file:

```
yspec: ./schema.yaml
```

- `text` — same as `string`, yet is displayed as multiline text in UI.
- `variant` — a type where the user can select a single value from the list of options in UI. The sources for that can be different. You can get a list of variants from another config parameter (the type of this config parameter should be `list`):

```
config:
  - name: mount
    type: variant
    source:
      type: config
      name: mount_points

  - name: mount_points
    type: list
    default:
      - /dev/rdisk0s1
      - /dev/rdisk0s2
      - /dev/rdisk0s3
    read_only: any
    ui_options:
      invisible: true
```

Also, you can get a list of variants from the built-in ADCM functions, such as `cluster_hosts`, that returns the list of all hosts in a current cluster:

```
config:
  - name: cluster_host
    type: variant
    source:
      type: builtin
      name: host_in_cluster
```

Built-in ADCM functions:

- `host_in_cluster` — returns the list of all hosts in the current cluster.
- `host_not_in_clusters` — returns the list of all hosts not included in any cluster.
- `service_in_cluster` — returns the list of all services installed in the current cluster.
- `service_to_add` — returns the list of all services not installed in the current cluster.
- `host` — receives arguments as `predicate` and `args`, where `predicate` is a name of internal host function and `args` are arguments for this function. All internal functions return the list of hosts (the list can be empty):

```
config:
  - name: vhost
    type: variant
    source:
       type: builtin
       name: host
       args:
         predicate: and
         args:
           - predicate: in_service
             args:
               service: UBER
           - predicate: or
             args:
               - predicate: in_component
                 args:
                   service: UBER
                   component: UBER_SERVER
               - predicate: in_cluster
                 args:
```

List of internal host functions:

- `in_cluster` — returns the list of all hosts in the current cluster;

  - `in_service` — returns the list of all hosts in the specified service;
  - `not_in_service` — returns the list of all hosts that are not in the specified service;
  - `in_component` — returns the list of all hosts in the specified component;
  - `not_in_component` — returns the list of all hosts that are not in the specified component;
  - `in_hc` — returns the list of all hosts in a host-component map of this cluster;
  - `not_in_hc` — returns the list of all hosts that are not in a host-component map of this cluster;
  - `and` — returns the union of sets of its arguments;
  - `or` — returns the intersection of sets of its arguments.
    You can list the variants simply in a bundle:

```
config:
  - name: city
    type: variant
    source:
      type: inline
      strict: False
      value:
        - Moscow
        - Voronez
        - Ufa
    default: Samara
```

  You can also specify an optional argument `strict` for the variant source. If `strict` is set to `False`, then users can not only select the value from the predefined list, but also enter their own values in UI.

- `group` — a type that allows you to combine configuration parameters into groups. Group can be activatable. This means that you can switch that group on and off via UI. When a group is switched off (`active = false`), you do not need to fill in its subvalues in UI. Group value in the Ansible inventory file would be null as well.

NOTE

Starting with ADCM 2.9.0, one group can be included in another group. The number of nested groups is unlimited.

```
---
- type: cluster
  name: Kafka
  version: 1

  config:
     - name: grafana
       type: group
       activatable: true
       active: false
       subs:
          - name: endpoint
            type: string
          - name: port
            type: integer
          - name: database
            type: group
            activatable: true
            active: false
            subs:
               - name: user_name
                 type: string
               - name: user_password
                 type: password
```

- `selection_group` — a type that allows you to define mutually exclusive nested groups so that a user can choose one of the nested groups.

IMPORTANT

For this type, first-level child configuration parameters can only be of the `group` type.

```
config:
  - name: catalog.manager
    type: selection_group
    default: static
    subs:
      - name: static
        type: group
        subs:
          - name: "catalog.config-dir"
            type: list
            default: ["/etc/catalog/"]
      - name: dynamic
        type: group
        subs:
          - name: "catalog.store"
            type: string
            default: "file"
```

config_group_customization

Every cluster prototype might have an optional `config_group_customization` property. This property allows the user to customize all cluster (service, component) parameters on some group of hosts. This functionality is disabled by default. The default value is `false`.

```
---
- type: cluster # service component
  name: control
  version: 1

config_group_customization: true
```

If you want to change the behavior of a certain parameter, you can use the `group_customization` option of the corresponding configuration parameter.

description

`description` is a detailed description of your prototype.

```
---
- type: cluster
  name: control
  version: 1
  display_name: "Controlling Software"
  description: |
     This software is intended for monitoring and controlling
     you cluster and does it by ...
```

display_name

`display_name` is a short human-readable name.

```
---
- type: cluster
  name: control
  version: 1
  display_name: "Controlling Software"
  description: |
     This software is intended for monitoring and controlling
     you cluster and does it by ...
```

edition

Every cluster or hostprovider prototype might have an optional `edition` property that corresponds to the bundle type (cluster or host, respectively).

Default value of `edition` is `community`. Value of `edition` can be used in upgrade definition.

```
---
- type: cluster
  name: adh
  display_name: Arenadata Hadoop
  version: 1
  edition: enterprise
```

export

Export is a way to specify settings that a cluster or service is able to transfer to other clusters.

Example

If you want to share some parts of your config with other clusters, you can use the `export` statement. In `export` you can specify any number of the first level `config` entries. You can export a config in a cluster or/and in a service:

```
---
- type: service
  name: Hadoop
  version: 2.1

  config:
     core-site:
        param1:
           type: string
        param2:
           type: integer
     quorum:
        type: integer
        default: 3

  export:
      - core-site
      - quorum
```

flag_autogeneration

NOTE

Some of this attribute's functionality was represented by the `allow_flags` parameter prior to the 2.2.0 release. The `allow_flags` parameter is no longer supported.

This statement is designed for automatic creation of the embedded flags. The statement can be defined on any prototype. The statement can be switched on and off on any prototype as well. If `flag_autogeneration` is defined both on a child object and its parent object, then the child object attribute is of higher priority.

Currently, only the `adcm_outdated_config` embedded flag is supported. This flag is raised automatically when object's configuration is changed. To enable the automatic creation of this embedded flag when the object's configuration changes, set the `enable_outdated_config` parameter to `True`.

```
---
- type: cluster
  name: adpg
  flag_autogeneration:
    enable_outdated_config: True
```

import

Import is a way to specify service settings and versions for a cluster or service to be able to receive them from other clusters.

Example

To use the `config` parts exported from another cluster, use the `import` statement in your cluster. You should specify the name and acceptable versions of an exported cluster (or a service in a cluster):

```
---
-
  type: cluster
  name: Hive
  version: 4

  import:
     Hadoop:
        versions:
           min: 1.8
           max: 2.5
```

When you bind your cluster and another cluster or service via UI, you can use the exported `config` parameters in your Ansible scripts like this:

```
{{  cluster.import.Hadoop.core-site.param2  }}
{{  cluster.import.Hadoop.quorum  }}
```

Optional import parameters

- `required` — the `import` statement might have an optional `required` parameter. If `required` is set to `true` the cluster or service will raise issue and block any action before the required import is properly binded with export cluster or service. The default value for `required` is `false`.
- `default` — the `import` statement might have an optional `default` parameter. The `default` parameter should be an array of config group names of this cluster or service. If this cluster/service is not binded with any export cluster/service import section, the Ansible inventory file will be filled with values from the specified config group. It's recommended to make default group names the same as export group names.
- `multibind` — import statement might have an optional `multibind` parameter. If it is set to `true`, you can bind many instances of a cluster or a service to the same cluster/service. In Ansible inventory file, the import values are represented as an array. The default value of `multibind` is `false`.

license

Every cluster, service, or hostprovider prototype might have an optional `license` property that corresponds to the bundle type. The value of a `license` property is a path to the license file relative to a bundle root.

If a prototype of a cluster or a service has a `license` property, a user of ADCM will be asked to accept a license after the bundle is loaded or the service is added. Before the user accepts a license, any action with the bundle or service is blocked.

```
---
- type: cluster
  name: adh
  display_name: Arenadata Hadoop
  version: 1
  license: misc/license.txt
```

```
---
- type: service
  name: hdfs
  display_name: HDFS
  version: 1
  license: misc/license.txt
```

NOTE

You can use a special dot syntax to specify a path to a license file. In this case, a license file will be searched in the directory that contains *config.yaml*:

```
license: ./license.txt
```

monitoring

Every service or component prototype might have an optional `monitoring` property. It defines if the service/component should be monitored. Default value is `active`. If you want to exclude service/component from a monitoring system, set it to `passive`.

required

Every service prototype might have an optional `required` property. Default value is `false`. If cluster cannot work without this service, set it to `true`. When required service is not added to cluster, cluster will have an issue.

requires

Every service prototype might have an optional `requires` property.

This parameter defines dependencies between the services, i.e. which services need other services or components to be installed simultaneously.

By default, a service does not depend on any other service or component. However, if a dependent service is specified for a service, then when adding that service without the dependent service, an issue is generated in ADCM. Thus, the dependency of a service on another service is checked at the stage of adding it. The dependency of a service on a component is checked at the stage of the host-component mapping being saved.

```
---
- type: service
  name: Hive
  version: 1.0
  requires:
    - service: HDFS
    - service: YARN
      component: TimeLineServer
```

upgrade

Upgrade is an operation available from a particular state of an object (cluster or hostprovider) that updates the metainformation about that object and moves it to another state. There are two types of upgrades:

- Switching the prototypes of the current bundle to the prototypes of the new one.
- Performing some kind of action that may consist of several subactions (including switching the prototypes).

Example

Cluster or host prototype might have an `upgrade` statement. This statement describes how to perform an upgrade from one version of a bundle to another. A prototype might have a few upgrade items to describe upgrades from different versions.

```
---
- type: cluster
  name: control
  version: 2.4
  description: "Monitoring and Control Software"

  upgrade:
      -
         name: Upgrade 1
         versions:
            min: 0.4
            max: 2.0
```

```
        description: New cool upgrade
        states:
           available: any
           on_success: upgradable
     -
        name: Upgrade 2
        versions:
           min: 1.0
           max_strict: 3.0
        scripts:
          - name: pre check
            display_name: Some pre-check before upgrade prototypes
            script_type: ansible
            script: ansible/pre-check.yaml
          - name: bundle upgrade
            display_name: Upgrade bundle
            script_type: internal
            script: bundle_switch
          - name: post tasks
            display_name: Some operation after prototype upgrades
            script_type: ansible
            script: ansible/post-task.yaml
        from_edition: community
        states:
           available:
             - installed
           on_success: upgradable
```

versions

`upgrade` should have a `versions` property. `versions` is the interval of the versions appropriate for upgrade.

```
upgrade:
   -
      name: Upgrade
      versions:
         min: 0.4
         max: 2.0
      states:
         available: any
         on_success: upgradable
```

NOTE

`min` and `max` use non-strict comparison (i.e. boundaries include the version of cluster/host). If you want the comparison to be strict, you can use the `min_strict` and/or the `max_strict` instead of the `min` / `max` .

name

`upgrade` should have a `name` property. `name` is the object name.

display_name

`upgrade` should have a `display_name` property. `display_name` is the object name that is shown to user via UI.

description

`upgrade` might have an optional `description` property. `description` allows you to write description for an upgrade.

states

`upgrade` should have a `states` property. `states` defines two clauses:

- `available` — lists the cluster states that allow upgrade. Special value `any` can be used for upgrade that is available in any state.
- `on_success` — cluster state after upgrade.

```
upgrade:
   -
      name: Upgrade 1
      versions:
         min: 0.4
         max: 2.0
      description: New cool upgrade
      states:
         available: [created, installed]
         on_success: upgradable
```

from_edition

`upgrade` might have an optional `from_edition` property. `from_edition` is the list of editions that allow upgrades.

Special value `any` can be used to specify the permission to upgrade from bundles with any edition. The default value of `from_edition` is `community`.

```
upgrade:
   -
      name: Upgrade 1
      versions:
         min: 0.4
         max: 2.0
      description: New cool upgrade
      states:
         available: [created, installed]
         on_success: upgradable
      from_edition:
         - community
         - enterprise
```

venv

Every cluster, service, component, hostprovider, or host prototype might have an optional `venv` property.

It defines the Ansible version that will be used when launching an action of the corresponding prototype.

Valid values:

- `default` — use Ansible 2.8 when launching an action;
- `2.9` — use Ansible 2.9 when launching an action;
- `2.16` — use Ansible 2.16 when launching an action (starting with ADCM 2.8.0).

The default value is `default`.

NOTE

With the release of ADCM 2.8.0, Ansible 2.8 is deprecated and will be removed from ADCM in Q3 2026.

```
---
- type: cluster # service component hostprovider host
  name: control
  version: 1

  venv: "2.9"
```

Create a simple bundle

Mikhail Serov

Contents

To Table of Contents

The config.yaml file
  Sections of the config.yaml file
Example

Creating a simple bundle for ADCM consists of preparing Ansible playbooks and configuration file in the YAML format (on this page it will also be called **config file**).

- Ansible playbooks are necessary to specify and run actions that will be performed at hosts.
- Config file is necessary to declare and specify the objects the bundle will contain. You can create a common configuration file for all objects or you can create separate configuration files for different objects. Configuration file should be named *config.yaml* or *config.yml*. Configuration files can be located inside the root directory, or inside a directory that you set.

The prepared Ansible playbooks and configuration files are archived in a file of .tgz format that is, essentially, an ADCM bundle.

The *config.yaml* file structure is given below.

The config.yaml file

ADCM bundles may be of two types: the cluster bundle or the infrastructure bundle.

- Configuration file for a cluster bundle may contain sections that specify clusters, services, and their components.
- Configuration file for an infrastructure bundle may contain sections that specify hostproviders and hosts.

> IMPORTANT
>
> The configuration file for a bundle may contain descriptions only for those objects that correspond to the bundle type. For example, the configuration file for a cluster bundle can contain descriptions for clusters and services while it cannot contain descriptions for hosts.

The main sections of the *config.yaml* file are listed below.

Sections of the config.yaml file

The configuration file for a cluster bundle may contain the following sections:

- `type` — the object type. Takes the `cluster` or `service` values in case of a cluster bundle and the `hostprovider` or `host` values in case of an infrastructure bundle.
- `name` — the object name.
- `description` — the object description.
- `version` — the object (cluster, service, or hostprovider) version. Is not applicable for components and hosts.
- `edition` — the object (cluster or hostprovider) edition.
- `license` — the path to the license agreement file.
- `adcm_min_version` — the minimum ADCM version the bundle works on.
- `upgrade` — the operation available from an object's (cluster or hostprovider) certain state that updates object's metainformation and moves it to a different state (with possible restrictions regarding the `version` and `edition` values). Product version may contain several product editions.
- `config` — the description of a configuration of an object (cluster, service, component, host, or hostprovider). After bundle upload and object creation, the object inside the ADCM interface will display the settings (on the ***Settings*** page) that were specified in this section.
- `actions` — the sets of actions that can be performed with an object (cluster, service, component, host, or hostprovider). Those actions are expressed as Ansible playbooks. The display of actions for an object in the ADCM interface depends on the object state.
- `import` — the settings of services that an object (cluster or service) can accept from another cluster's objects. For example, a cluster may accept the settings of the ZooKeeper service from another cluster (only the service settings are transmitted, the service itself is not being physically added). This functionality is essentially needed for the settings exchange between different clusters.
- `export` — the settings of services that an object (cluster or service) can transmit to another cluster's objects. This functionality is essentially needed for the settings exchange between different clusters.

Example

Below you can find an example of the *config.yaml* file for a ZooKeeper cluster bundle. The file contains the sections discussed above.

```yaml
---
- type: cluster
  name: Zookeeper cluster
  description: Zookeeper cluster software
  version: 2.0.0
  edition: community
  license: EULA.txt
  adcm_min_version: 2022.10.04.17

  upgrade:
    - name: 2.0.0 community
      versions:
        min: "1.5.0"
        max_strict: "1.9.9"
      from_edition: ['community']
      states:
        available: any
        on_success: upgrade from 1.5.X

  config:
    - name: centos7_repos
      display_name: "Centos 7 repositories"
      type: group
      subs:
        - name: Zookeeper
          type: map
          description: Arenadata Zookeeper yum repository
          default:
            desc: Arenadata Zookeeper yum repository
            url: http://downloads.arenadata.io/zookeeper/3.5.8/centos/7/community/$basearch/
            enabled: "True"

  actions:
    Install:
      display_name: Install
      type: task
      scripts:
        - name: preinstall
          display_name: "Preinstall"
          script_type: ansible
          script: ansible/cluster_preinstall.yaml
          params:
            ansible_tags: preinstall, service_dependency_check
        - name: zookeeper
          display_name: "Zookeeper"
          script_type: ansible
          script: adcm_service_zookeeper/ansible/zookeeper.yml
          params:
            ansible_tags: install, configure, start
      states:
        available:
          - created
        on_success: installed


- type: service
  name: zookeeper
  display_name: Zookeeper
  version:

  export:
    - Main

  config:
    - name: Main
      type: group
      subs:
        - name: connect
          type: string
          read_only: any
          required: no
          description: >
            Zookeeper connection string that is used by other services or clusters.
            It is generated automatically.
        - name: data_dirs
          display_name: dataDir
          type: string
          default: "/var/lib/zookeeper"
          description: >
            The location where Zookeeper stores the in-memory database snapshots and, unless specified otherwise, the transaction
log of updates to the database.
            Specified as the dataDir parameter in the zoo.cfg configuration file.

  actions:
    Start:
      type: job
      script_type: ansible
      script: ./ansible/zookeeper.yml
      params:
        ansible_tags: configure, configure_main_info, start
      states:
        available:
          - installed

  components:
    SERVER:
      display_name: Zookeeper Server
      constraint: [odd]
      actions:
        Start:
          type: job
          script_type: ansible
          script: ./ansible/zookeeper_server.yml
          params:
            ansible_tags: configure, configure_main_info, start
          states:
            available:
              - installed
```

ADCM API

Mikhail Serov

To Table of Contents

IMPORTANT

API v1 release is now deprecated. Starting with ADCM 2.8.0, API v1 was removed from ADCM and is no longer supported.

You can view the generated documentation on ADCM API v2 by clicking the ? icon in the top right corner on any page of the ADCM web interface and selecting ***ADCM API*** from the drop-down list.

OpenAPI 3 specification can be downloaded at *http://<IP address>:8000/api/v2/schema/*.

For more information and examples, please refer to the documentation.

To use ADCM API v2, you need an access token. It is a digital key that allows users to access functions of ADCM API v2. To get the access token, send the POST request to the host where the ADCM container is started.

To access protected endpoints of ADCM API v2, pass the received access token in the header of the request to the required endpoint.

The following is an example of the GET request that specifies the access token to get a list of all clusters with the `curl` command:

```
$ curl --request GET \
--url http://10.92.40.154:8000/api/v2/clusters/ \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--header 'Authorization: Token 884ecc261b81563811451ed01dd45859998d893a'
```

ADCM API v2 usage examples

Mikhail Serov, Adelya Zvyagintseva

Contents

To Table of Contents

- Get an access token
  - Example request
  - Example response
- Upload a bundle
  - Example request
  - Example response
- Accept a license agreement
  - Example request
  - Example response
- List prototypes
  - Example request
  - Example response
- Create a hostprovider
  - Example request
  - Example response
- List hostproviders
  - Example request
  - Example response
- Create a host
  - Example request
  - Example response
- List hosts
  - Example request
  - Example response
- Delete a host
  - Example request
  - Example response
- Create a cluster
  - Example request
  - Example response
- Change a name and description of a cluster
  - Example request
  - Example response
- List clusters
  - Example request
  - Example response
- Delete a cluster
  - Example request
  - Example response
- Add a service

  - Example request
  - Example response
- List services
  - Example request
  - Example response
- Delete a service
  - Example request
  - Example response
- Add a host to a cluster
  - Example request
  - Example response
- Remove a host from a cluster
  - Example request
  - Example response
- List versions of a configuration
  - Example request
  - Example response
- Create a configuration version
  - Example request
  - Example response
- Map components to a host
  - Example request
  - Example response
- List the current mapping of components to hosts
  - Example request
  - Example response
- List actions
  - Example request
  - Example response
- Launch actions
  - Example request
  - Example response
- List host groups for actions
  - Example request
  - Example response
- Create a host group for actions
  - Example request
  - Example response
- Add a host to a group for actions
  - Example request
  - Example response
- Remove a host from a group for actions
  - Example request
- Launch actions for host groups
  - Example request
  - Example response

- Delete a non-blocking concern
  - Example request
  - Example response
- Delete a host group for actions
  - Example request
- List upgrades
  - Example request
  - Example response
- Launch upgrades
  - Example request
  - Example response

This page lists some of the HTTP requests that are supported by ADCM API v2.

If the request returns a paginated list, the response will contain the parameters described below.

Response JSON object parameters for pagination

Parameter	Type	Description
count	integer	Number of elements
next	integer	Next page
previous	integer	Previous page
results	array of objects	A list containing paginated information about objects according to the performed request

Get an access token

```
POST /api/v2/token/ HTTP/1.1
```

This request generates an access token that allows the user to access protected endpoints of ADCM API v2.

Request JSON object parameters

Parameter	Type	Description
username	string	User account name
password	string	User account password

Response JSON object parameters

Parameter	Type	Description
token	string	Access token

Example request

```
POST /api/v2/token/ HTTP/1.1
Accept: application/json

{
  "username": "admin",
  "password": "admin"
}
```

Example response

```
HTTP 200 OK
Allow: POST
Content-Type: application/json
Vary: Accept

{
  "token": "884ecc261b81563811451ed01dd45859998d893a"
}
```

Upload a bundle

```
POST /api/v2/bundles/ HTTP/1.1
```

This request uploads a new bundle to ADCM.

Request JSON object parameters

Parameter	Type	Description
file	string	Bundle file

Response JSON object parameters

Parameter	Type	Description
id	integer	Bundle ID
name	string	Bundle name
displayName	string	Displayed bundle name
version	string	Bundle version
edition	string	Bundle edition
mainPrototype	object	Bundle prototype object
uploadTime	string	Bundle upload time
signatureStatus	string	Bundle signature status. Possible values: `valid`, `invalid`, `absent`

Example request

```
POST /api/v2/bundles/ HTTP/1.1
Accept: application/json
Content-Type: multipart/form-data

{
  "file": (binary)
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 21,
    "name": "SSH Common",
    "displayName": "SSH Common",
    "version": "2.9-dev",
    "edition": "community",
    "mainPrototype": {
        "id": 437,
        "name": "SSH Common",
        "displayName": "SSH Common",
        "description": "Simple ssh hosts",
        "type": "provider",
        "license": {
            "status": "absent",
            "text": null
        },
        "version": "2.9-dev"
    },
    "uploadTime": "2023-12-06T12:20:09.553497Z",
    "signatureStatus": "absent"
}
```

Accept a license agreement

This request accepts the license agreement for the specified prototype.

```
POST /api/v2/prototypes/<prototypeId>/license/accept/ HTTP/1.1
```

Example request

```
POST /api/v2/prototypes/5/license/accept/ HTTP/1.1
```

Example response

```
HTTP 200 OK
```

List prototypes

```
GET /api/v2/prototypes/ HTTP/1.1
```

This request lists all prototypes in ADCM. Add the ADCM object type at the end of the request to see only the prototypes of that type. The response supports pagination.

Response JSON object parameters

Parameter	Type	Description
id	integer	Prototype ID
name	string	Prototype name
displayName	string	Displayed prototype name
description	string	Prototype description
type	string	Prototype type
bundle	object	Bundle object

license	object	Prototype license object
version	string	Prototype version

Example request

```
GET /api/v2/prototypes/?type=provider HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 331,
            "name": "Yandex Compute VM Provider",
            "displayName": "Yandex Compute VM Provider",
            "description": "Manage Yandex Compute Cloud VMs",
            "type": "provider",
            "bundle": {
                "id": 11,
                "edition": "community"
            },
            "license": {
                "status": "absent",
                "text": null
            },
            "version": "3.4-1"
        }
    ]
}
```

Create a hostprovider

```
POST /api/v2/hostproviders/ HTTP/1.1
```

This request creates a new hostprovider. You can see all hostprovider prototype IDs at *http://<IP address>:8000/api/v2/prototypes/?type=provider*, using the IP address of the host where the ADCM container is started.

Request JSON object parameters

Parameter	Type	Description
prototypeId	integer	ID of a prototype
name	string	Hostprovider name
description	string	Hostprovider description

Response JSON object parameters

Parameter	Type	Description
id	integer	Hostprovider ID
name	string	Hostprovider name

state	string	Hostprovider state
multiState	array of strings	List of extended hostprovider states
prototype	object	Hostprovider prototype object
description	string	Hostprovider description
concerns	array of objects	List of points that require attention or action
isUpgradable	boolean	Sign of the presence (availability) of a hostprovider upgrade
mainInfo	string	Hostprovider main info

Example request

```
POST /api/v2/hostproviders/ HTTP/1.1
Accept: application/json

{
  "prototypeId": 132,
  "name": "new test hostprovider",
  "description": ""
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST
Content-Type: application/json
Vary: Accept

{
  "id": 2,
  "name": "new test hostprovider",
  "state": "created",
  "multiState": [],
  "prototype": {
      "id": 132
      "name": "VMware vCloud Director",
      "displayName": "VMware vCloud Director",
      "version": "1.1.0-1"
  },
  "description": "",
  "concerns": [],
  "isUpgradable": false,
  "mainInfo": null
}
```

List hostproviders

```
GET /api/v2/hostproviders/ HTTP/1.1
```

This request lists all hostproviders that currently exist in ADCM. The response supports pagination.

JSON object results parameters

Parameter	Type	Description
id	integer	Hostprovider ID
name	string	Hostprovider name
state	string	Hostprovider state

multiState	array of strings	List of extended hostprovider states
prototype	object	Hostprovider prototype object
description	string	Hostprovider description
concerns	array of objects	List of points that require attention or action
isUpgradable	boolean	Sign of the presence (availability) of a hostprovider upgrade
mainInfo	string	Hostprovider main info

Example request

```
GET /api/v2/hostproviders/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 4,
            "name": "test hostprovider",
            "state": "created",
            "multiState": [],
            "prototype": {
                "id": 331,
                "name": "Yandex Compute VM Provider",
                "displayName": "Yandex Compute VM Provider",
                "version": "3.4-1"
            },
            "description": "",
            "concerns": [],
            "isUpgradable": false,
            "mainInfo": null
        }
    ]
}
```

Create a host

```
POST /api/v2/hosts/ HTTP/1.1
```

This request creates a new host.

Request JSON object parameters

Parameter	Type	Description
hostproviderId	integer	Hostprovider ID
name	string	Host name
clusterId	integer	Cluster ID

Response JSON object parameters

Parameter	Type	Description
id	integer	Host ID
name	string	Host name
description	string	Host description
state	string	Host state
status	string	Host status. Possible values: `up`, `down`
hostprovider	object	Hostprovider object
prototype	object	Prototype object
concerns	array of objects	List of points that require attention or action
isMaintenanceModeAvailable	boolean	Maintenance mode availability for a cluster host
maintenanceMode	string	Maintenance mode status. Possible values: `on`, `off`
multiState	array of strings	List of extended host states
cluster	object	Cluster object
components	array of objects	List of host component objects

Example request

```
POST /api/v2/hosts/ HTTP/1.1
Accept: application/json

{
  "name": "test",
  "hostproviderId": 4,
  "clusterId": 9
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST
Content-Type: application/json
Vary: Accept

{
    "id": 8,
    "name": "test",
    "description": "",
    "state": "created",
    "status": "down",
    "hostprovider": {
        "id": 4,
        "name": "test hostprovider",
        "displayName": "test hostprovider"
    },
    "prototype": {
        "id": 332,
        "name": "Yandex Compute VM",
        "displayName": "Yandex Compute VM",
        "version": "3.4-1"
    },
    "concerns": [],
```

```
    "isMaintenanceModeAvailable": false,
    "maintenanceMode": "off",
    "multiState": [],
    "cluster": {
        "id": 9,
        "name": "ADH"
    },
    "components": []
}
```

List hosts

```
GET /api/v2/hosts/ HTTP/1.1
```

This request lists all hosts that currently exist in ADCM. If you wish to see detailed information about a particular host, add the host ID to the request:

```
GET /api/v2/hosts/<hostId>/ HTTP/1.1
```

The response supports pagination.

JSON object results parameters

Parameter	Type	Description
id	integer	Host ID
name	string	Host name
description	string	Host description
state	string	Host state
status	string	Host status. Possible values: `up`, `down`
hostprovider	object	Hostprovider object
prototype	object	Prototype object
concerns	array of objects	List of points that require attention or action
isMaintenanceModeAvailable	boolean	Maintenance mode availability for a cluster host
maintenanceMode	string	Maintenance mode status. Possible values: `on`, `off`
multiState	array of strings	List of extended host states
cluster	object	Cluster object
components	array of objects	List of component objects

Example request

```
GET /api/v2/hosts/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 8,
            "name": "bds-adbc",
            "description": "",
            "state": "running",
            "status": "down",
            "hostprovider": {
                "id": 2,
                "name": "Yandex",
                "displayName": "Yandex"
            },
            "prototype": {
                "id": 26,
                "name": "Yandex Compute VM",
                "displayName": "Yandex Compute VM",
                "version": "3.7-1"
            },
            "concerns": [],
            "isMaintenanceModeAvailable": false,
            "maintenanceMode": "off",
            "multiState": [],
            "cluster": {
                "id": 2,
                "name": "Test ADB cluster"
            },
            "components": [
                {
                    "id": 18,
                    "name": "server",
                    "displayName": "ADBM server"
                },
                {
                    "id": 17,
                    "name": "server",
                    "displayName": "ADB Control server"
                }
            ]
        }
    ]
}
```

Delete a host

This request deletes the specified host. You can only delete a host that does not belong to any cluster.

```
DELETE /api/v2/hosts/<hostId>/ HTTP/1.1
```

Example request

```
DELETE /api/v2/hosts/1/ HTTP/1.1
```

Example response

```
HTTP 204 No Content
```

Create a cluster

```
POST /api/v2/clusters/ HTTP/1.1
```

This request creates a new cluster. You can see all cluster prototype IDs at *http://<IP address>:8000/api/v2/prototypes/?type=cluster*, using the IP address of the host where the ADCM container is started.

Request JSON object parameters

Parameter	Type	Description
prototypeId	integer	ID of a prototype
name	string	Cluster name
description	string	Cluster description

Response JSON object parameters

Parameter	Type	Description
id	integer	Cluster ID
name	string	Cluster name
description	string	Cluster description
state	string	Cluster state
multiState	array of strings	List of extended cluster states
status	string	Cluster status. Possible values: `up`, `down`
prototype	object	Cluster prototype object
concerns	array of objects	List of points that require attention or action
isUpgradable	boolean	Sign of the presence (availability) of a cluster upgrade
mainInfo	string	Cluster main info

Example request

```
POST /api/v2/clusters/ HTTP/1.1
Accept: application/json

{
  "name": "test",
  "prototypeId": 264,
  "description": "test cluster"
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST
Content-Type: application/json
Vary: Accept

{
    "id": 10,
    "name": "test",
    "description": "test cluster",
    "state": "created",
    "multiState": [],
    "status": "down",
    "prototype": {
        "id": 264,
        "name": "hadoop",
        "displayName": "Hadoop",
        "version": "2.1.10_b1-1"
    },
    "concerns": [],
    "isUpgradable": false,
    "mainInfo": "<p>Documentation: <a href=https://docs.arenadata.io/adh/>https://docs.arenadata.io/adh/</a></p>\n"
}
```

Change a name and description of a cluster

```
PATCH /api/v2/clusters/<clusterId>/ HTTP/1.1
```

This request allows you to change the name and description of the specified cluster.

Request JSON object parameters

Parameter	Type	Description
name	string	Cluster name. String length limit: 2—150 characters
description	string	Cluster description

Response JSON object parameters

Parameter	Type	Description
id	integer	Cluster ID
name	string	Cluster name. String length limit: 1000 characters
description	string	Cluster description
state	string	Cluster state. String length limit: 1000 characters
multiState	array of strings	List of extended cluster states
status	string	Cluster status. Possible values: `up`, `down`
prototype	object	Cluster prototype object
concerns	array of objects	List of points that require attention or action
isUpgradable	boolean	Sign of the presence (availability) of a cluster upgrade
mainInfo	string	Cluster main info

Example request

```
PATCH /api/v2/clusters/73/ HTTP/1.1
Accept: application/json

{
  "name": "Test ADB cluster",
  "description": "Example"
}
```

Example response

```
HTTP 200 OK
Allow: GET, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 73,
    "name": "Test ADB cluster",
    "description": "Example",
    "state": "created",
    "multiState": [],
    "status": "down",
    "prototype": {
        "id": 47,
        "name": "ADB",
        "displayName": "ADB",
        "version": "6.26.0_arenadata53_b1-1"
    },
    "concerns": [
        {
            "id": 1845,
            "type": "issue",
            "reason": {
                "message": "${source} has an issue with host-component mapping",
                "placeholder": {
                    "source": {
                        "name": "Test ADB cluster",
                        "type": "cluster_mapping",
                        "params": {
                            "clusterId": 73
                        }
                    }
                }
            },
            "isBlocking": true,
            "cause": "host-component",
            "owner": {
                "id": 73,
                "type": "cluster"
            }
        }
    ],
    "isUpgradable": false,
    "mainInfo": "The main goal of ADB bundle is an easy and fast installation and managing of Arenadata Database with Arenadata Cluster Manager. ADB bundle consists of ADB, time synchronization and the monitoring client services. These services can be installed either on cloud or bare metal hosts (depending on host bundles you choose). After installation Arenadatabase cluster will be available for psql connect on port 5432 of the master host address.\n"
}
```

List clusters

```
GET /api/v2/clusters/ HTTP/1.1
```

This request lists all clusters that currently exist in ADCM. If you wish to see detailed information about a particular cluster, add the cluster ID to the request:

```
GET /api/v2/clusters/<clusterId>/ HTTP/1.1
```

The response supports pagination.

JSON object results parameters

Parameter	Type	Description
id	integer	Cluster ID
name	string	Cluster name
description	string	Cluster description
state	string	Cluster state
multiState	array of strings	List of extended cluster states
status	string	Cluster status. Possible values: `up`, `down`
prototype	object	Cluster prototype object
concerns	array of objects	List of points that require attention or action
isUpgradable	boolean	Sign of the presence (availability) of a cluster upgrade
mainInfo	string	Cluster main info

Example request

```
GET /api/v2/clusters/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 9,
            "name": "ADH",
            "description": "",
            "state": "created",
            "multiState": [],
            "status": "down",
            "prototype": {
                "id": 264,
                "name": "hadoop",
                "displayName": "Hadoop",
                "version": "2.1.10_b1-1"
            },
            "concerns": [],
            "isUpgradable": false,
            "mainInfo": "<p>Documentation: <a href=https://docs.arenadata.io/adh/>https://docs.arenadata.io/adh/</a></p>\n"
        },
        {
            "id": 10,
            "name": "test",
            "description": "test cluster",
            "state": "created",
            "multiState": [],
            "status": "down",
            "prototype": {
                "id": 264,
                "name": "hadoop",
```

```
                "displayName": "Hadoop",
                "version": "2.1.10_b1-1"
            },
            "concerns": [],
            "isUpgradable": false,
            "mainInfo": "<p>Documentation: <a href=https://docs.arenadata.io/adh/>https://docs.arenadata.io/adh/</a></p>\n"
        }
    ]
}
```

Delete a cluster

This request deletes the specified cluster.

```
DELETE /api/v2/clusters/<clusterId>/ HTTP/1.1
```

Example request

```
DELETE /api/v2/clusters/3/ HTTP/1.1
```

Example response

```
HTTP 204 No Content
```

Add a service

```
POST /api/v2/clusters/<clusterId>/services/ HTTP/1.1
```

This request adds a service to the specified cluster. Service prototype ID is required. You can see all service prototype IDs at *http://<IP address>:8000/api/v2/prototypes/?type=service*, using the IP address of the host where the ADCM container is started.

Request JSON object parameters

Parameter	Type	Description
prototypeId	integer	Service prototype ID

Response JSON object parameters

Parameter	Type	Description
id	integer	Service ID
name	string	Service name
description	string	Service description
displayName	string	Displayed service name
prototype	object	Service prototype object
cluster	object	Cluster object
status	string	Service status
state	string	Service state
multiState	array of strings	List of extended service states

concerns	array of objects	List of points that require attention or action
isMaintenanceModeAvailable	boolean	Availability of the maintenance mode for the service
maintenanceMode	string	Maintenance mode status. Possible values: `on`, `off`
mainInfo	string	Service main info

Example request

```
POST /api/v2/clusters/2/services/ HTTP/1.1
Accept: application/json

[
    {
        "prototypeId": 278
    }
]
```

Example response

```
HTTP 201 Created
Allow: GET, POST
Content-Type: application/json
Vary: Accept

[
    {
        "id": 15,
        "name": "zookeeper",
        "description": "",
        "displayName": "Zookeeper",
        "prototype": {
            "id": 278,
            "name": "zookeeper",
            "displayName": "Zookeeper",
            "version": "3.5.10"
        },
        "cluster": {
            "id": 10,
            "name": "test"
        },
        "status": "up",
        "state": "created",
        "multiState": [],
        "concerns": [
            {
                "id": 64,
                "reason": {
                    "message": "${source} has an issue with host-component mapping",
                    "placeholder": {
                        "source": {
                            "type": "cluster",
                            "name": "test",
                            "params": {
                                "clusterId": 10
                            }
                        }
                    }
                },
                "isBlocking": true,
                "cause": "host-component"
            }
        ],
        "isMaintenanceModeAvailable": false,
        "maintenanceMode": "off",
        "mainInfo": null
    }
]
```

List services

```
GET /api/v2/clusters/<clusterId>/services/ HTTP/1.1
```

This request lists all services that are added to a specified cluster. If you wish to see detailed information about a particular service, add the service ID to the request:

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/ HTTP/1.1
```

The response supports pagination.

JSON object results parameters

Parameter	Type	Description
id	integer	Service ID
name	string	Service name
description	string	Service description
displayName	string	Displayed service name
prototype	object	Service prototype object
cluster	object	Cluster object
status	string	Service status
state	string	Service state
multiState	array of strings	List of extended service states
concerns	array of objects	List of points that require attention or action
isMaintenanceModeAvailable	boolean	Availability of the maintenance mode for the service
maintenanceMode	string	Maintenance mode status. Possible values: `on`, `off`
mainInfo	string	Service main info

Example request

```
GET /api/v2/clusters/2/services/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 15,
            "name": "zookeeper",
            "description": "",
            "displayName": "Zookeeper",
            "prototype": {
```

```
                "id": 278,
                "name": "zookeeper",
                "displayName": "Zookeeper",
                "version": "3.5.10"
            },
            "cluster": {
                "id": 10,
                "name": "test"
            },
            "status": "up",
            "state": "created",
            "multiState": [],
            "concerns": [
                {
                    "id": 64,
                    "reason": {
                        "message": "${source} has an issue with host-component mapping",
                        "placeholder": {
                            "source": {
                                "type": "cluster",
                                "name": "test",
                                "params": {
                                    "clusterId": 10
                                }
                            }
                        }
                    },
                    "isBlocking": true,
                    "cause": "host-component"
                }
            ],
            "isMaintenanceModeAvailable": false,
            "maintenanceMode": "off",
            "mainInfo": "<div style=\"width:600px\">\n<p>Centralized service for maintaining configuration\ninformation, naming, providing distributed synchronization,\nand providing group services.</p>\n</div>\n"
        }
    ]
}
```

Delete a service

This request deletes a service from the specified cluster. A service for whose components the mapping to hosts exists is not always possible to remove.

```
DELETE /api/v2/clusters/<clusterId>/services/<serviceId>/ HTTP/1.1
```

Example request

```
DELETE /api/v2/clusters/3/services/1/ HTTP/1.1
```

Example response

```
HTTP 204 No Content
```

Add a host to a cluster

```
POST /api/v2/clusters/<clusterId>/hosts/ HTTP/1.1
```

This request adds a host to the specified cluster. A host ID is required.

Request JSON object parameters

Parameter	Type	Description
hostId	integer	Host ID

Response JSON object parameters

Parameter	Type	Description
id	integer	Host ID
name	string	Host name
description	string	Host description
state	string	Host state
status	string	Host status. Possible values: `up`, `down`
hostprovider	object	Hostprovider object
prototype	object	Host prototype object
concerns	array of objects	List of points that require attention or action
isMaintenanceModeAvailable	boolean	Availability of the maintenance mode for the host
maintenanceMode	string	Maintenance mode status. Possible values: `on`, `off`
multiState	array of strings	List of extended host states
cluster	object	Cluster object
components	array of objects	List of component objects

Example request

```
POST /api/v2/clusters/10/hosts/ HTTP/1.1
Accept: application/json

[
    {
    "hostId": 8
    }
]
```

Example response

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 8,
        "name": "test",
        "description": "",
        "state": "created",
        "status": "down",
        "hostprovider": {
            "id": 4,
            "name": "test hostprovider",
            "displayName": "test hostprovider"
        },
        "prototype": {
            "id": 332,
            "name": "Yandex Compute VM",
            "displayName": "Yandex Compute VM",
            "version": "3.4-1"
```

```
        },
        "concerns": [],
        "isMaintenanceModeAvailable": false,
        "maintenanceMode": "off",
        "multiState": [],
        "cluster": {
            "id": 10,
            "name": "test"
        },
        "components": []
    }
]
```

Remove a host from a cluster

```
DELETE /api/v2/clusters/<clusterId>/hosts/<hostId>/ HTTP/1.1
```

This request unlinks the specified host from the specified cluster.

Example request

```
DELETE /api/v2/clusters/3/hosts/6/ HTTP/1.1
```

Example response

```
HTTP 204 No Content
```

List versions of a configuration

```
GET /api/v2/hostproviders/<hostproviderId>/configs/ HTTP/1.1
```

```
GET /api/v2/hosts/<hostId>/configs/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/configs/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/configs/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/configs/ HTTP/1.1
```

These requests list the configuration versions of a hostprovider, host, cluster, service, or component, respectively. To view detailed information about an object configuration (list of configuration attributes and their values), specify the configuration version ID in the query, for example (for a cluster):

```
GET /api/v2/clusters/<clusterId>/configs/<configId>/ HTTP/1.1
```

The response supports pagination.

JSON object results parameters

Parameter	Type	Description
id	integer	Configuration version ID
isCurrent	boolean	Defines whether the configuration version is current
creationTime	datetime	Date and time the configuration version was created
description	string	Configuration version description

Example request

```
GET /api/v2/clusters/27/configs/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 3,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1349,
            "isCurrent": true,
            "creationTime": "2025-08-01T12:02:34.188737Z",
            "description": "upgrade"
        },
        {
            "id": 1338,
            "isCurrent": false,
            "creationTime": "2025-07-28T16:21:12.037662Z",
            "description": "ansible update"
        },
        {
            "id": 1332,
            "isCurrent": false,
            "creationTime": "2025-07-28T16:06:40.193811Z",
            "description": "init"
        }
    ]
}
```

Create a configuration version

```
POST /api/v2/hostproviders/<hostproviderId>/configs/ HTTP/1.1
```

```
POST /api/v2/hosts/<hostId>/configs/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/configs/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/configs/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/configs/ HTTP/1.1
```

These requests create a new configuration version of a hostprovider, host, cluster, service, or component, respectively.

Request JSON object parameters

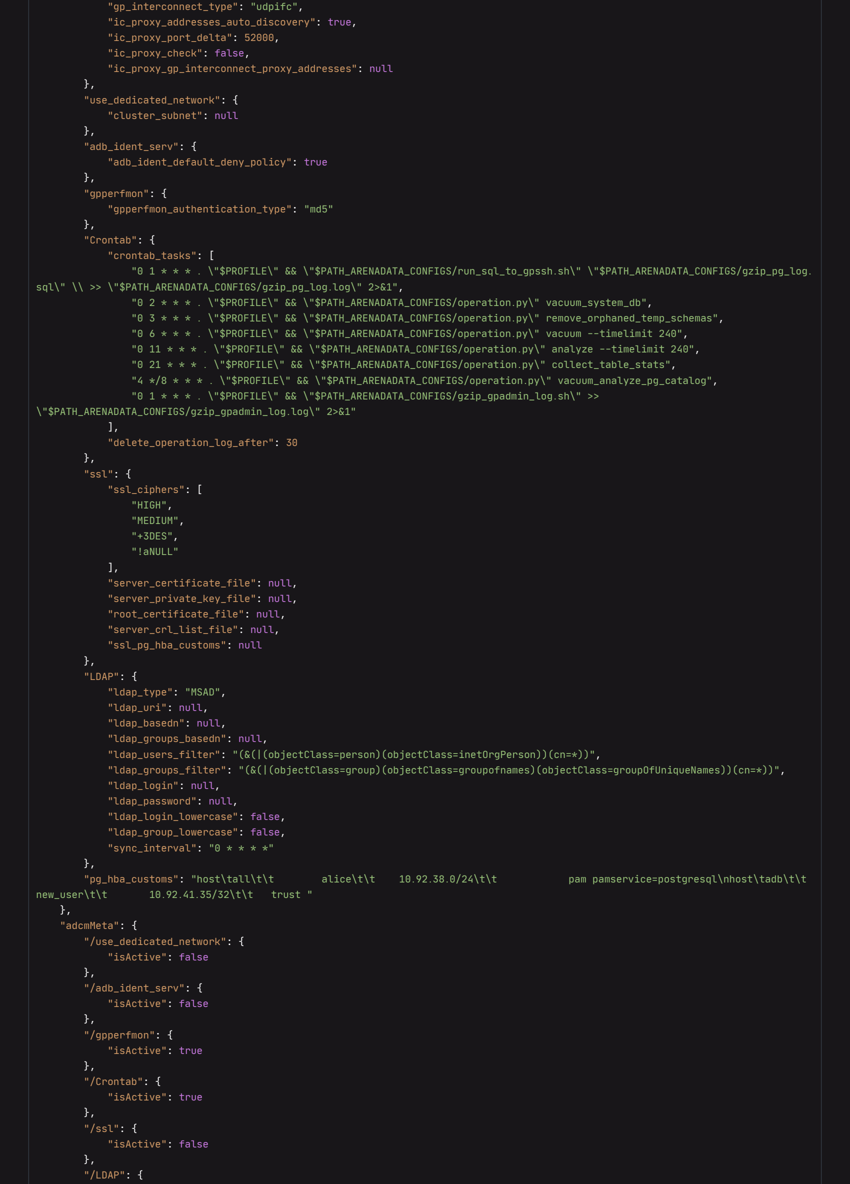
Parameter	Type	Description
config	object	List of configuration attributes and their values
adcmMeta	object	Ability to activate or deactivate a section (configuration attributes) in the configuration tree
description	string	Configuration version description

Response JSON object parameters

Parameter	Type	Description
id	integer	Configuration version ID
isCurrent	boolean	Defines whether the configuration version is current
creationTime	datetime	Date and time the configuration version was created
config	object	List of configuration attributes and their values
adcmMeta	object	Ability to activate or deactivate a section (configuration attributes) in the configuration tree
description	string	Configuration version description

Example request

```
POST /api/v2/clusters/27/services/105/configs/ HTTP/1.1
Accept: application/json

{
  "config": {
        "__main_info": "<p style=\"font-size:16px; \">\n<a href=\"https://arenadata.tech/en/products/arenadata-db/\"
target=\"_blank\">\nArenadata DB (ADB)</a> is a distributed database based on using MPP\n(massively parallel processing)
principle. It takes its origin\nfrom Pivotal's Greenplum database.</p><p style=\"font-size:16px; \">JDBC URL:
jdbc:postgresql://10.92.38.37:5432</p>\n<p style=\"font-size:16px; \">SSL: disabled</p>\n",
        "components": {
            "adb": "{\"logs\": [{\"type\": \"search_and_archive_directories\", \"paths\": [\"*/master/gpseg*/pg_log\",
\"*/mirror/gpseg*/pg_log\", \"*/primary/gpseg*/pg_log\", \"/home/*/gpAdminLogs\", \"/home/*/arenadata_configs/operation_log\"]}]}"
        },
        "Main": {
            "dbname": "adb",
            "enable_mirroring": true,
            "madlib": false,
            "postgis": false,
            "autocoredump": false,
            "diskquota": false,
            "plpython3": false
        },
        "Advanced": {
            "num_segments_per_host": 4,
            "segment_datadirs": {
                "/data1": ""
            },
            "segment_nobarrier": false,
            "master_datadir": "/data1",
            "master_mount_storage": false,
            "master_server_storage": null,
            "master_nobarrier": false,
            "config_dir": "arenadata_configs",
            "sysctl_params": "[[\"kernel.core_uses_pid\", \"1\"], [\"kernel.shmmni\", \"4096\"], [\"kernel.sem\", \"250 2048000 200
8192\"], [\"kernel.msgmnb\", \"65536\"], [\"kernel.msgmax\", \"65536\"], [\"kernel.msgmni\", \"2048\"], [\"net.ipv4.
tcp_syncookies\", \"1\"], [\"net.ipv4.conf.default.accept_source_route\", \"0\"], [\"net.ipv4.tcp_max_syn_backlog\", \"4096\"],
[\"net.ipv4.conf.all.arp_filter\", \"1\"], [\"net.ipv4.ip_local_port_range\", \"10000 65535\"], [\"net.ipv4.ipfrag_high_thresh\",
\"41943040\"], [\"net.ipv4.ipfrag_low_thresh\", \"31457280\"], [\"net.ipv4.ipfrag_time\", \"60\"], [\"net.core.
netdev_max_backlog\", \"10000\"], [\"net.core.rmem_max\", \"2097152\"], [\"net.core.wmem_max\", \"2097152\"], [\"vm.
overcommit_memory\", \"2\"], [\"vm.overcommit_ratio\", \"95\"], [\"vm.swappiness\", \"10\"], [\"vm.zone_reclaim_mode\", \"0\"],
[\"vm.dirty_expire_centisecs\", \"500\"], [\"vm.dirty_writeback_centisecs\", \"100\"], [\"vm.dirty_background_ratio\", \"0\"],
[\"vm.dirty_ratio\", \"0\"], [\"vm.dirty_background_bytes\", \"1610612736\"], [\"vm.dirty_bytes\", \"4294967296\"], [\"kernel.
core_pipe_limit\", \"0\"]]",
            "system_user_gid": 2042,
            "system_user_name": "gpadmin",
            "system_user_uid": 2042,
            "disable_firewall": true,
            "edit_hosts": true,
            "instrument_shmem_size": 5120,
            "max_scan_on_shmem": 300
        },
        "ic_proxy_properties": {
```



```
            "gp_interconnect_type": "udpifc",
            "ic_proxy_addresses_auto_discovery": true,
            "ic_proxy_port_delta": 52000,
            "ic_proxy_check": false,
            "ic_proxy_gp_interconnect_proxy_addresses": null
        },
        "use_dedicated_network": {
            "cluster_subnet": null
        },
        "adb_ident_serv": {
            "adb_ident_default_deny_policy": true
        },
        "gpperfmon": {
            "gpperfmon_authentication_type": "md5"
        },
        "Crontab": {
            "crontab_tasks": [
                "0 1 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/run_sql_to_gpssh.sh\" \"$PATH_ARENADATA_CONFIGS/gzip_pg_log.
sql\" \\ >> \"$PATH_ARENADATA_CONFIGS/gzip_pg_log.log\" 2>&1",
                "0 2 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" vacuum_system_db",
                "0 3 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" remove_orphaned_temp_schemas",
                "0 6 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" vacuum --timelimit 240",
                "0 11 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" analyze --timelimit 240",
                "0 21 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" collect_table_stats",
                "4 */8 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" vacuum_analyze_pg_catalog",
                "0 1 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.sh\" >>
\"$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.log\" 2>&1"
            ],
            "delete_operation_log_after": 30
        },
        "ssl": {
            "ssl_ciphers": [
                "HIGH",
                "MEDIUM",
                "+3DES",
                "!aNULL"
            ],
            "server_certificate_file": null,
            "server_private_key_file": null,
            "root_certificate_file": null,
            "server_crl_list_file": null,
            "ssl_pg_hba_customs": null
        },
        "LDAP": {
            "ldap_type": "MSAD",
            "ldap_uri": null,
            "ldap_basedn": null,
            "ldap_groups_basedn": null,
            "ldap_users_filter": "(&(|(objectClass=person)(objectClass=inetOrgPerson))(cn=*))",
            "ldap_groups_filter": "(&(|(objectClass=group)(objectClass=groupofnames)(objectClass=groupOfUniqueNames))(cn=*))",
            "ldap_login": null,
            "ldap_password": null,
            "ldap_login_lowercase": false,
            "ldap_group_lowercase": false,
            "sync_interval": "0 * * * *"
        },
        "pg_hba_customs": "host\tall\t\t        alice\t\t    10.92.38.0/24\t\t            pam pamservice=postgresql\nhost\tadb\t\t
new_user\t\t       10.92.41.35/32\t\t   trust "
    },
    "adcmMeta": {
        "/use_dedicated_network": {
            "isActive": false
        },
        "/adb_ident_serv": {
            "isActive": false
        },
        "/gpperfmon": {
            "isActive": true
        },
        "/Crontab": {
            "isActive": true
        },
        "/ssl": {
            "isActive": false
        },
        "/LDAP": {
```

```
            "isActive": false
        }
    },
  "description": "update"
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1357,
    "isCurrent": true,
    "creationTime": "2025-08-07T10:59:17.110045Z",
    "config": {
        "__main_info": "<p style=\"font-size:16px; \">\n<a href=\"https://arenadata.tech/en/products/arenadata-db/\"
target=\"_blank\">\nArenadata DB (ADB)</a> is a distributed database based on using MPP\n(massively parallel processing)
principle. It takes its origin\nfrom Pivotal's Greenplum database.</p><p style=\"font-size:16px; \">JDBC URL:
jdbc:postgresql://10.92.38.37:5432</p>\n<p style=\"font-size:16px; \">SSL: disabled</p>\n",
        "components": {
            "adb": "{\"logs\": [{\"type\": \"search_and_archive_directories\", \"paths\": [\"*/master/gpseg*/pg_log\",
\"*/mirror/gpseg*/pg_log\", \"*/primary/gpseg*/pg_log\", \"/home/*/gpAdminLogs\", \"/home/*/arenadata_configs/operation_log\"]}]}"
        },
        "Main": {
            "dbname": "adb",
            "enable_mirroring": true,
            "madlib": false,
            "postgis": false,
            "autocoredump": false,
            "diskquota": false,
            "plpython3": false
        },
        "Advanced": {
            "num_segments_per_host": 4,
            "segment_datadirs": {
                "/data1": ""
            },
            "segment_nobarrier": false,
            "master_datadir": "/data1",
            "master_mount_storage": false,
            "master_server_storage": null,
            "master_nobarrier": false,
            "config_dir": "arenadata_configs",
            "sysctl_params": "[[\"kernel.core_uses_pid\", \"1\"], [\"kernel.shmmni\", \"4096\"], [\"kernel.sem\", \"250 2048000 200
8192\"], [\"kernel.msgmnb\", \"65536\"], [\"kernel.msgmax\", \"65536\"], [\"kernel.msgmni\", \"2048\"], [\"net.ipv4.
tcp_syncookies\", \"1\"], [\"net.ipv4.conf.default.accept_source_route\", \"0\"], [\"net.ipv4.tcp_max_syn_backlog\", \"4096\"],
[\"net.ipv4.conf.all.arp_filter\", \"1\"], [\"net.ipv4.ip_local_port_range\", \"10000 65535\"], [\"net.ipv4.ipfrag_high_thresh\",
\"41943040\"], [\"net.ipv4.ipfrag_low_thresh\", \"31457280\"], [\"net.ipv4.ipfrag_time\", \"60\"], [\"net.core.
netdev_max_backlog\", \"10000\"], [\"net.core.rmem_max\", \"2097152\"], [\"net.core.wmem_max\", \"2097152\"], [\"vm.
overcommit_memory\", \"2\"], [\"vm.overcommit_ratio\", \"95\"], [\"vm.swappiness\", \"10\"], [\"vm.zone_reclaim_mode\", \"0\"],
[\"vm.dirty_expire_centisecs\", \"500\"], [\"vm.dirty_writeback_centisecs\", \"100\"], [\"vm.dirty_background_ratio\", \"0\"],
[\"vm.dirty_ratio\", \"0\"], [\"vm.dirty_background_bytes\", \"1610612736\"], [\"vm.dirty_bytes\", \"4294967296\"], [\"kernel.
core_pipe_limit\", \"0\"]]",
            "system_user_gid": 2042,
            "system_user_name": "gpadmin",
            "system_user_uid": 2042,
            "disable_firewall": true,
            "edit_hosts": true,
            "instrument_shmem_size": 5120,
            "max_scan_on_shmem": 300
        },
        "ic_proxy_properties": {
            "gp_interconnect_type": "udpifc",
            "ic_proxy_addresses_auto_discovery": true,
            "ic_proxy_port_delta": 52000,
            "ic_proxy_check": false,
            "ic_proxy_gp_interconnect_proxy_addresses": null
        },
        "use_dedicated_network": {
            "cluster_subnet": null
        },
```

```
        "adb_ident_serv": {
            "adb_ident_default_deny_policy": true
        },
        "gpperfmon": {
            "gpperfmon_authentication_type": "md5"
        },
        "Crontab": {
            "crontab_tasks": [
                "0 1 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/run_sql_to_gpssh.sh\" \"$PATH_ARENADATA_CONFIGS/gzip_pg_log.
sql\" \\ >> \"$PATH_ARENADATA_CONFIGS/gzip_pg_log.log\" 2>&1",
                "0 2 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" vacuum_system_db",
                "0 3 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" remove_orphaned_temp_schemas",
                "0 6 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" vacuum --timelimit 240",
                "0 11 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" analyze --timelimit 240",
                "0 21 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" collect_table_stats",
                "4 */8 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/operation.py\" vacuum_analyze_pg_catalog",
                "0 1 * * * . \"$PROFILE\" && \"$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.sh\" >>
\"$PATH_ARENADATA_CONFIGS/gzip_gpadmin_log.log\" 2>&1"
            ],
            "delete_operation_log_after": 30
        },
        "ssl": {
            "ssl_ciphers": [
                "HIGH",
                "MEDIUM",
                "+3DES",
                "!aNULL"
            ],
            "server_certificate_file": null,
            "server_private_key_file": null,
            "root_certificate_file": null,
            "server_crl_list_file": null,
            "ssl_pg_hba_customs": null
        },
        "LDAP": {
            "ldap_type": "MSAD",
            "ldap_uri": null,
            "ldap_basedn": null,
            "ldap_groups_basedn": null,
            "ldap_users_filter": "(&(|(objectClass=person)(objectClass=inetOrgPerson))(cn=*))",
            "ldap_groups_filter": "(&(|(objectClass=group)(objectClass=groupofnames)(objectClass=groupOfUniqueNames))(cn=*))",
            "ldap_login": null,
            "ldap_password": null,
            "ldap_login_lowercase": false,
            "ldap_group_lowercase": false,
            "sync_interval": "0 * * * *"
        },
        "pg_hba_customs": "host\tall\t\t        alice\t\t    10.92.38.0/24\t\t            pam pamservice=postgresql\nhost\tadb\t\t
new_user\t\t       10.92.41.35/32\t\t   trust "
    },
    "adcmMeta": {
        "/use_dedicated_network": {
            "isActive": false
        },
        "/adb_ident_serv": {
            "isActive": false
        },
        "/gpperfmon": {
            "isActive": true
        },
        "/Crontab": {
            "isActive": true
        },
        "/ssl": {
            "isActive": false
        },
        "/LDAP": {
            "isActive": false
        }
    },
    "description": "update"
}
```

Map components to a host

```
POST /api/v2/clusters/<clusterId>/mapping/ HTTP/1.1
```

This request allows you to add components to existing hosts and save the mapping. Specify the components that should be added to the specified hosts. If host-component mapping already existed for the cluster, then in order to save a new mapping, you have to copy and resend everything that existed before you created a new mapping.

Request JSON object parameters

Parameter	Type	Description
hostId	integer	Host ID
componentId	integer	Component ID

Response JSON object parameters

Parameter	Type	Description
id	integer	Mapping ID
hostId	integer	Host ID
componentId	integer	Component ID

Example request

```
POST /api/v2/clusters/3/mapping/ HTTP/1.1
Accept: application/json

[
  {
    "hostId": 2,
    "componentId": 2
  }
]
```

Example response

```
HTTP 201 Created
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 1,
    "hostId": 2,
    "componentId": 2
  }
]
```

List the current mapping of components to hosts

```
GET /api/v2/clusters/<clusterId>/mapping/ HTTP/1.1
```

This request lists the current host-component mapping (if it exists) for the specified cluster. The order is not guaranteed.

Response JSON object parameters

Parameter	Type	Description
id	integer	Mapping ID

hostId	integer	Host ID
componentId	integer	Component ID

Example request

```
GET /api/v2/clusters/8/mapping/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 3,
        "hostId": 4,
        "componentId": 33
    }
]
```

List actions

```
GET /api/v2/clusters/<clusterId>/actions/ HTTP/1.1
```

This request lists all cluster actions. If you wish to see detailed information about a particular action, add the action ID to the request:

```
GET /api/v2/clusters/<clusterId>/actions/<actionId>/ HTTP/1.1
```

Hostproviders, hosts, and services also have actions that can be accessed in a similar way.

Response JSON object parameters

Parameter	Type	Description
id	integer	Action ID
name	string	Action name
displayName	string	Displayed action name
startImpossibleReason	string	Indicates the reason due to which action launch is not possible

Example request

```
GET /api/v2/clusters/5/actions/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 4,
    "name": "install",
    "displayName": "Install",
    "startImpossibleReason": null
  },
  {
    "id": 9,
    "name": "hadoop_2.1.10_b1-1_community_upgrade_2.1.10_b1_community_2.1.8_b3_strict_false-2.1.10_b1-1_strict_false_editions-
community_state_available-installed_faulty_installed_created_state_on_success-upgradable",
    "displayName": "Upgrade: 2.1.10_b1  community",
    "startImpossibleReason": null
  }
]
```

Launch actions

```
POST /api/v2/clusters/<clusterId>/actions/<actionId>/run/ HTTP/1.1
```

Action can be started if the `startImpossibleReason` attribute is `null`. Otherwise, this attribute contains the reason why the action cannot be started. Some actions are not available if the ADCM object (host or service) is in maintenance mode.

Host-component mapping rules that affect actions are listed in `hostComponentMapRules` section of action information. `hostComponentMapRules : []` means that there are no host-component mapping rules that affect actions.

Actions for services, components, hostproviders, and hosts are launched similarly to cluster actions shown below.

Request JSON object parameters

Parameter	Type	Description
hostComponentMap	array of objects	Information on mapping between hosts and components
configuration	object	List of configuration attributes and their values
isVerbose	boolean	Defines whether to show details
shouldBlockObject	boolean	Specifies whether to create a blocking or non-blocking concern. Possible values: `True`, `False`
description	string	Comment on the action. String length limit: 255 characters

Response JSON object parameters

Parameter	Type	Description
id	integer	Action ID
name	string	Action name
displayName	string	Displayed action name
action	object	Action information
status	string	Action status
startTime	string	Moment the action was started
endTime	string	Moment the action was stopped
duration	float	Action duration

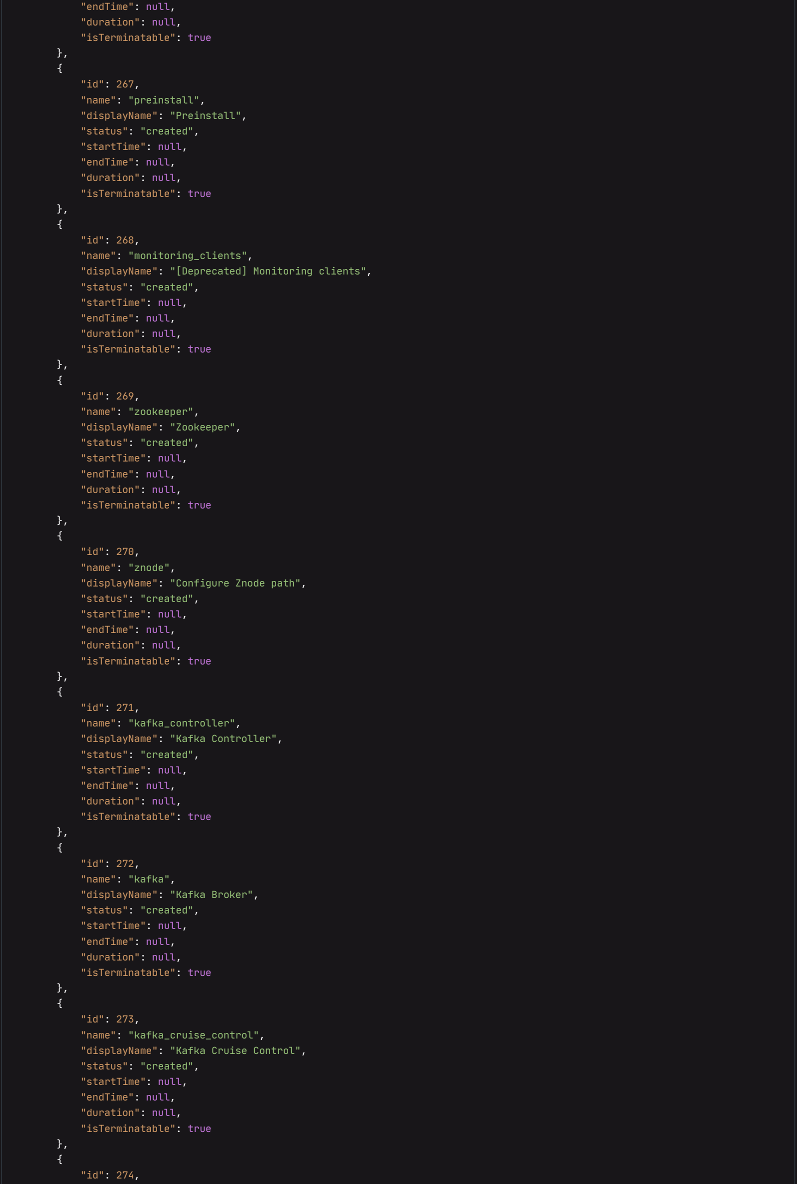
isTerminatable	boolean	Ability to terminate the execution of a job related to action
childJobs	array of objects	Linked jobs
objects	array of objects	ADCM objects related to action
description	string	Comment on the action

Example request

```
POST /api/v2/clusters/8/actions/391/run/ HTTP/1.1
Accept: application/json

{
    "hostComponentMap": [],
    "configuration": {
        "config": {
            "disable_selinux": true,
            "disable_firewalld": true,
            "install_jdk": true,
            "swap_off": true,
            "configure_etc": false,
            "service_check_on": true
        },
        "adcmMeta": {}
    },
    "isVerbose": false,
    "shouldBlockObject": true,
    "description": "Install ADS cluster"
}
```

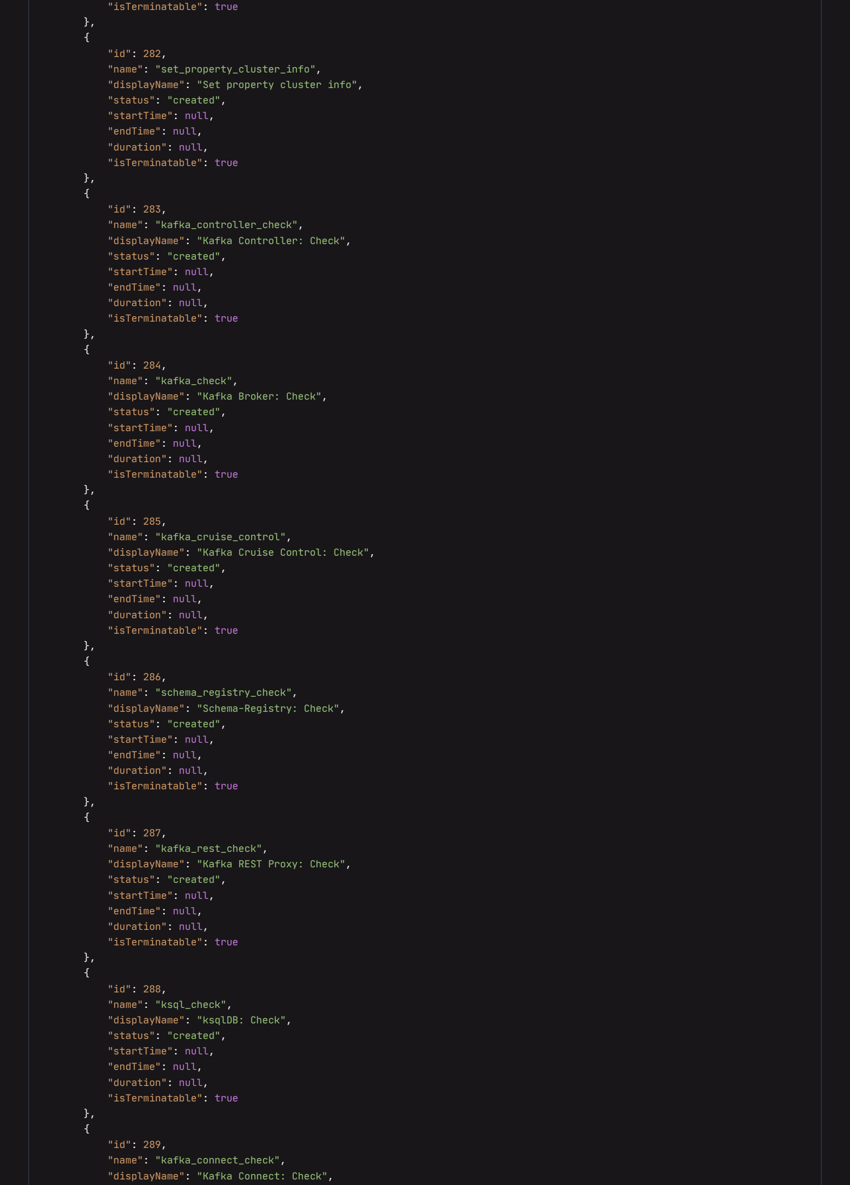
Example response

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 24,
    "name": "Install",
    "displayName": "Install",
    "action": {
        "id": 391,
        "name": "Install",
        "displayName": "Install"
    },
    "status": "created",
    "startTime": null,
    "endTime": null,
    "duration": null,
    "isTerminatable": true,
    "childJobs": [
        {
            "id": 265,
            "name": "cluster_precheck",
            "displayName": "Cluster precheck",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 266,
            "name": "repos",
            "displayName": "Repositories setup",
            "status": "created",
            "startTime": null,
```



```
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 267,
            "name": "preinstall",
            "displayName": "Preinstall",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 268,
            "name": "monitoring_clients",
            "displayName": "[Deprecated] Monitoring clients",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 269,
            "name": "zookeeper",
            "displayName": "Zookeeper",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 270,
            "name": "znode",
            "displayName": "Configure Znode path",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 271,
            "name": "kafka_controller",
            "displayName": "Kafka Controller",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 272,
            "name": "kafka",
            "displayName": "Kafka Broker",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 273,
            "name": "kafka_cruise_control",
            "displayName": "Kafka Cruise Control",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 274,
```

```
            "name": "schema_registry",
            "displayName": "Schema-Registry",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 275,
            "name": "kafka_rest",
            "displayName": "Kafka REST Proxy",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 276,
            "name": "ksql",
            "displayName": "ksqlDB",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 277,
            "name": "kafka_connect",
            "displayName": "Kafka Connect Worker",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 278,
            "name": "nifi_server",
            "displayName": "NiFi Server",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 279,
            "name": "nifi_registry",
            "displayName": "NiFi-Registry",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 280,
            "name": "minifi_server",
            "displayName": "MiNiFi C2 Server",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 281,
            "name": "minifi_agent",
            "displayName": "MiNiFi Agent",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
```



```
            "isTerminatable": true
        },
        {
            "id": 282,
            "name": "set_property_cluster_info",
            "displayName": "Set property cluster info",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 283,
            "name": "kafka_controller_check",
            "displayName": "Kafka Controller: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 284,
            "name": "kafka_check",
            "displayName": "Kafka Broker: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 285,
            "name": "kafka_cruise_control",
            "displayName": "Kafka Cruise Control: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 286,
            "name": "schema_registry_check",
            "displayName": "Schema-Registry: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 287,
            "name": "kafka_rest_check",
            "displayName": "Kafka REST Proxy: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 288,
            "name": "ksql_check",
            "displayName": "ksqlDB: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 289,
            "name": "kafka_connect_check",
            "displayName": "Kafka Connect: Check",
```

```
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 290,
            "name": "nifi_server_check",
            "displayName": "NiFi Server: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 291,
            "name": "nifi_registry_check",
            "displayName": "NiFi-Registry: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 292,
            "name": "minifi_server_check",
            "displayName": "MiNiFi C2 Server: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 293,
            "name": "minifi_agent_check",
            "displayName": "MiNiFi Agent: Check",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        },
        {
            "id": 294,
            "name": "prometheus_monitoring",
            "displayName": "Prometheus monitoring: install",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": true
        }
    ],
    "objects": [
        {
            "type": "cluster",
            "id": 8,
            "name": "ADS"
        }
    ],
    "description": "Install ADS cluster"
}
```

List host groups for actions

```
GET /api/v2/clusters/<clusterId>/action-host-groups/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/ HTTP/1.1
```

These requests list action host groups configured at the cluster, service, or component level, respectively. To view detailed information about a particular host group, specify the group ID in the query, for example (for a cluster's group of hosts):

```
GET /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/ HTTP/1.1
```

The response supports pagination.

JSON object results parameters

Parameter	Type	Description
id	integer	Host group ID
name	string	Host group name
description	string	Host group description
hosts	array of objects	List of hosts in the group. Each object in the list contains a host ID and name

Example request

```
GET /api/v2/clusters/1/services/1/action-host-groups/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 2,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "name": "hosts_1-3",
            "description": "",
            "hosts": [
                {
                    "id": 1,
                    "name": "dev-adqm-1.ru-central1.internal"
                },
                {
                    "id": 3,
                    "name": "dev-adqm-3.ru-central1.internal"
                }
            ]
        },
        {
            "id": 2,
            "name": "hosts_2-4",
            "description": "",
            "hosts": [
                {
                    "id": 2,
                    "name": "dev-adqm-2.ru-central1.internal"
                },
                {
                    "id": 4,
                    "name": "dev-adqm-4.ru-central1.internal"
```

```
                }
            ]
        }
    ]
}
```

Create a host group for actions

```
POST /api/v2/clusters/<clusterId>/action-host-groups/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/ HTTP/1.1
```

These requests create a new host group for actions at the cluster, service, or component level, respectively.

Request JSON object parameters

Parameter	Type	Description
name	string	Host group name
description	string	Host group description

Response JSON object parameters

Parameter	Type	Description
id	integer	Host group ID
name	string	Host group name
description	string	Host group description
hosts	array of objects	List of hosts in the group

Example request

```
POST /api/v2/clusters/1/action-host-groups/ HTTP/1.1
Accept: application/json

{
    "name": "test_group_of_hosts",
    "description": "This host group is created from API."
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 3,
    "name": "test_group_of_hosts",
    "description": "This host group is created from API.",
    "hosts": []
}
```

Add a host to a group for actions

```
POST /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/hosts/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/hosts/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/hosts/ HTTP/1.1
```

These requests add a host to a host group for actions at the cluster, service, or component level, respectively. A host ID is required.

Hosts (with their IDs) that can be added to a group

```
GET /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/host-candidates/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/host-candidates/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/host-candidates/
HTTP/1.1
```

Request JSON object parameters

Parameter	Type	Description
hostId	integer	Host ID

Response JSON object parameters

Parameter	Type	Description
id	integer	Host ID
name	string	Host name

Example request

```
POST /api/v2/clusters/1/action-host-groups/3/hosts/ HTTP/1.1
Accept: application/json

{
    "hostId": 3
}
```

Example response

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 3,
    "name": "dev-adqm-3.ru-central1.internal"
}
```

Remove a host from a group for actions

```
DELETE /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/hosts/<hostId>/ HTTP/1.1
```

```
DELETE /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/hosts/<hostId>/ HTTP/1.1
```

```
DELETE /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/hosts/<hostId>/ HTTP/1.1
```

These requests remove a host from a group for actions at the cluster, service, or component level, respectively.

Example request

```
DELETE /api/v2/clusters/1/services/1/components/1/action-host-groups/6/hosts/1/ HTTP/1.1
```

Launch actions for host groups

```
POST /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/actions/<actionId>/run/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/actions/<actionId>/run/ HTTP/1.1
```

```
POST /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/actions/<actionId>/run/ HTTP/1.1
```

These requests start an action for a host group, created at the cluster, service, or component level, respectively. An action can be started if its `startImpossibleReason` attribute is `null`. Otherwise, this attribute contains the reason why the action cannot be started.

A set of actions available for execution on hosts in a group depends on the ADCM object at the level of which the host group is created — product, object type (cluster, service, or component), and state of this object.

List of actions available for a host group

```
GET /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/actions/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/actions/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/actions/ HTTP/1.1
```

Information about a specific action

```
GET /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/actions/<actionId>/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/actions/<actionId>/ HTTP/1.1
```

```
GET /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/actions/<actionId>/ HTTP/1.1
```

Request JSON object parameters

Parameter	Type	Description
hostComponentMap	array of objects	Information on mapping between hosts and components
configuration	object	List of configuration attributes and their values

isVerbose	boolean	Defines whether to show details
shouldBlockObject	boolean	Specifies whether to create a blocking or non-blocking concern. Possible values: `True`, `False`
description	string	Comment on the action. String length limit: 255 characters

Response JSON object parameters

Parameter	Type	Description
id	integer	Action ID
name	string	Action name
displayName	string	Displayed action name
action	object	Action information
status	string	Action status
startTime	string	Moment the action was started
endTime	string	Moment the action was stopped
duration	float	Action duration
isTerminatable	boolean	Ability to terminate the execution of a job related to action
childJobs	array of objects	Linked jobs
objects	array of objects	ADCM objects related to action
description	string	Comment on the action

Example request

```
POST /api/v2/clusters/1/services/1/action-host-groups/1/actions/162/run/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 20,
    "name": "restart",
    "displayName": "Restart",
    "action": {
        "id": 162,
        "name": "restart",
        "displayName": "Restart"
    },
    "status": "created",
    "startTime": null,
    "endTime": null,
    "duration": null,
    "isTerminatable": false,
    "childJobs": [
        {
            "id": 57,
            "name": "restart",
            "displayName": "Restart",
```

```
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": false
        }
    ],
    "objects": [
        {
            "type": "cluster",
            "id": 1,
            "name": "Test ADQM cluster"
        },
        {
            "type": "service",
            "id": 1,
            "name": "ADQMDB"
        },
        {
            "type": "action_host_group",
            "id": 1,
            "name": "hosts_1-3"
        }
    ]
}
```

Delete a non-blocking concern

```
DELETE /api/v2/concerns/<concernId> HTTP/1.1
```

This request deletes the specified non-blocking concern. You can delete a non-blocking concern if you have one of the following roles:

- `Cluster Administrator`
- `Service Administrator`
- `Provider Administrator`
- `ADCM Administrator`

Example request

```
DELETE /api/v2/concerns/498/ HTTP/1.1
```

Example response

```
HTTP 204 No Content
```

Delete a host group for actions

```
DELETE /api/v2/clusters/<clusterId>/action-host-groups/<groupId>/ HTTP/1.1
```

```
DELETE /api/v2/clusters/<clusterId>/services/<serviceId>/action-host-groups/<groupId>/ HTTP/1.1
```

```
DELETE /api/v2/clusters/<clusterId>/services/<serviceId>/components/<componentId>/action-host-groups/<groupId>/ HTTP/1.1
```

These requests remove a host group from a cluster, service, or component configuration, respectively.

Example request

```
DELETE /api/v2/clusters/1/action-host-groups/3/ HTTP/1.1
```

List upgrades

```
GET /api/v2/clusters/<clusterId>/upgrades/ HTTP/1.1
```

This request lists all available cluster upgrades. Hostprovider upgrades can be listed similarly.

Response JSON object parameters

Parameter	Type	Description
id	integer	Upgrade ID
name	string	Upgrade name
displayName	string	Displayed upgrade name

Example request

```
GET /api/v2/clusters/12/upgrades/ HTTP/1.1
```

Example response

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
  {
    "id": 28,
    "name": "2.8.1_arenadata1_b1 enterprise",
    "displayName": "2.8.1_arenadata1_b1 enterprise",
  }
]
```

Launch upgrades

```
POST /api/v2/clusters/<clusterId>/upgrades/<upgradeId>/run/ HTTP/1.1
```

Launching upgrades is similar to launching actions. Upgrades are available if `isUpgradable : true` is present in cluster information. You must accept the bundle license agreement before launching the upgrade.

Host-component mapping rules that affect upgrades are listed in `hostComponentMapRules` section of upgrade information. `hostComponentMapRules : []` means that there are no host-component mapping rules that affect upgrades.

Request JSON object parameters

Parameter	Type	Description
hostComponentMap	array of objects	Information on mapping between hosts and components
configuration	object	List of configuration attributes and their values
isVerbose	boolean	Defines whether to show details

Response JSON object parameters

Parameter	Type	Description
id	integer	Upgrade ID

name	string	Upgrade name
displayName	string	Displayed upgrade name
action	object	Action information
status	string	Upgrade status
startTime	string	Moment the upgrade was started
endTime	string	Moment the upgrade was stopped
duration	float	Upgrade duration
isTerminatable	boolean	Ability to terminate the execution of a job related to upgrade
childJobs	array of objects	Linked jobs
objects	array of objects	ADCM objects related to upgrade

Example request

```
POST /api/v2/clusters/19/upgrades/46/run/ HTTP/1.1
Accept: application/json

{
  "hostComponentMap": [],
  "configuration": null,
  "isVerbose": false
}
```

Example response

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 20,
    "name": "hadoop_2.1.10_b1-1_enterprise_upgrade_2.1.10_b1_enterprise_2.1.8_b3_strict_false-2.1.10_b1-1_strict_false_editions-
enterprise_state_available-installed_faulty_installed_created_state_on_success-upgradable",
    "displayName": "Upgrade: 2.1.10_b1  enterprise",
    "action": {
        "id": 2528,
        "name": "hadoop_2.1.10_b1-1_enterprise_upgrade_2.1.10_b1_enterprise_2.1.8_b3_strict_false-2.1.10_b1-1_strict_false_editions-
enterprise_state_available-installed_faulty_installed_created_state_on_success-upgradable",
        "displayName": "Upgrade: 2.1.10_b1  enterprise"
    },
    "status": "created",
    "startTime": null,
    "endTime": null,
    "duration": null,
    "isTerminatable": false,
    "childJobs": [
        {
            "id": 471,
            "name": "bundle upgrade",
            "displayName": "Upgrade bundle",
            "status": "created",
            "startTime": null,
            "endTime": null,
            "duration": null,
            "isTerminatable": false
        }
    ],
    "objects": [
        {
            "type": "cluster",
            "id": 19,
            "name": "adh-2.1.8"
        }
    ]
}
```

ADCM AIO Client

Adelya Zvyagintseva

Contents

To Table of Contents

- Quick start with ADCM AIO Client
- Upload a bundle
- Create a hostprovider
- Create hosts
- Create a cluster
- Add services
- Add hosts to a cluster
- Manage mapping
- Manage an object configuration
  - Get a configuration parameter value
  - Change a configuration parameter value
- Manage an object config group
  - Create a cluster config group
  - List cluster config groups
  - Add hosts to a cluster config group
  - Get the value of a service config group parameter
  - Change the value of a service config group parameter
- Manage action host groups
  - Create an action host group
  - List action host groups
  - Add a host to an action host group
  - Launch an action for an action host group
- Import
- Perform object actions
  - List actions
  - Prepare an action
  - Launch an action
  - Get a job status
- Work with object collections
- Errors

CAUTION

Currently, the Alpha version of ADCM AIO Client is implemented. If necessary, backward compatibility with future versions can be broken. Release will be assigned the Alpha status as long as the major version number is zero (`0.<minor>.<patch>`). The ADCM AIO Client releases are presented in the GitHub repository.

Asynchronous HTTP Client for ADCM (ADCM AIO Client) allows you to perform multiple requests simultaneously without blocking the main thread which is provided by the asynchronous architecture.

Due to asynchronous request handling, ADCM AIO Client enables you to manage tasks more efficiently, reducing latency and improving throughput. The implemented advanced error handling mechanisms are designed to manage failures gracefully in an asynchronous environment.

The object model of ADCM AIO Client reflects the ADCM architecture and the structure of its objects, such as clusters, hosts, services, components, bundles, etc.

This article contains examples of using ADCM AIO Client to manage the ADCM objects and configurations.

Quick start with ADCM AIO Client

Follow the steps below to set up and use ADCM AIO Client.

CAUTION

Before installation, make sure that Python 3.12.0 or later and `pip` are available within your environment.

1. Install ADCM AIO Client using the following command:

```
$ pip install adcm_aio_client
```

2. Configure the connection to ADCM. Below is an example of getting the list of all clusters to describe how to configure the connection:

```
import asyncio
from adcm_aio_client import ADCMSession, Credentials

# User credentials to connect to the ADCM web interface
credentials = Credentials(username="<username>", password="<password>")

# Configure the request retry
kwargs = {"verify": False, "timeout": 10, "retry_interval": 1, "retry_attempts": 1}

async def get_clusters():

    # Create the connection to ADCM
    async with ADCMSession(url="http://<ip_address>:<port>", credentials=credentials, **kwargs) as client:
        # Get clusters
        clusters = await client.clusters.all()

asyncio.run(get_clusters())
```

where:

- `timeout` — the ADCM connection timeout in seconds (the default value is `600`);
- `retry_interval` — the interval between ADCM connection attempts (the default value is `3`);
- `retry_attempts` — the number of retry attempts for the ADCM connection (the default value is `1`);
- `<username>` — the user account name;
- `<password>` — the user account password;
- `<ip_address>` — the IP address of the host where the ADCM container is deployed;
- `<port>` — the port number where the ADCM container is running.

To establish an HTTPS connection to ADCM (using the SSL certificate), pass the `verify` parameter. This parameter determines the full path to the file (including the file name) that contains the certificate verified through a certificate authority (CA).

NOTE

Make sure that the URL begins with `https` and port `8443` is entered as the `<port>` parameter value.

```
import asyncio
from adcm_aio_client import ADCMSession, Credentials

credentials = Credentials(username="<username>", password="<password>")

verify = "/<file_path>/<certificate_file_name>.pem"

async def get_clusters():
    # Create the connection to ADCM using SSL
    async with ADCMSession(url="https://<ip_address>:8443", credentials=credentials, verify=verify) as client:
        # Get clusters
        clusters = await client.clusters.all()

asyncio.run(get_clusters())
```

Upload a bundle

```
# Upload a bundle with the subsequent acceptance of the license agreement
await client.bundles.create("https://downloads.arenadata.io/ADB/5.25.0_arenadata6/bundles/adcm_cluster_adb_v5.25.0_arenadata6_b1-1_enterprise.tgz", accept_license=True)
```

Create a hostprovider

```
# Determine the uploaded bundle based on which the hostprovider will be created
bundle = await client.bundles.get(name__eq="SSH Common")

# Create the hostprovider
hostprovider = await client.hostproviders.create(bundle=bundle, name="SSH")
```

Create hosts

```
# Determine the hostprovider
hostprovider=await client.hostproviders.get(name__eq="SSH")

# Create the host
host = await client.hosts.create(name="zar-adb-master", hostprovider=hostprovider))
```

Create a cluster

```
# Determine the uploaded bundle based on which the cluster will be created
bundle = await client.bundles.get(name__eq="ADB", version__eq="5.25.0_arenadata6_b1-1")

# Create the cluster
cluster = await client.clusters.create(bundle=bundle, name="Test_ADB_cluster")
```

Add services

```
# Specify the cluster and service to be added
cluster = await client.clusters.get(name__eq="Test_ADB_cluster")
service_name_conditions = Filter(attr="display_name", op="ieq", value="ADB")

# Add the service with the subsequent acceptance of the license agreement and definition of dependencies between services
await cluster.services.add(service_name_conditions, accept_license=True, with_dependencies=False)
```

Add hosts to a cluster

```
# Specify the required cluster and host to be added
cluster = await client.clusters.get(name__eq = "Test_ADB_cluster")
host = await client.hosts.get(name__ieq = "zar-adb-master")

# Add the host to the cluster
added_host = await cluster.hosts.add(host=host)
```

Manage mapping

ADCM AIO Client provides the `mapping.add(component: Component | List[Component], host: Host| List[Host])` method to configure host-component mapping.

NOTE

Before the `mapping.save()` method is called, the host-component mapping is saved locally.

```
"""
Mapping components to hosts locally
"""
# Get hosts and components for mapping
components = await mapping.components.filter(display_name__iin=component_name_list)
hosts = await mapping.hosts.filter(name__iin=host_name_list)

# Pass obtained hosts and components as an input to the function
await mapping.add(component=components, host=hosts)

# Define host-component mapping
master_node = ["zar-adb-master"]
standby_node = ["zar-adb-standby"]
segment_node = ["zar-adb-segment1", "zar-adb-segment2"]

master_component = ["ADB Master"]
standby_component = ["ADB Standby"]
segment_component = ["ADB Segment"]

cluster = await client.clusters.get(name__eq="Test_ADB_cluster")
mapping = await cluster.mapping

await map_components_to_hosts(mapping, master_component, master_node)
await map_components_to_hosts(mapping, standby_component, standby_node)
await map_components_to_hosts(mapping, segment_component, segment_node)

# Save host-component mapping
await mapping.save()
```

Also, ADCM AIO Client allows you to manage mapping using the following methods:

- `mapping.all()` — returns the current mapping pairs.
- `mapping.empty()` — resets local changes in the host-component mapping, leaving them empty.
- `mapping.refresh()` — actualizes current local changes in mapping taking into account remote changes. In case of conflicts, it applies one of the following strategies:
  - `apply_local_changes` — local changes have higher priority than remote changes.
  - `apply_remote_changes` — remote changes have higher priority than local changes.

  The given example shows how to use the `mapping.refresh()` method. In case of conflicts, local changes will be applied because `strategy` takes the `apply_local_changes` value.

```
cluster = await client.clusters.get(name__eq="Test_ADB_cluster")
mapping = await cluster.mapping

component_2 = await mapping.components.filter(display_name__eq="ADBM server")
host_1 = await mapping.hosts.filter(name__eq="zar-adb-master")
host_2 = await mapping.hosts.filter(name__eq="zar-adb-control")

await mapping.refresh(strategy=apply_local_changes)

await mapping.remove(component_2, host_1)
await mapping.add(component_2, host_2)

await mapping.save()
```

Manage an object configuration

ADCM AIO Client allows you to get and change the value of an object (hostprovider, cluster, service, and component) and ADCM configuration parameter.

Local changes of the configuration parameter values can also be reset by using the `config.reset()` method if required.

Below are examples of configuration settings for a cluster.

NOTE

Pay attention to how to specify the path to a configuration parameter in the tree of configuration settings.

Get a configuration parameter value

```
# Specify the required cluster
cluster = await client.clusters.get(name__eq="Test_ADB_cluster")

# Get the object that controls the cluster configuration
config = await cluster.config

# Get the configuration parameter value
parameter = config['Vault integration']['Address of the Vault server'].value
```

Change a configuration parameter value

```
# Specify the required cluster
cluster = await client.clusters.get(name__eq="Test_ADB_cluster")

# Get the object that controls the cluster configuration
config = await cluster.config

# Get the configuration parameter
parameter = config['Vault integration']['Address of the Vault server']

# Set a new configuration parameter value
parameter.set('192.168.1.120')

# Save a new configuration
await config.save()
```

Manage an object config group

ADCM AIO Client provides methods described below to perform the following operations with config groups:

- Create a cluster config group
- List cluster config groups
- Add hosts to a cluster config group
- Get a value of a service config group parameter
- Change the value of a service config group parameter

Also, ADCM AIO Client allows you to call the following methods for managing object config groups:

- `configuration.reset()` — resets local changes in the config group configuration.
- `config_group.hosts.remove(host: Host| List[Host] | Filter)` — removes hosts from the config group.
- `config_group.delete()` — removes the config group from the cluster.

Create a cluster config group

```
# Specify the required cluster
cluster = await client.clusters.get(name__eq="Test_ADH_cluster")

# Specify the object that controls the host group of the cluster
config_group = cluster.config_host_groups

# Get host objects from the list of hosts
hosts_for_add = await cluster.hosts.filter(name__iin=["zar-adh-segment1", "zar-adh-segment2"])

# Create the config group
await config_group.create(name="config-group-1", hosts=hosts_for_add)
```

List cluster config groups

```
# Specify the required cluster
cluster = await client.clusters.get(name__eq="Test_ADH_cluster")

# Get the list of config groups
config_group_list = await cluster.config_host_groups.all()
```

Add hosts to a cluster config group

```
# Specify the required cluster
cluster = await client.clusters.get(name__eq="Test_ADH_cluster")

# Specify the config group to add a host
config_group = await cluster.config_host_groups.get(name__eq="config-group-2")

# Get the added host
added_host = await cluster.hosts.filter(name__iin="zar-adh-master")

# Add the host
host_add = await config_group.hosts.add(host=added_host)
```

Get the value of a service config group parameter

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="Test_ADH_cluster")
service = await cluster.services.get(name__eq="hbase")

# Specify the service config group
config_group = await service.config_host_groups.get(name__eq="service-group-1")

# Get the object that controls the config group configuration
configuration = await config_group.config

parameter = configuration["hbase-site.xml"]["hbase.client.max.perregion.tasks"].value
```

Change the value of a service config group parameter

NOTE

The `configuration["<Path to parameter>"].sync()` method allows you to turn on synchronization of the parameter value with the object primary configuration. To turn off synchronization and change the parameter value, use the `configuration["<Path to parameter>"].desync()` method.

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="Test_ADH_cluster")
service = await cluster.services.get(name__eq="hbase")

# Specify the service config group
config_group = await service.config_host_groups.get(name__eq="service-group-1")
configuration = await config_group.config

# Enable parameter synchronization with primary configurations
configuration["hbase-site.xml"]["hbase.client.max.perregion.tasks"].desync()

# Set a new config group parameter value
configuration["hbase-site.xml"]["hbase.client.max.perregion.tasks"].set(50)

# Save a new configuration
await configuration.save()
```

Manage action host groups

ADCM AIO Client provides methods that allow you to perform the following operations with action host groups:

- Create an action host group
- List action host groups
- Add a host to an action host group
- Launch an action for an action host group

Also, ADCM AIO Client allows you to call the following methods for managing action host groups:

- `action_host_group.hosts.remove()` — removes hosts from the action host group.
- `action_host_group.delete()` — removes the action host group from the ADCM object.

Create an action host group

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="Test ADQM cluster")
service = await cluster.services.get(name__eq="adqmdb")

# Specify the object that controls the action host group
action_group = service.action_host_groups

# Identify host objects from the list of hosts
hosts_for_add = await cluster.hosts.filter(name__iin=["dev-adqm-01.ru-central1.internal", "dev-adqm-04.ru-central1.internal"])

# Create the action host group
action_group_create = await action_group.create(name="hosts_1-4", hosts=hosts_for_add)
```

List action host groups

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="Test_ADQM_cluster")
service = await cluster.services.get(name__eq="adqmdb")

# Get the list of action host groups
action_group_list = await service.action_host_groups.all()
```

Add a host to an action host group

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="Test ADQM cluster")
service = await cluster.services.get(name__eq="adqmdb")

# Specify the action host group
action_host_group = await service.action_host_groups.get(name__eq="hosts_1-3")

# Get the added hosts
added_host = await cluster.hosts.filter(
    name__iin=["dev-adqm-02.ru-central1.internal", "dev-adqm-04.ru-central1.internal"])

# Add the host
host_add = await action_host_group.hosts.add(host=added_host)
```

Launch an action for an action host group

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="Test_ADQM_cluster")
service = await cluster.services.get(name__eq="adqmdb")

# Specify the action host group
action_host_group = await service.action_host_groups.get(name__eq="hosts_1-3")

# Specify the action
action = await action_host_group.actions.get(display_name__eq="Restart")

task = await action.run()
```

Import

ADCM AIO Client provides the `cluster_import.add(*sources: Cluster | Service)` method to transfer specific service settings to a cluster.

```
# Specify the cluster in which the service settings will be imported
cluster = await client.clusters.get(name__eq="Test_ADB_cluster")

# Get the object that controls import in the cluster
cluster_import = await cluster.imports

# Specify the cluster that contains service settings to be imported
cluster_for_import = await client.clusters.get(name__eq="Monitoring")

# Get the service settings to be imported
imported_service = await monitoring_cluster.services.get(name__eq="grafana")

# Import
imports = await cluster_import.add([imported_service])
```

Also, ADCM AIO Client allows you to manage import using the following methods:

- `cluster_import.remove(*sources: Cluster | Service)` — removes imported settings of one service from a cluster.
- `cluster_import.set()` — removes imported settings of all services from a cluster.

Perform object actions

ADCM AIO Client provides methods that allow you to perform object actions, including the following:

1. List actions.
2. Prepare an action.
3. Launch an action:
   - without blocking an action execution thread;
   - with blocking an action execution thread.
4. Get a job status.

As an example, consider the process of launching the Expand action.

List actions

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__eq="ADB_cluster")
service = await cluster.services.get(name__eq="adb")

# Get all service actions
actions = await service.actions.all()
```

Prepare an action

Because the ***Expand*** action allows you to add new hosts to a cluster component, you need to prepare the action for the launch by specifying the following settings:

1. Set object configuration parameters.
2. Map cluster hosts to components.

Also, ADCM AIO Client allows you to run actions as follows:

- in the non-blocking mode;
- with the option to get detailed logs.

Set configuration parameters

NOTE

Methods for managing an object configuration are also applicable to actions containing object configuration parameters.

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__ieq="Test_ADB_cluster")
service = await cluster.services.get(name__eq="adb")

# Specify the action
expand_action = await cluster.actions.get(name__eq="Expand")

# Get the object of the action containing configuration parameters
configuration = await expand_action.config

# Determine the parameter and its new value
configuration["Reboot timeout, sec"].set(605)
```

Map hosts to components

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__ieq="Test_ADB_cluster")
service = await cluster.services.get(name__eq="adb")

# Specify the action
expand_action = await cluster.actions.get(name__eq="Expand")

# Specify hosts and components for mapping
components = await service.components.filter(name__iin=["ADB Segment", "PXF"])
hosts = await cluster.hosts.get(name__eq="test-host")

# Get the object of the action including host-component mapping
mapping = await expand_action.mapping

# Add the local mapping changes
await mapping.add(components, hosts)
```

Write detailed logs

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__ieq="Test_ADB_cluster")
service = await cluster.services.get(name__eq="adb")

# Specify the action
expand_action = await cluster.actions.get(name__eq="Expand")

expand_action.verbose = True
```

Set the non-blocking mode

```
# Specify the required cluster and service
cluster = await client.clusters.get(name__ieq="Test_ADB_cluster")
service = await cluster.services.get(name__eq="adb")

# Specify the action
expand_action = await cluster.actions.get(name__eq="Expand")

expand_action.non_blocking = True
```

Launch an action

Without blocking the action execution thread

```
job = await expand_action.run()
```

With blocking an action execution thread

NOTE

A custom condition can be used to exit on timeout.

```
job = await expand_action.run()
await job.wait(exit_condition=default_exit_condition)
```

Get a job status

When launching actions, ADCM starts jobs. To follow the action execution, get the job status using the `job.get_status()` method.

```
job_status = await job.get_status()
```

Work with object collections

ADCM AIO Client provides methods described below to perform the following operations:

- `all()` — returns a list of all objects;
- `list()` — returns a paginated list of objects satisfying request parameters;

  Example of using pagination

```
clusters = await client.clusters.list(query={"limit": 4, "offset": 10})
```

  where:

  - `limit` — the maximum number of objects in a list;
  - `offset` — the number of objects to skip.
- `get()` — returns one object satisfying the specified conditions;
- `filter()` — returns the list of objects that match the given conditions.

To filter object collections, use the `filter()` method specifying filter parameters and field lookups described below.

Field lookups

Field lookup	Description
eq	Equal
ieq	Case-insensitive equal
ne	Not equal
ine	Case-insensitive not equal
contains	Substring search
icontains	Case-insensitive substring search
in	Match of one of the given comma-separated values in the `list` format
iin	Case-insensitive match of one of the given comma-separated values in the `list` format
exclude	Not equal to the given comma-separated values in the `list` format
iexclude	Case-insensitive not equal to the given comma-separated values in the `list` format

Note that different objects support different filter parameters.

Bundle

- `name`
- `display_name`
- `version`
- `edition`

Cluster

- `name`
- `bundle`
- `status`

Service

- `name`
- `display_name`
- `status`

Component

- `name`
- `display_name`

Hostprovider

- `name`
- `bundle`

Host

- `name`
- `status`

Action

- `name`
- `display_name`

Job

- `status`
- `name`
- `display_name`
- `object`
- `action`

Config group, action host group

`name`

The `bundle` and `action` parameters can be used only with lookups `eq`, `ne`, `in`, and `exclude`.

```
clusters = await client.clusters.filter(name__iin=['monitoring', 'adb'])
```

```
clusters = await client.clusters.filter(name__icontains="test_adb_cluster")
```

Errors

The table below lists errors that may occur during processing requests.

Errors

Error code	Description
WaitTimeoutError	Execution timeout has exceeded the threshold
ClientInitError	Failed to connect to ADCM at specified URL
NotSupportedVersionError	1. Failed to detect ADCM version at the specified URL. Most likely, the ADCM version is lower than the minimum supported ADCM version. 2. The ADCM version is not supported.
NoCredentialsError	Unable to locate credentials
AuthenticationError	Login to ADCM at the specified URL has failed. Possible cause is incorrect credentials
LoginError	Login to ADCM at the specified URL has failed
LogoutError	Logout from ADCM at the specified URL has failed
RetryRequestError	Request failed during the specified retry interval
ResponseDataConversionError	1. Invalid return type. Supported types: `dict` and `list`. 2. Input data is not in valid JSON format.
BadRequestError	Request cannot be processed due to user error
UnauthorizedError	Request failed due to the lack of valid authentication credentials
PermissionDeniedError	Current user does not have permission to perform a request
NotFoundError	The object cannot be found
ConflictError	Request could not be completed due to a conflict with the current state of the server
ServerError	Server was unable to process the request
BadGatewayError	While processing the request, server received an invalid response from upstream server
ServiceUnavailableError	Server is unable to handle the request at this time due to a temporary overload or scheduled maintenance
AccessorError	Unspecified accessor error
MultipleObjectsReturnedError	More than one object found
ObjectDoesNotExistError	No objects found with the given filter
OperationError	1. Failed to download the bundle from the specified URL. 2. HTTP error occurred.
HostNotInClusterError	Host is not added to the cluster, or it is not upgraded
ConfigComparisonError	1. The previous version of the *schema.yaml* specification file does not match the current file. 2. Cannot refresh configuration after cluster upgrade: *schema.yaml* is different for local and remote servers.
ConfigNoParameterError	No configuration parameter with specified name in the tree of configuration settings
NoMappingInActionError	Selected action does not allow making changes to the mapping
NoConfigInActionError	Selected action does not allow making changes to configuration parameters
InvalidFilterError	Incorrect use of the `filter()` method. For example, invalid inline filter format or specifying an empty object collection

ADCM releases

Mikhail Serov, Adelya Zvyagintseva

Contents

To Table of Contents

2.9.1
2.9.0
2.8.0
2.7.1
2.7.0
2.6.0
2.5.1
2.5.0
2.4.0
2.3.0
2.2.0
2.1.2
2.1.1
2.1.0
2.0.0
2023.11.28.07
2023.10.10.08
2023.09.28.17
2023.07.10.17
2023.06.14.16
2023.04.21.12
2023.04.10.22
2023.03.27.20
2022.12.26.18
2022.12.07.10
2022.10.04.17
2022.08.10.17
2022.05.26.14
2022.04.18.13
2022.03.25.16
2022.02.01.06

2.9.1

Date: 09.12.2025

Bug fixes

- Fixed a bug that caused default values for new password input fields to be stored in plain text when updating the product
- Fixed the error that occurred when creating a subhost if the host configuration contained nested groups
- Fixed the error that occurred during the product update when configuration parameters of the `file` type were used

2.9.0

Date: 01.12.2025

New features

- It is now possible to leave comments when running actions. These comments are displayed on the ***Jobs*** page
- Added the ability to run ADCM with HashiCorp Consul environment variables intended for products. These variables will allow products to integrate their services with HashiCorp Consul
- Added support for nested groups. To improve navigation, configurations can have multiple levels of nested groups
- Added support for mutually exclusive groups. To improve navigation, a new configuration element was added that allows you to select one group from a list

Improvements

- UUID of each object was added to the *config.json* configuration file
- Name of an object for which an action was started was added to the drop-down list of jobs that appears when you click the job indicator (the bell icon)
- Eliminated ambiguity when using filters on the ***Bundles*** page
- Added the ability to go to API description from the drop-down list in the upper right corner of the web interface pages
- Added the ability to view job execution logs in the full-screen mode
- Detailed information on available versions was added to the cluster update dialog box
- Added the icon for quick access to parameter descriptions when working with object configurations. Hovering over the icon displays an expanded description of the selected configuration parameter

Bug fixes

- Fixed a problem with the frozen screen containing logs of a long-running job
- Fixed display of a service information
- Fixed the error that occurred when mapping services to hosts and caused changes in the component order
- Clicking ***All hosts*** led to the selection of all hosts including those that were in maintenance mode
- Fixed a bug that caused an empty tooltip to be displayed when selecting mandatory configuration parameters

Misc/Internal

- Upgraded the `urllib3` library to 2.5.0
- Upgraded the `requests` library to 2.32.5
- Upgraded the `social-auth-app-django` library to 5.5.1
- Removed the `pip` library
- Upgraded the `Django` library to 5.2.7
- Removed the `sqlite-libs` library
- Upgraded the `setuptools` library to 80.9.0
- Removed the `pycryptodome` library
- Upgraded the `cryptography` library to 45.0.7

2.8.0

Date: 30.09.2025

New features

- Added support for Ansible 2.16
- Implemented the new config_apply function in the prototype `script` parameter
- Added the ability to reuse a host in different clusters

Improvements

- The configuration page was redesigned
- Cluster actions with the `host_action: true` parameter were added to the list of host actions on the ***Hosts*** tab of the cluster page

Bug fixes

- Fixed the error with incorrect information in logs on the ***Ansible [stdout]*** tab
- Fixed the error that occured when trying to download logs of an action host group
- Fixed the error that occurred when comparing configuration changes made using the `adcm_config` module
- Fixed a bug that caused the modal window with an action to close when the previous action was completed

Misc/Internal

- Fonts used in ADCM are stored locally and not downloaded from the Internet
- Deprecated API v1 was removed from ADCM
- The Yandex Cloud hostprovider bundle version 3.9 and below is no longer supported

2.7.1

Date: 23.07.2025

Bug fixes

- Fixed the error that occurred if a bundle contained a job after removing a service by the `adcm_delete_service` module

2.7.0

Date: 04.07.2025

Improvements

- Improved flag management
- Added the ability to create a configuration revision

Bug fixes

- Updated the API v2 schema to bring it to a consistent state
- Fixed the error with the status code `500` for an action containing the `config_jinja` parameter with an empty default value
- Fixed the `adcm_min_version` parameter check when uploading a bundle
- Fixed a bug that occurred when uploading a bundle with an empty configuration
- Fixed the display of the ***Sync interval*** configuration parameter that could contain a negative value

Misc/Internal

- In the `bundle_revert` function, added the ability to recover deleted objects
- Refactored the bundle upload mechanism
- Extended the `hc_apply` functionality
- SQLite is no longer supported
- Changed formats of responses for ADCM API v2 listed below.

  Changes

  - Response with the status code `401`:
    - Old format: `{"detail": "Authentication credentials were not provided."}`.
    - New format: `{"code": "API_ERROR", "level": "ERROR", "desc":"Authentication credentials were not provided."}`.
  - Response with the status code `403`:
    - Old format: `{"detail": "You do not have permission to perform this action."}`.
    - New format: `{"code": "API_ERROR", "level": "ERROR", "desc": "You do not have permission to perform this action."}`.
  - Response with the status code `404`:
    - Old format: `{"detail": "Not found."}`.
    - New format: `{"code": "API_ERROR", "level": "ERROR", "desc": "Not found."}`.

2.6.0

Date: 03.04.2025

New features

- Introduced restriction for unsigned bundle uploads
- Introduced the unsafe option for Ansible inventory parameters of the `string` type
- Added the groups variable to the Jinja2 template rendering context

Improvements

- Corrected UI scaling issues on high-resolution monitors
- Eliminated host variable duplication in inventory
- Disabled legacy TLS versions in the nginx configuration

Bug fixes

- Corrected display of the empty service configuration
- Fixed incorrect schema path resolving during upgrade
- Resolved the action host group termination error
- Fixed the cluster blocking issue when running an action on the hostprovider
- Fixed the service blocking issue when the host is blocked
- Fixed the issue of pagination number reset when selecting a previously saved configuration
- Prevented display of unnecessary outdated configuration flags
- Fixed the cluster statuschecker accuracy after the upgrade
- Corrected visibility of the ADCM Ansible settings

Misc/Internal

- Optimized the ADCM image build process
- SQLite support will be discontinued from ADCM 2.7.0
- API v1 will be deprecated and removed from ADCM in the third quarter of 2025

2.5.1

Date: 26.02.2025

Bug fixes

- Fixed the issue of calculations for collecting statistics when sending statistics is disabled
- Fixed the issue with updating already installed old versions of ADCM

2.5.0

Date: 28.12.2024

New features

- Added ability to launch actions in non-blocking mode which allows running actions in parallel or executing long actions without fully blocking objects

Improvements

- Redesigned the job page
- Upgraded Django to 5.1.1 requiring PostgreSQL 13 or later

Bug fixes

- Fixed the template loader issue for the Jinja environment
- Fixed a hostprovider archive naming to include the hostprovider details in downloaded logs
- Fixed a service deletion error occurred when removal was wrongly blocked by remaining tasks in the `CREATED` status
- Corrected API v2 schemas listed below.

  Changes in schemas

  - Corrected the schema for */api/v2/adcm/config-schema/*.
  - Updated the `startTime` and `endTime` fields in task and job descriptions to be used as nullable type.
  - Adjusted the `200 OK` HTTP response code for */api/v2/clusters/<clusterId>/mapping/components/*.
  - Fixed schema definitions for */api/v2/clusters/<clusterId>/ansible-config-schema/*.
  - Removed the `serviceId` field of the query type from */api/v2/clusters/<clusterId>/service-candidates/*.
  - Ensured that the `objects.type` field enum value matches `action_host_group` for */api/v2/tasks/*.
  - Removed filters from the POST and PATCH methods for */api/v2/clusters/<clusterId>/hosts/* and */api/v2/rbac/roles/<roleId>/* to streamline request handling.
  - Fixed response examples for */api/v2/rbac/roles/<roleId>/object-candidates/*.
  - Fixed response examples for */api/v2/rbac/roles/categories/*.
  - Added the ability to manage services in bulk for */services/*.

Misc/Internal

- Backported the hashi_vault plugin for Ansible 2.9
- Corrected the WebSocket event name from `create_provider_concern` to `create_hostprovider_concern`
- For consistency, renamed the models as follows:
  - `ClusterObject` → `Service`.
  - `ServiceComponent` → `Component`.
  - `ConfigGroup` → `ConfigHostGroup`.

2.4.0

Date: 24.10.2024

New features

- Added host sorting by state in the UI
- Extended logging settings to set log levels for each log type

Improvements

- Optimized the audit code; improved performance and maintainability
- Enhanced the APIv2 logic for creation, deletion, and distribution of concerns
- Refactored dump-restore to support the `secretfile` type and the maintenance mode
- Implemented consistent host sorting on the host-component mapping page

Bug fixes

- Fixed the configuration parameter type `file`/`secretfile` in the action config to pass content instead of paths
- Enabled min/max constraints for integers in *config_jinja* to resolve server errors with the status code `500`
- Fixed the issue with missing operations in the *audit.log* files in CEF format
- Fixed the service dependency errors after adding components
- Resolved issues with activatable groups in *scripts_jinja*
- Fixed the multiple bundle upload issue
- Fixed the incomplete Ansible log output in the UI
- Fixed auto-reloading the service configuration page
- Eliminated the possibility of multiple uploads of the same bundle in parallel sessions
- Eliminated the possibility of saving the host-component maps with components on hosts in the maintenance mode

Misc/Internal

- Extended statistic collection
- Removed the deprecated logrotate functionality
- Starting with ADCM 2.5.0, the minimum required version of PostgreSQL will be 14

2.3.0

Date: 05.09.2024

New features

- Introduced action host groups, which are designed to launch actions on several hosts added to a group

Improvements

- Updated the behavior of the `outdated_config` flag for objects with the `created` status

Bug fixes

- Flags are now correctly removed from hosts when a cluster is deleted
- The `pipelining` option now defaults to `False`
- Fixed SSL error for external DB

2.2.0

Date: 05.07.2024

New features

- Added options for Ansible settings
- Added support for external databases with SSL configuration
- Added support for configuring UWSGI options in a separate settings file
- Added options for service-to-host mapping
- Added auto-scroll for the ***Jobs*** section
- Implemented the ability to dynamically generate the list of action sub-actions

Improvements

- Reworked ADCM flags
- Added logging of the client version to audit logs
- Added options for removing `admin` user
- Optimized the host creation operation

Bug fixes

- Fixed error related to removing hosts
- Fixed issues related to displaying information regarding permissions
- Fixed validation issues

Misc/Internal

- Added the `pattern` attribute for string-based types
- Separated logic for working with ADCM objects and Ansible
- Optimized logic for the status server
- Reworked API v2 schema
- Reworked statistics service
- This version of ADCM does not support ET prior to v2024070400

2.1.2

Date: 20.05.2024

Bug fixes

- Fine-tune LDAP stability

2.1.1

Date: 11.04.2024

Bug fixes

- Fixed migration error related to the `ui_options` field
- Fixed config group name conflicts across clusters
- Fixed LDAP authentication error by updating the user search base

2.1.0

Date: 20.03.2024

New features

- Introduced a user blocking feature
- Enhanced LDAP integration and permissions management
- Added a new UI loading tool

Improvements

- Implemented a bulk operation for adding hosts to a cluster
- Enhanced the display logic for Ansible logs
- Refactored inventory generation
- Implemented idempotence in the `adcm_config` module

Bug fixes

- Fixed an IPv6 issue
- Fixed an error occurring during the upgrade of SSH bundles
- Fixed an error when adding a duplicate host to a configuration group
- Fixed an error during configuration posting via API v2
- Enhanced security by ensuring that only superusers can view LDAP-related tasks
- Fixed an error when saving a host-component mapping with duplicate links
- Fixed unauthorized changes to maintenance mode for services and components
- Resolved an issue that prevented the removal of a user from all groups via the API

Misc/Internal

- Fixed an error encountered when posting imports twice

2.0.0

Date: 25.12.2023

New features

- New UI is introduced
- ADCM Auditor role is added

Improvements

- ADCM API 2.0 is now recommended for production use
- Functionality for failed user login attempts reset has been upgraded to user unblocking
- Improved performance of adding a service. Now this operation includes bulk operation logic

Misc/Internal

- ADCM versioning rule is changed. The new version is 2.0.0
- Login length in audit records is limited to 150 characters
- Audit module log level is set to `INFO`

2023.11.28.07

Date: 28.11.2023

New features

- IP addresses collection in audit logs is supported

2023.10.10.08

Date: 10.10.2023

Bug fixes

- Requirement for invisible fields within structure is removed
- The issue caused by non-unique names during termination of an action run is resolved

2023.09.28.17

Date: 28.09.2023

New features

- Bundle signature check mechanism is implemented
- Added an optional feature for aggregating non-sensitive cluster details
- API v2 initial release is supported. Since it is in the alpha phase, note that its features may undergo further adjustments. API v2 is not yet recommended for use in production environments. For more information, please refer to the documentation

Improvements

- Role-based access model is based on groups instead of users

Bug fixes

- `adcm_config` does not work with the `integer` data type
- `required import` issues depend on the order in which they are declared
- `file` and `secretfile` types do not work for action configs
- Action run stops when the `before upgrade` field is generated in the inventory
- It is not possible to manually tick the ***ADCM Administrator*** checkbox for an LDAP user

Misc/Internal

- Adding invisible fields is implemented for the `structure` config parameter type
- `config_jinja` is supported for the `task` action type

2023.07.10.17

Date: 10.07.2023

Bug fixes

- Passing `NonType (null)` to inventory instead of an empty `map` configuration parameter
- Incorrect variables in inventory for the `*.maintenance_mode` groups if the host is not added to the configuration group
- License does not show up during bundle upgrade
- `adcm_hostid` and other variables are missing when a host is placed in Maintenance Mode
- VM initialisation does not start due to the `six` package being absent

2023.06.14.16

Date: 14.06.2023

New features

- Brute-force protection mechanism is implemented
- Password policy is changed

Improvements

- RBAC functionality is improved

Bug fixes

- Variables defined in the `CLUSTER` group are not available for a host in maintenance mode
- Wrong order of objects on the ***Hosts - Components*** page
- The ***Hosts - Components*** page is not updated in case of reverting an upgrade
- `500` error when trying to upload a bundle that contains similar action names
- Wrong order of actions within a job
- Host in maintenance mode is not included in `target`
- Different event identifiers for SQLite and PostgreSQL
- Empty configuration parameter cannot be saved in a configuration group
- Database errors after ADCM upgrade
- `Gateway timeout` error after adding a new policy

Misc/Internal

- Service dependency mechanism is implemented
- `display_name` is added for upgrades
- Added ability to commit changes in `hc_map` during an operation being executed
- Added ability to manage activatable group state via `adcm_config`
- Added ability to access previous parameters during the upgrade

2023.04.21.12

Date: 21.04.2023

Bug fixes

- Prolonged update procedure for a cluster with RBAC policy

2023.04.10.22

Date: 10.04.2023

Bug fixes

- Incorrect order of starting subactions within actions

2023.03.27.20

Date: 27.03.2023

New features

- Ability to use an external database to store metadata is added
- Data migration from SQLite to PostgreSQL is supported. PostgreSQL database is recommended for use in the product environment. SQLite database is intended for experimental purposes

Improvements

- Time taken to switch maintenance mode on and off is reduced
- Time taken to save configs and config groups is reduced
- ADCM documentation link is updated
- Django is updated to the latest version (3.2)
- Bell icon user experience is improved

Bug fixes

- Critical error on the server when a user logs in via LDAP
- Incorrect inheritance of maintenance mode
- Unable to remove a component from the host in maintenance mode
- Empty list shows when the last item is deleted from the second or further page
- Trash button gives the `SERVICE_CONFLICT` error if the service has binded imports
- Job is not terminated after a cluster is deleted
- Some of the UI elements for hostproviders are broken
- Save page for the components map freezes
- Component concern link leads to the wrong page
- Adding a policy leads to a gateway-timeout error during every ADCM operation
- Cluster concern event is not sent when the cluster is created
- `500` error when trying to terminate a process that is already stopped
- ***Ask for help*** button leads to the outdated link
- Required message for empty fields does not show at all times it should
- Group config cannot be saved if a field is changed
- Error in URL when a job is selected via the bell button
- Expected service concerns are not displayed in the menu at the left side
- Config group parameter values are not synchronized between different inventory host groups
- Bell icon does not react to events that arrive from backend

Misc/Internal

- Bundle downgrade mechanism is implemented
- License agreement confirmation mechanism is implemented for services
- Mechanism for dynamic generation of configs is implemented for actions
- `secretfile` config parameter type is added
- `secretmap` config parameter type is added
- `skip` option (`allow_to_terminate`) is added for subactions

2022.12.26.18

Date: 26.12.2022

New features

- The ability to set the `password` and `secrettext` config parameters to empty

Bug fixes

- Column sorting does not work properly
- Lists of cluster and hostprovider bundles have duplicated entries after cluster or hostprovider is created

2022.12.07.10

Date: 07.12.2022

New features

- The ability to rename a cluster without recreating the object from scratch
- The download functionality is implemented for action logs
- The maintenance mode is implemented for services and components
- The ability to view audit logs via UI

Improvements

- The status aggregation logic is improved according to service/component statuses
- Ansible is upgraded to 2.9.27
- The ability to map the ***Maintenance mode*** button to an action
- The ability to map the ***Trash*** service removal button to an action

Bug fixes

- ***LDAP Sync*** action does not synchronize all users
- When a job is stopped, the `stderr` and `stdout` files are blank
- Filters for groups in ADCM LDAP integration settings do not work for users in those groups
- When a bundle is uploaded, there is an error of the action name being too long
- Change the name of the link to hostprovider at the ***HOSTS*** tab
- Issue does not disappear after a cluster is upgraded

Misc/Internal

- The ability to forward the display name of services and components in inventory

2022.10.04.17

Date: 04.10.2022

New features

- The ability to generate logs in CEF format is added to support the security audit functionality
- The user deactivation functionality is substituted for user deletion
- The ability to generate inventory host groups for hosts in maintenance mode is added
- The ability to rename clusters and hosts via API
- The ability to download action logs in bulk via API

Improvements

- UI: several fixes in *group-config*

Bug fixes

- Backend: the `500` error when launching action with config parameter that is able to be activated
- Backend: the `500` error when uploading bundles with similar upgrade names
- Backend: the `CLUSTER_SERVICE_NOT_FOUND` error in the ***JOBS*** page when pagination size equals 50 or 100
- Backend: the config parameters of different objects are not synchronized across config groups
- Backend: the service delete functionality works regardless of state
- Backend: the config group customization does not affect the service/component configuration
- UI: the concern link leads to the object details page, yet not to the page with the concern owner

2022.08.10.17

Date: 10.08.2022

New features

- The user authentication via Active Directory/LDAP is supported
- The multiphase bundle upgrade functionality is added
- The ability to remove deprecated components and services during upgrade is added

Bug fixes

- Backend: the brackets are not allowed in the upgrade name in case of pre/post-logic used
- Backend: the spaces are not allowed in the bundle name in case of upgrade pre/post-logic used
- Backend: the inability to save a group config containing an empty parameter of a list type
- Backend: the passwords in ADCM logs are unmasked

2022.05.26.14

Date: 26.05.2022

New features

- The maintenance mode is supported for hosts
- The support of group config for parameter types that are able to be activated is added
- The ability to remove services that are in the `created` state is added

Improvements

- UI: the possibility to include the configuration field in the *group-config* in case of `config_group_customization=False`, `advanced: true`, and not only

Bug fixes

- Fixed: the inability to save configuration if component is not selected within the constraint of `[0, odd]`
- Backend: the service configuration group ignores the configuration and presence of collocated services
- Backend: the `500` error is received when launching action
- Backend: expand on the locked hosts
- Backend: the `GROUP_CONFIG_CHANGE_UNSELECTED_FIELD` error
- Backend: the group configuration changes are not saved after the ***Save*** button is clicked
- UI: the group configuration field should be enabled

2022.04.18.13

Date: 18.04.2022

New features

- The multiphase bundle upgrade functionality is added
- The Mitogen support is disabled
- The support of static jinja2 files during the bundle build is added for ADCMClient

Bug fixes

- Backend: fix the Ansible *service_facts* module
- Backend: missing Ansible Vault secrets when importing/exporting clusters

2022.03.25.16

Date: 25.03.2022

New features

- The filtering of clusters, services, hosts, and hostproviders according to access rights
- The filtering of object's actions and their task logs according to access rights is implemented for backend

Improvements

- Multiple vulnerabilities eliminated (CVE)

Bug fixes

- Fixed: the weblinks are displayed incorrectly
- Backend: optimize the policy application algorithm
- Backend: the default value of the read-only configuration parameters does not change after the bundle upgrade
- Backend: the database is locked when the bundle is uploaded during the cluster installation
- Backend: `Internal server error` occurs when navigating to the ***JOBS*** tab
- Backend: the `min > max` import range allows to upload a bundle and make an import as if it was the `max > min` import range
- UI: turn off route redirection if permissions are denied

2022.02.01.06

Date: 01.02.2022

New features

- The RBAC functionality is added
- Ansible 2.9 is supported (as an alternative) for SDK
- The previous cluster state is enabled during upgrade
- The abort job functionality is enabled after deleting an object

Improvements

- The date, time, and log level are included in the *adcm_debug.log* file

Bug fixes

- Fixed: some bugs related to the *group-configs*
- Backend: the configuration parameters file resets to the file path instead of the file value
- Backend: the endless job after a cluster is deleted
- Backend: the service component is losing its own configuration during upgrade
- Backend: the inability to start a host action
- Backend: the import from another object of the same bundle is forbidden
- UI: the broken links on the ***JOBS*** tab
- UI: the ***Back*** button does not work on the ***JOBS*** tab
- UI: the job that is in progress does not refresh the bell icon
- UI: the failed jobs are shown before the succeeded ones under the bell icon
- UI: the default values for the list and map types are displayed instead of the current values

ADCM Installer releases

Adelya Zvyagintseva

Contents

To Table of Contents

1.1.1
1.1.0

1.1.1

Date: 25.11.2025

Bug fixes

Fixed loading the BusyBox image

1.1.0

Date: 25.09.2025

New features

Added the ability to configure and deploy OpenBao for identity-based management of secrets and encryption

Added the ability to configure and deploy HashiCorp Consul in the dev mode