AD.S CONTROL

Arenadata Streaming Control

Arenadata Streaming Control (ADS Control) is a solution for managing Arenadata Streaming (ADS) clusters. It supports managing multiple ADS clusters that have the Kafka and Kafka Connect services installed. The manager provides the ability to create, edit, and delete Kafka connectors.

Version CURRENT Language: EN

Contents

1. Get started
   - Installation
     - Offline installation
       - Create a cluster
       - Add services
       - Add hosts to a cluster
       - Add components
       - Configure services
       - Configure a cluster
       - Import ET settings
       - Install a cluster
       - Integrate with ADS cluster
   - Start working with ADS Control

2. Access management
   - LDAP authentication

3. Cluster maintenance
   - Cluster actions
   - Service actions

4. Web user interface
   - UI pages
     - Overview
     - General
     - Topics
     - Brokers
     - Consumer groups
     - Kafka Connects
     - Schema-Registry
   - Usage examples
     - Mirror Maker 2
     - Debezium
       - PostgreSQL
       - MS SQL
     - Iceberg
     - SMT

5. References
   - ADS Control configuration parameters

6. Release notes
   - ADS Control releases
   - Version compatibility matrix

Installation

To Table of Contents

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

> ADS Control is only available in the Enterprise version.
>
> For ADS Control, only **offline** installation can be performed.
>
> An ADS Control cluster can be integrated with multiple ADS clusters, including clusters installed after ADS Control was installed. The ability to integrate with an ADS Control cluster is available starting with ADS 1.7.1.b1.
>
> It is recommended to switch off any antivirus software before ADS Control installation and during the ADS Control operation process. In case of using antivirus, please contact your antivirus software vendor to find out the settings necessary to allow ADS Control.

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

Installation of ADS Control can be performed before and after installing and configuring an ADS cluster with the monitoring service, except for the final step Integrate with ADS cluster, which can only be performed based on the added ADS cluster.

Below are the **offline** installation steps for ADCM, ET, ADS and monitoring (1-5), which are detailed in the ADS documentation, as well as installing ADS Control.

For ADCM, ADS, and monitoring, online installation can also be performed.

AD.CM

Step 1.

Install ADCM

AD.CM

Step 2.

Prepare hosts

AD.CM

Step 3.

Install ET cluster

AD.CM

Step 4.

Install ADS cluster

AD.CM

Step 5.

Install monitoring

AD.CM

Step 6.

Install ADS Control

Create a cluster

Olga Semme, Daria Barysheva

AD.S CONTROL

Contents

To Table of Contents

Step 1. Get the Enterprise version of a cluster bundle

Step 2. Upload a cluster bundle to ADCM

Step 3. Create a cluster based on the uploaded bundle

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

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

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

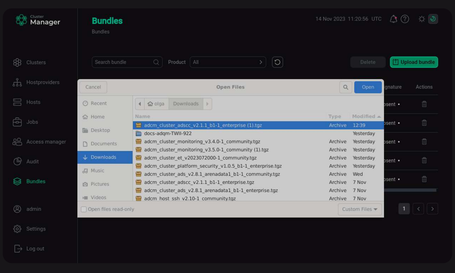
Step 1. Get the Enterprise version of a cluster bundle

Please get a cluster bundle from the Arenadata support team.

Step 2. Upload a cluster bundle to ADCM

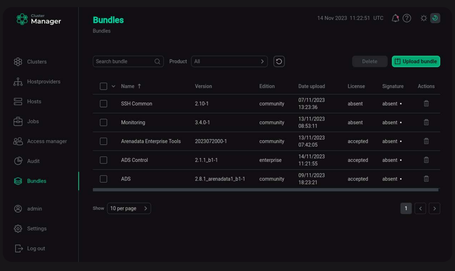
To upload a bundle to ADCM, follow the steps:

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



Upload a bundle

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



The result of successful uploading a bundle

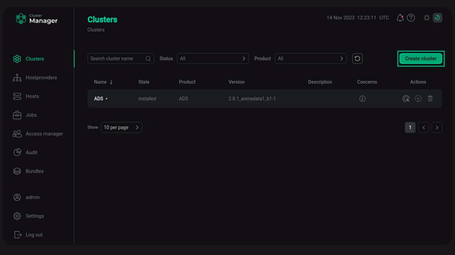
Step 3. Create a cluster based on the uploaded bundle

IMPORTANT

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

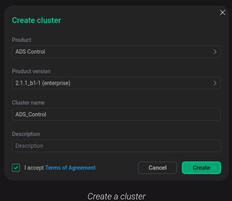
The steps for cluster creation are given below:

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



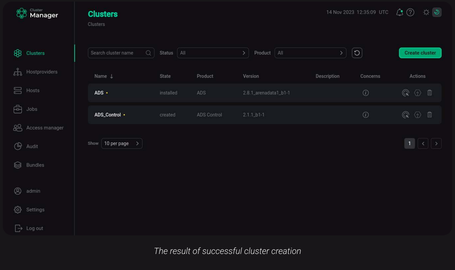
Switch to cluster creation

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



Create a cluster

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



The result of successful cluster creation

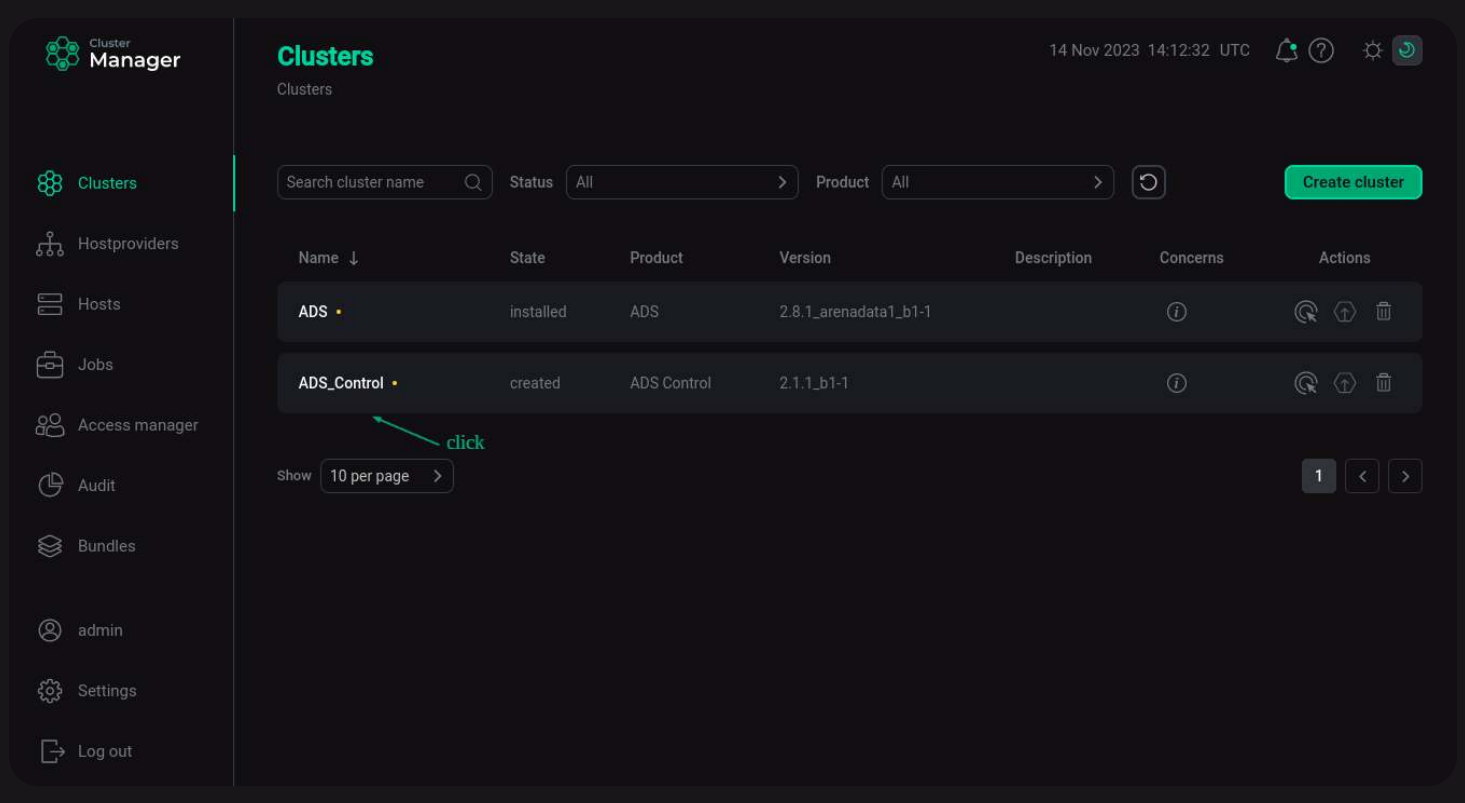
Add services

Olga Semme, Daria Barysheva

To Table of Contents

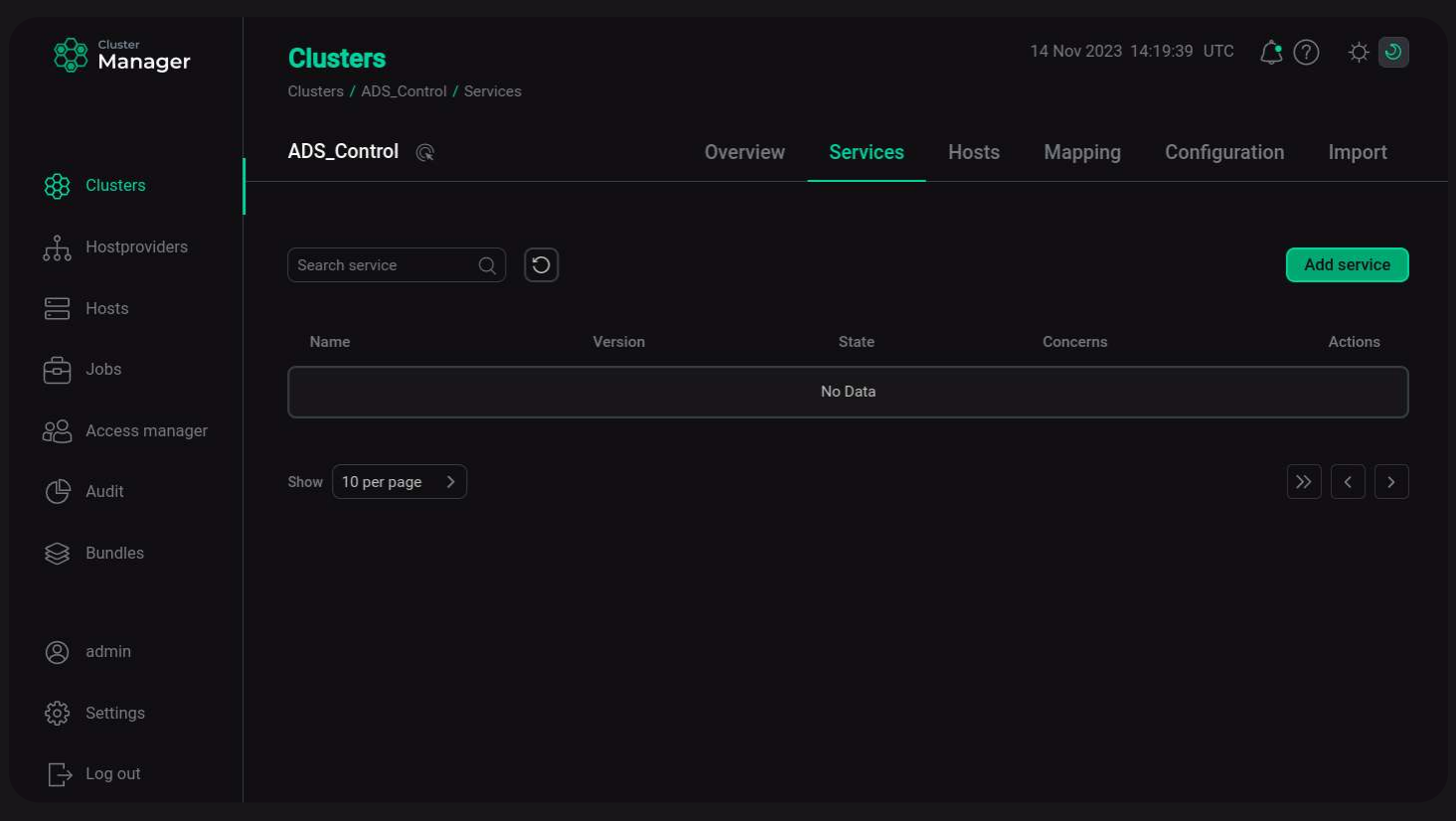
In ADCM a **service** means a software that performs some function. The steps for adding services to a cluster are listed below:

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



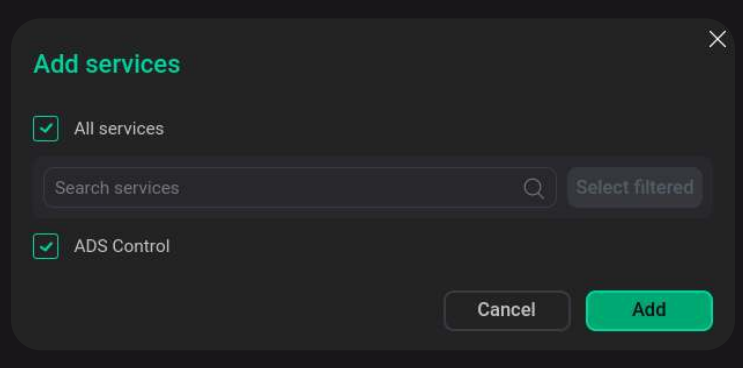
Select a cluster

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



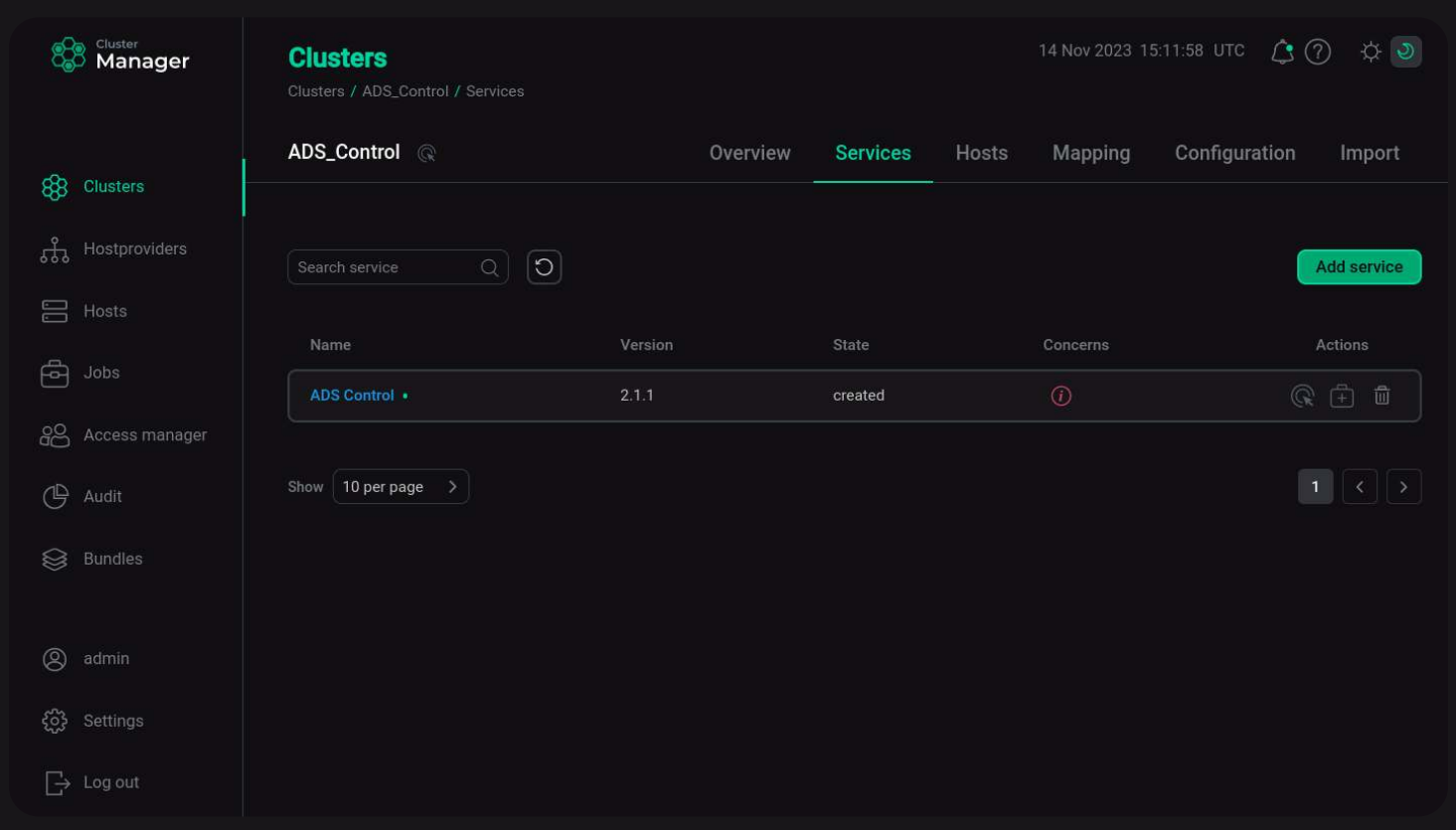
Switch to adding services

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



Select services

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



The result of successful adding services to a cluster

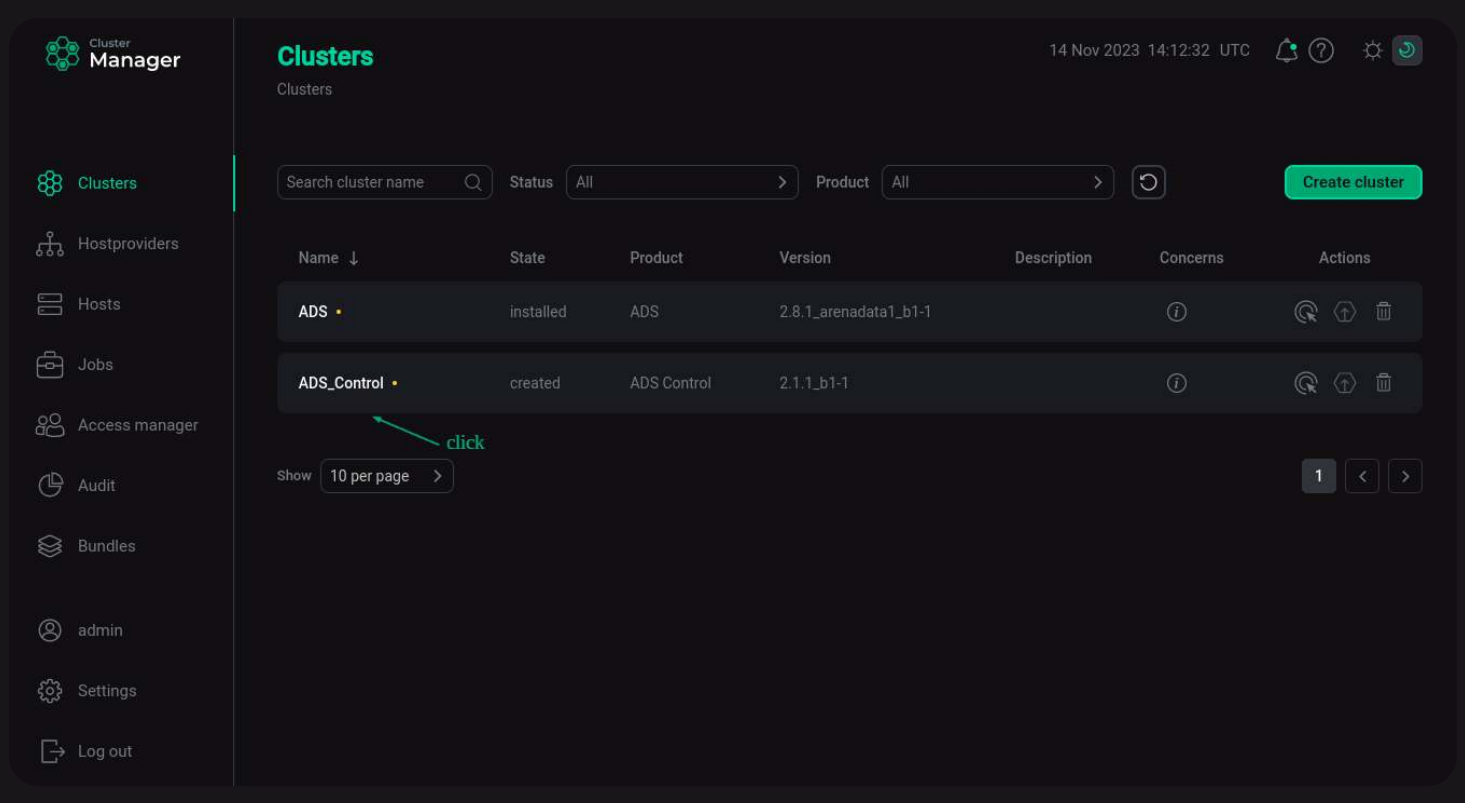
Add hosts to a cluster

Olga Semme, Daria Barysheva

To Table of Contents

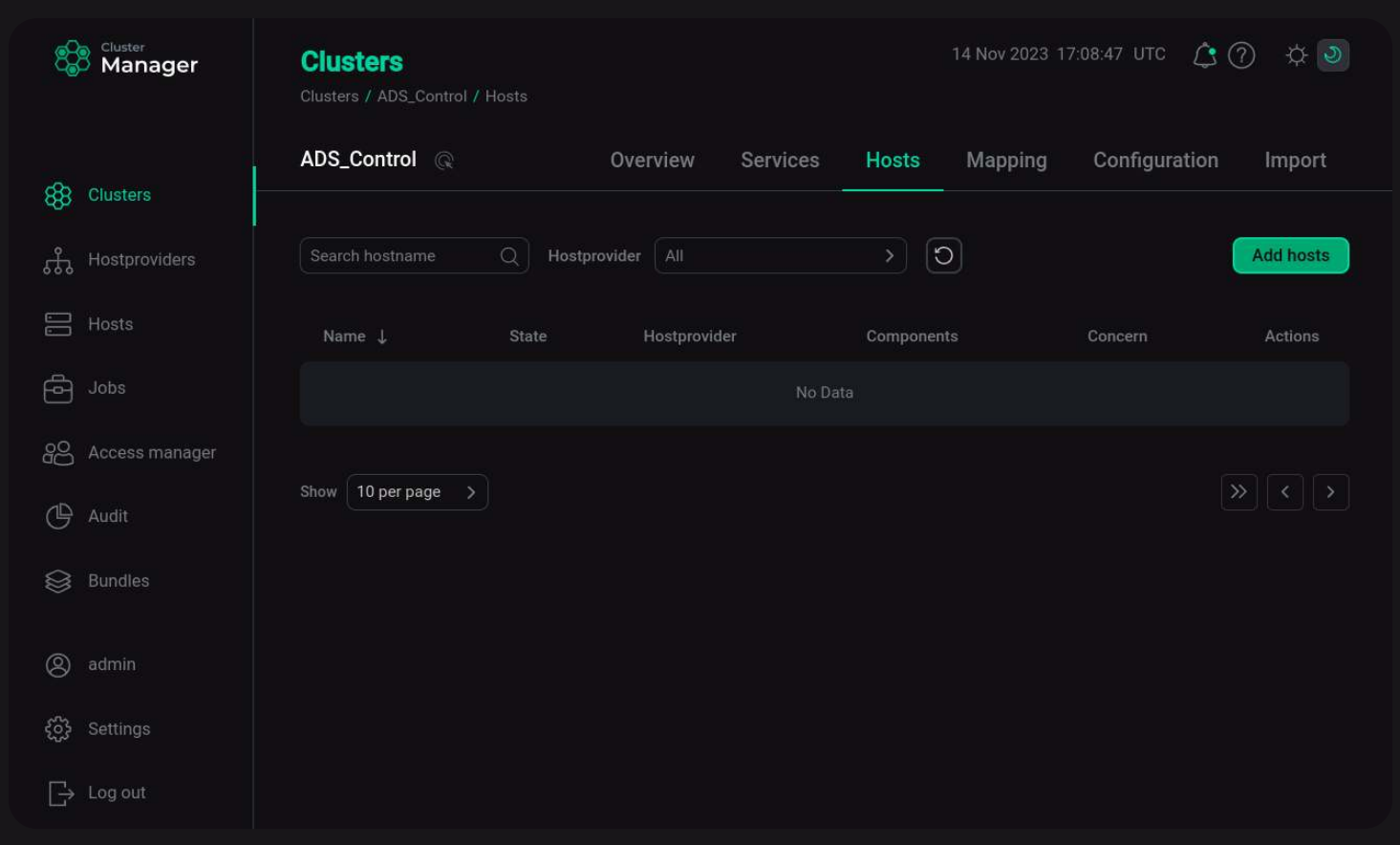
The steps for adding hosts to a cluster are listed below:

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



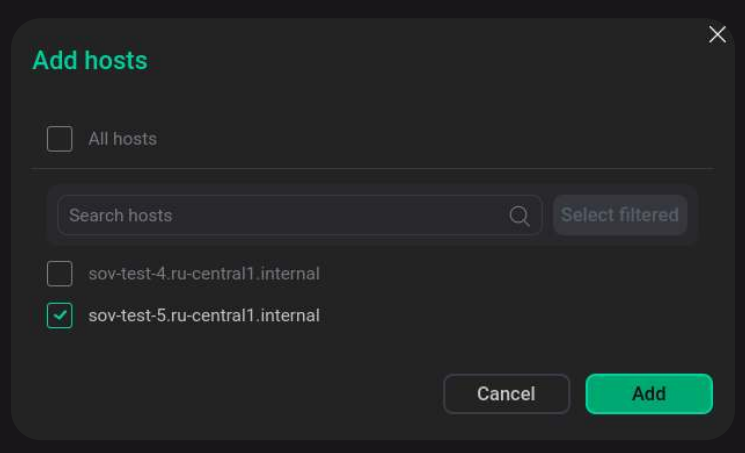
Select a cluster

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



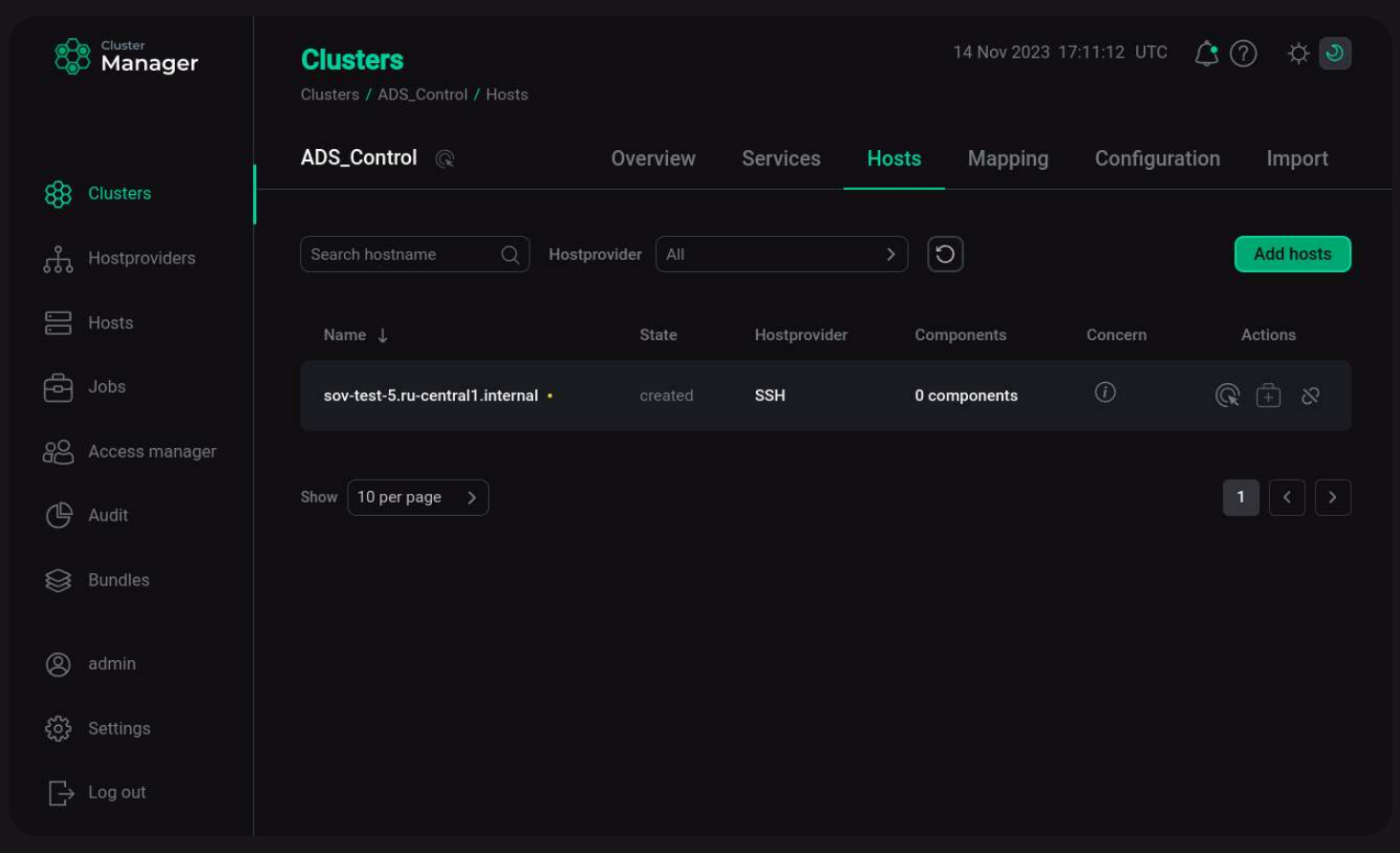
Switch to adding hosts

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



Select hosts

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



The result of successful adding hosts to a cluster

Add components

Olga Semme, Daria Barysheva

To Table of Contents

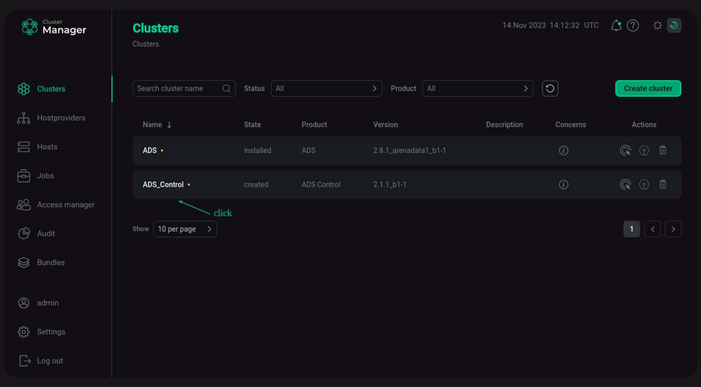
In ADCM a **component** means a part of a service that should be deployed on one or several cluster hosts. The ADS Control service consists of the ADS Control component.

IMPORTANT

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

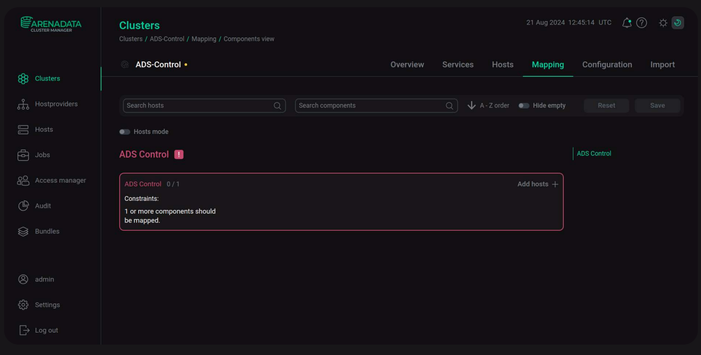
The steps for adding components to hosts are listed below:

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



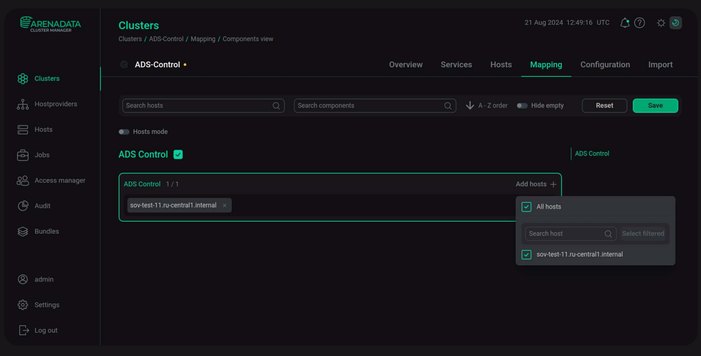
Select a cluster

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



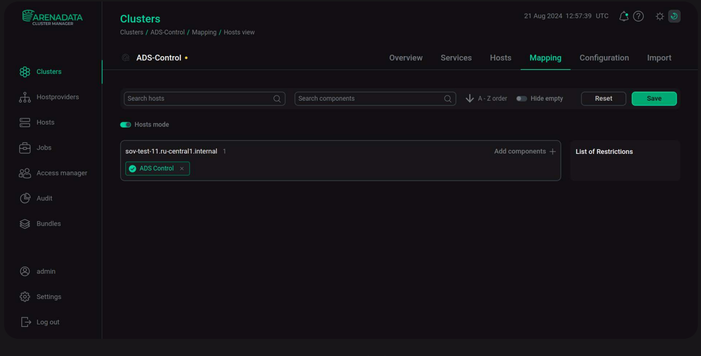
Switch to mapping service components

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



Save mapping of components

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



Use mapping by hosts

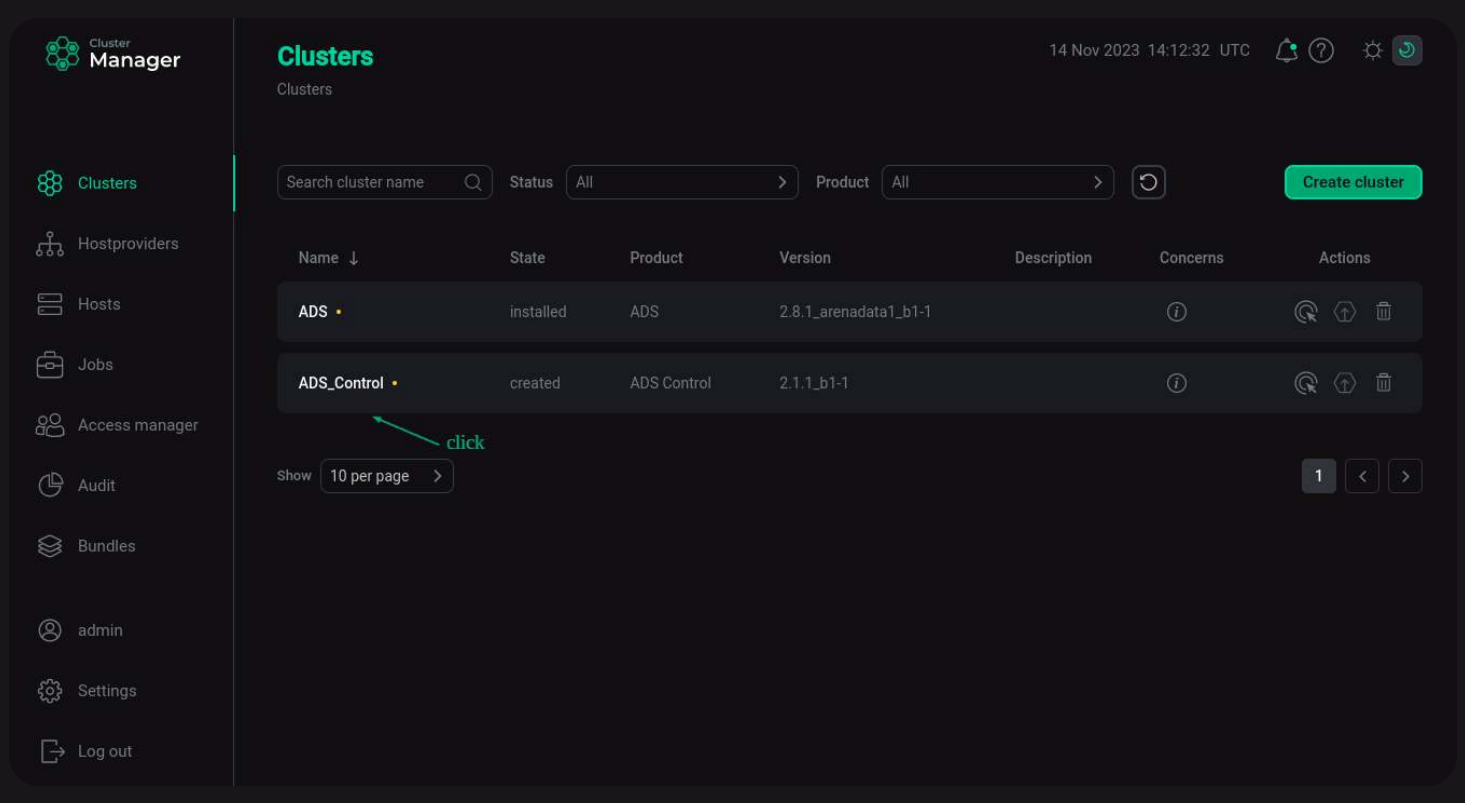
Configure services

Olga Semme, Daria Barysheva

To Table of Contents

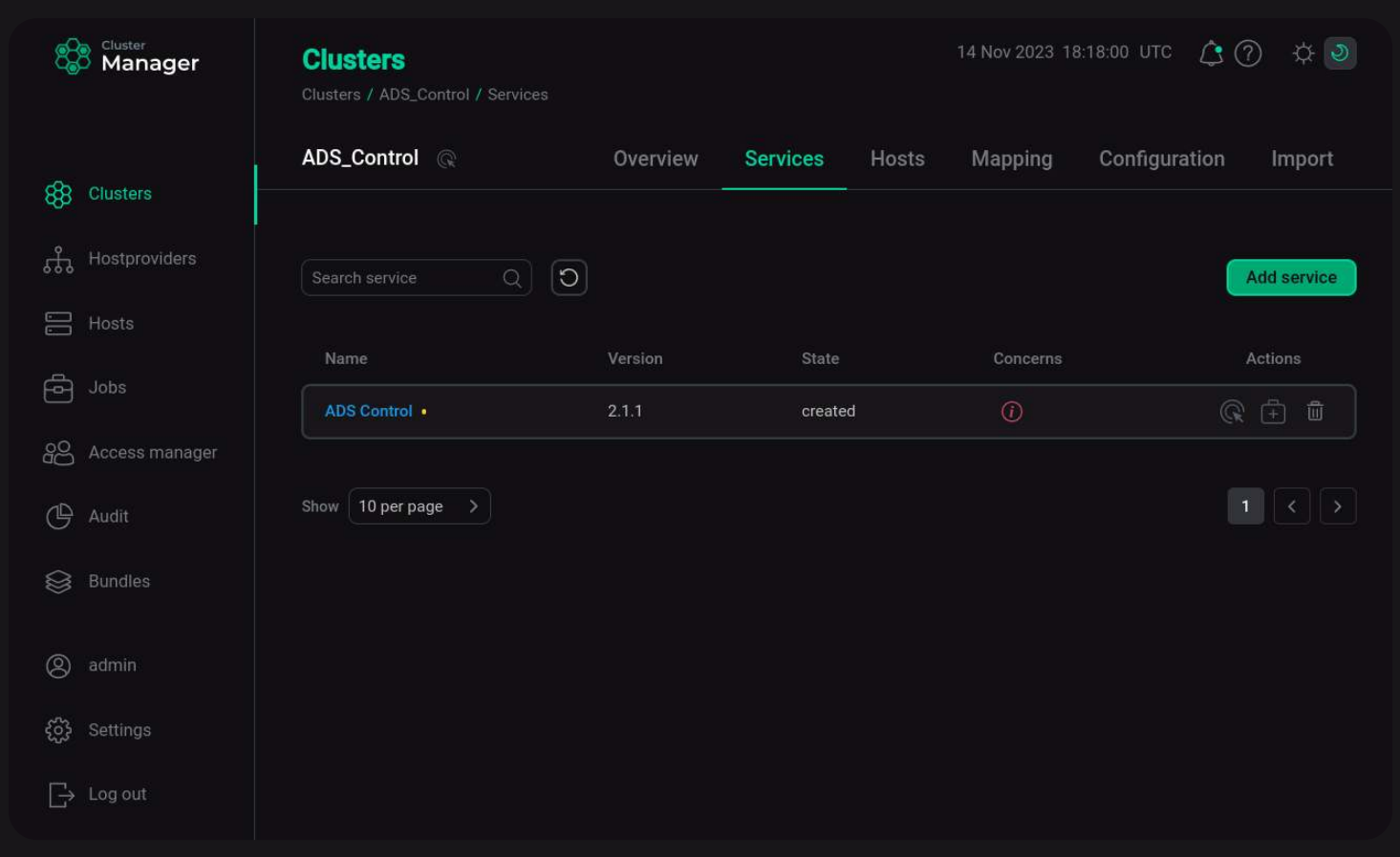
After adding all necessary services, you can configure its parameters. The steps for configuring cluster services are listed below:

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



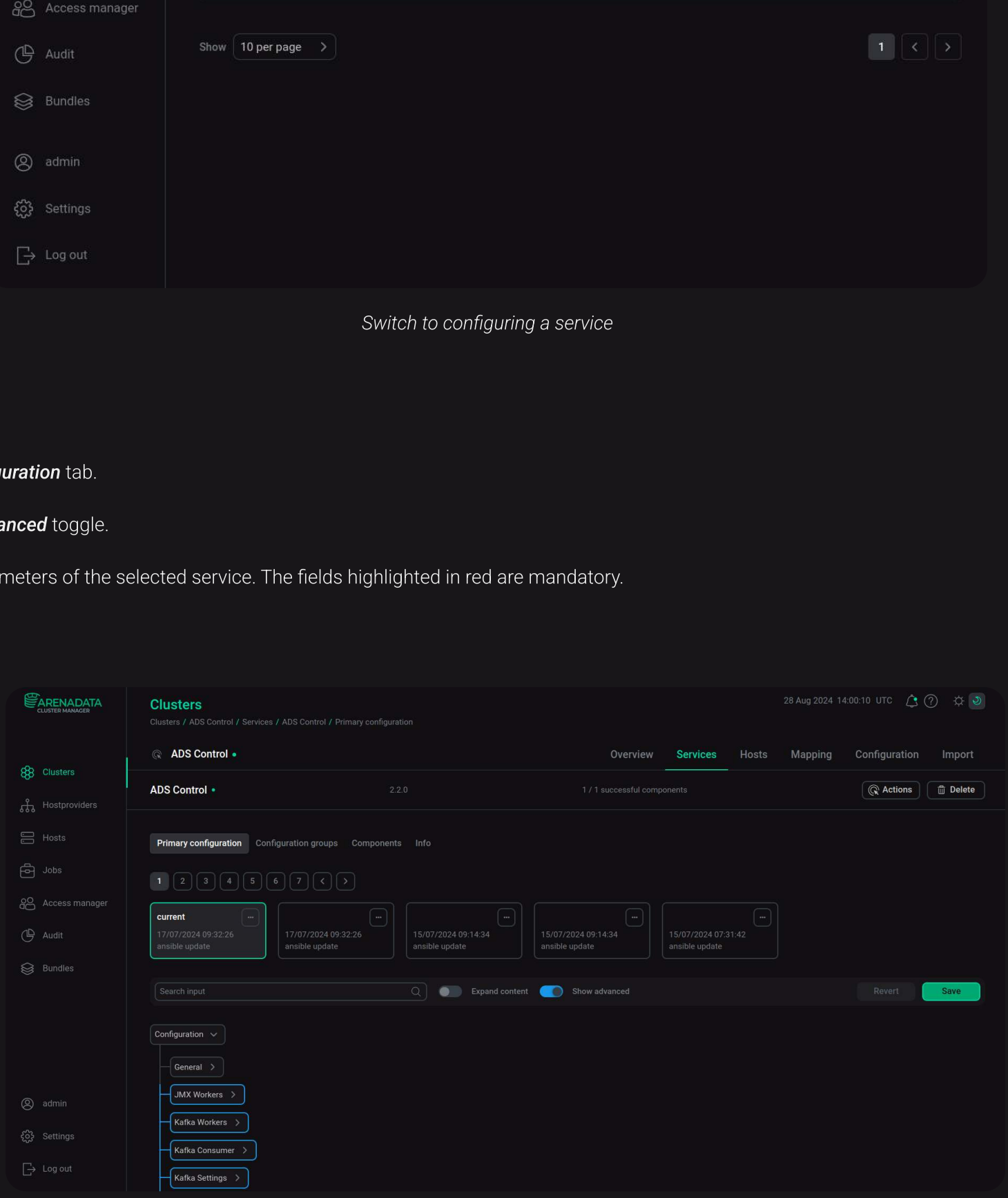
Select a cluster

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



Switch to configuring a service

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



Configure a service

NOTE

For more information about services configuration parameters, see ADS Control configuration parameters.

Configure a cluster

Olga Semme, Daria Barysheva

Contents

To Table of Contents

Primary configuration

Configuration groups

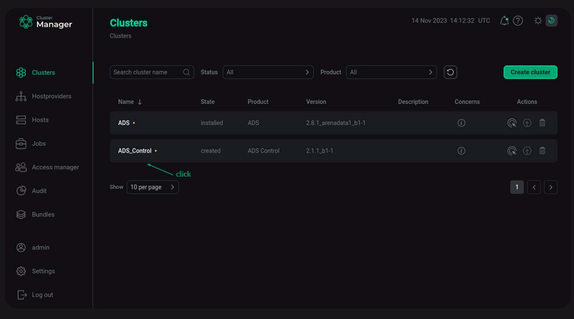
Ansible settings

IMPORTANT

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

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

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



Select a cluster

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

Primary configuration

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

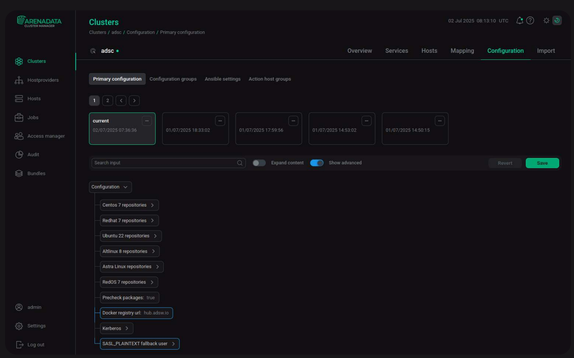
Cluster configuration parameters are listed below:

- Paths to the repositories that are used during the installation process depending on the cluster operating system (CentOS, Red Hat, Ubuntu, ALT Linux 8, or Astra Linux). Only the marked repositories are registered on the cluster servers.

Repository name	Parameter	Description	Default value
ADSCC	desc	Repository description	ADS Control repository
	url	Arenadata repository path	Specified depending on OS
	enabled	Enables use of the repository	true

- ***Precheck packages*** — enables package availability check before installation.
- ***Docker registry url*** — specifies the location of the Arenadata Docker Registry. Displayed if the ***Show advanced*** switch is activated.
- ***Kerberos*** — configuration for authentication using the Kerberos protocol. The description of Kerberos parameters for ADS Control corresponds to the description of parameters for working with Kerberos in ADS.
- ***SASL_PLAINTEXT fallback user*** — fallback user settings for use with imported ADS clusters with SASL PLAINTEXT authentication enabled (for ADS versions that do not import the SASL_PLAINTEXT user into the ADS Control cluster — from 3.3.2.2.b1 to 3.7.2.1.b1). Displayed if the ***Show advanced*** switch is activated.

Name	Description	Default value
Username	Username. A user with SASL PLAINTEXT authentication with the same name and password should be created in the ADS cluster to connect to Kafka. It is recommended to use a separate user for this purpose, not involved in regular (non-service) connections to Kafka	adscc
Password	User password	adscc-kafka-password



Cluster configuration window

Configuration groups

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

Ansible settings

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

Ansible settings

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

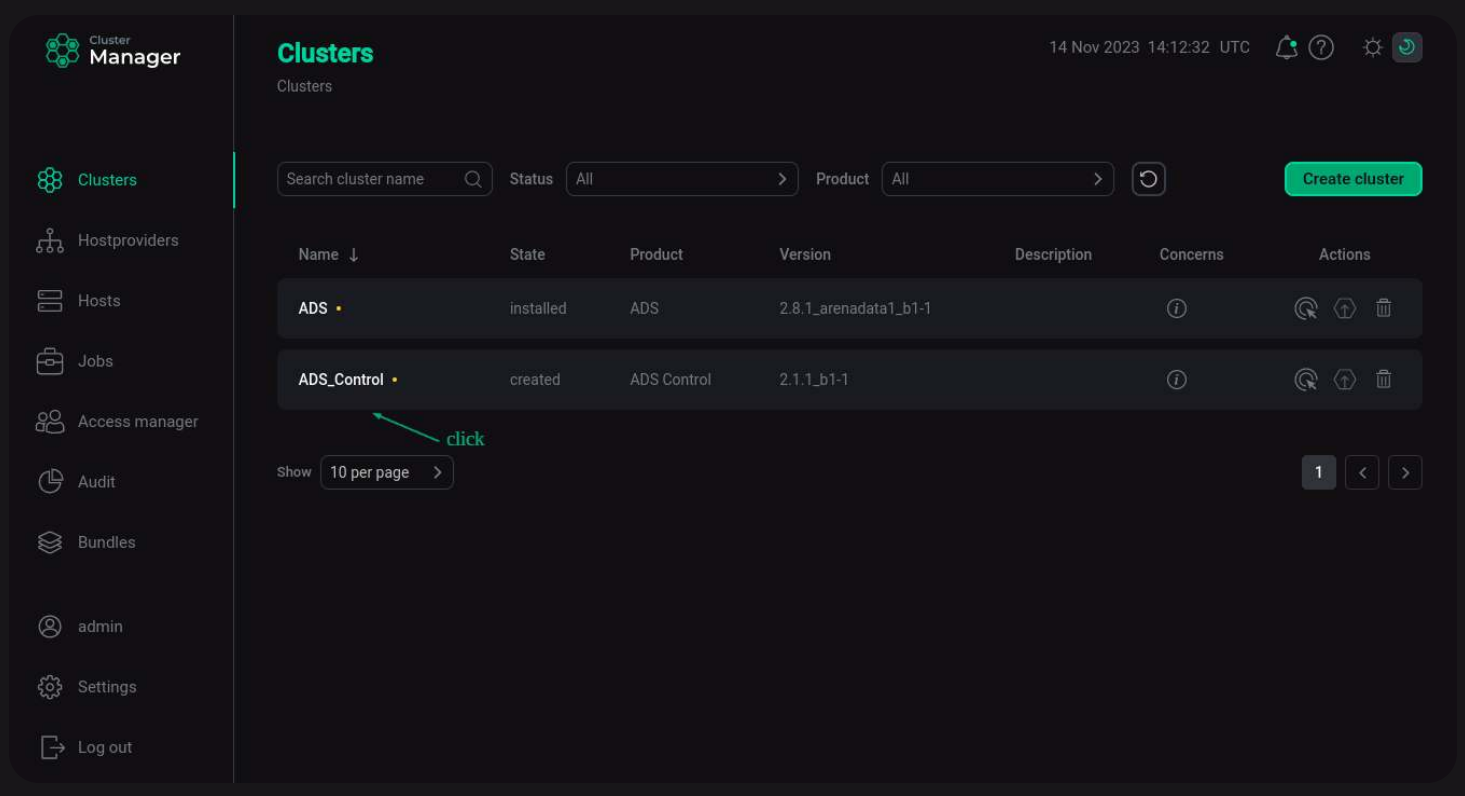
Import ET settings

Olga Semme, Daria Barysheva

To Table of Contents

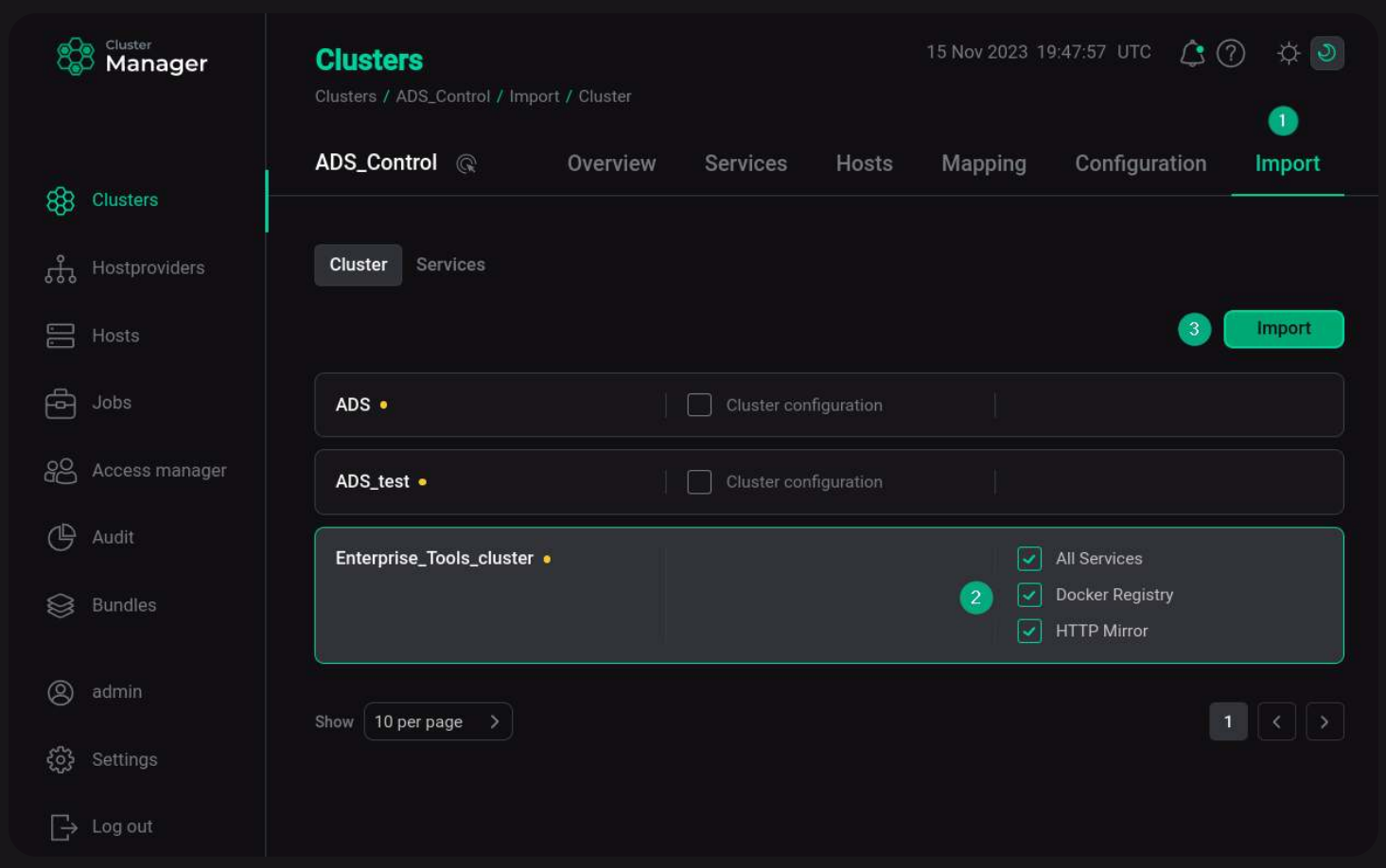
In order to determine that a new cluster should be installed offline, it is necessary to integrate it with the previously created Enterprise Tools (ET) cluster. Follow the steps:

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



Select a cluster

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



Integration with a ET cluster

NOTE

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

Install a cluster

Olga Semina, Daria Barysheva

ADS CONTROL

Contents

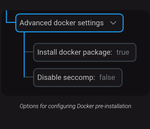
To Table of Contents

Prerequisites
Step 1. Run installation
Step 2. View the results of installation
Step 3. Check the cluster state after installation

Prerequisites

Before you install the ADS Control service on hosts with the CentOS 7 operating system, the current version of Docker must be preinstalled.

The Docker pre-installation is configured on the configuration page of the ADS Control service in the ***Advanced docker settings*** group (available after enabling the ***Show advanced*** switch).



Options for configuring Docker pre-installation

The configuration can be done in two independent ways.

First way:

1. Set the ***Disable seccomp*** parameter in the ***Advanced docker settings*** group to `true` and save the configuration.
2. Run the cluster installation as described below.

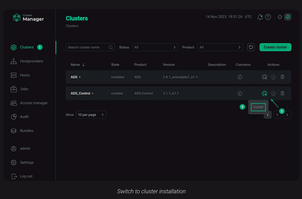
Second way:

1. Install Docker on the host according to the Install Docker Engine on CentOS instruction.
2. Install the *python-docker-py* package on the host.
3. Set the ***Install docker package*** parameter in the ***Advanced docker settings*** group to `false` and save the configuration.
4. Run the cluster installation as described below, with SELinux disabled on the host in installation options. After installation, reboot the host.

Step 1. Run installation

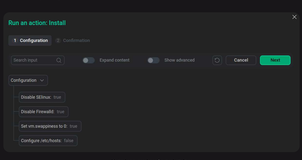
To install a cluster, you need to:

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



Switch to cluster installation

2. In the dialog box that opens, select the required options for installation.



Select parameters for installation

The purpose of the parameters is given below.

Parameter	Description
Disable SELinux	Disables SELinux on added hosts. In order for this setting to be applied, after the completion of the ***Install*** operation, you must restart the hosts manually
Disable Firewalld	Disables firewalld on hosts
Set vm.swappiness to 0	Disables swapping on hosts
Configure /etc/hosts	Writes added nodes to */etc/hosts* on cluster hosts. It is recommended to disable this option if DNS is configured

3. Verify the action in the opened window.

Verify the action

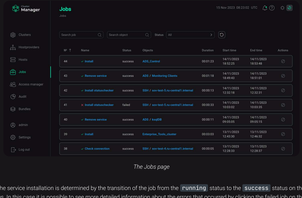
4. Wait until the job is completed.

NOTE

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

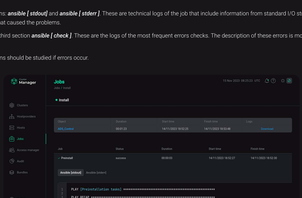
Step 2. View the results of installation

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



The Jobs page

The successful completion of the service installation is determined by the transition of the job from the `running` status to the `success` status on the ***Jobs*** page. If the job fails, it switches to the `failed` status. In this case it is possible to see more detailed information about the errors that occurred by clicking the failed job on the ***Jobs*** page.

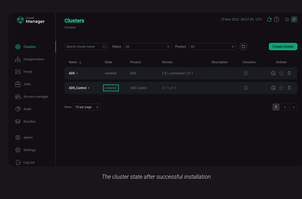


Switch to a job page

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

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

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

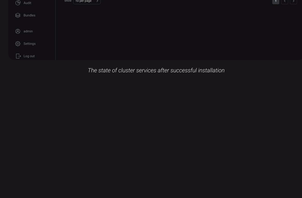


Job details

Step 3. Check the cluster state after installation

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

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



The cluster state after successful installation

- The service status displayed on the ***Services*** tab in the cluster menu changes from `created` to `installed`.

The state of cluster services after successful installation

Integrate with ADS cluster

Olga Semme, Daria Barysheva

To Table of Contents

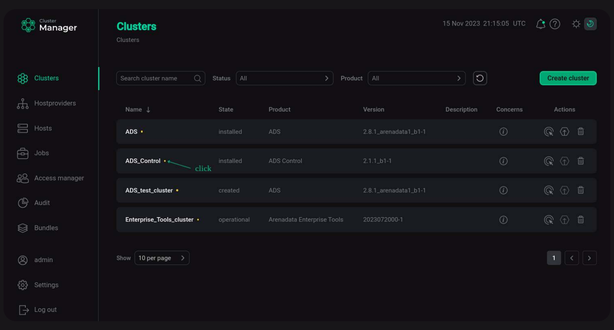
To set up an ADS Control cluster, you must integrate it with a pre-installed ADS cluster.

CAUTION

The ability to integrate with an ADS Control cluster is available starting with ADS 1.7.1.b1.

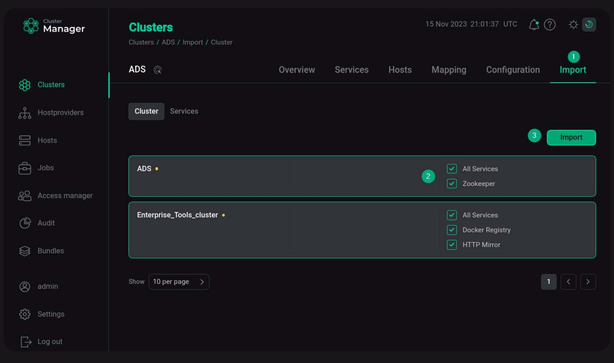
To do this, perform in your ADS Control cluster the following actions:

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



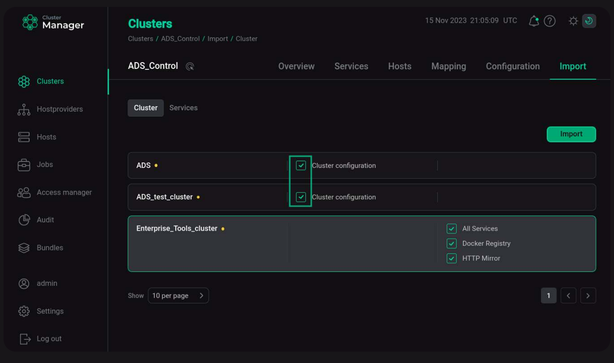
Select a cluster

2. Open the ***Import*** tab in the menu.
3. Select the desired ADS cluster.
4. Click ***Import***.



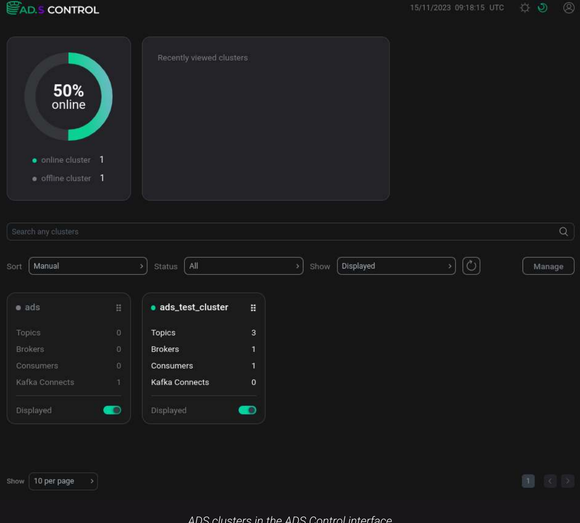
Integration with ADS cluster

Also, for one ADS Control cluster, it is possible to integrate with several ADS clusters.



Integration with multiple ADS clusters

As a result of successful integration in ADS Control user interface, you can manage all integrated ADS clusters. For more information about ADS Control features, see Start working with ADS Control.



ADS clusters in the ADS Control interface

Start working with ADS Control

Olga Semme

Contents

To Table of Contents

Installation

Connect to ADS Control

Manage users

Use ADS Control

Arenadata Streaming Control (ADS Control) is a solution for managing ADS clusters. Supports managing multiple ADS clusters that have Kafka and Kafka Connect services installed. Provides the ability to create, edit, and delete Kafka connectors.

Installation

Installation of ADS Control is described in the Get started with Arenadata Streaming Control article.

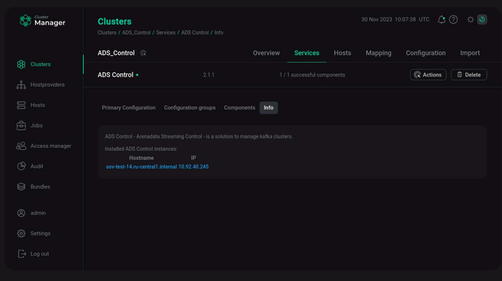
After installation, the ADCM interface displays the ADS Control cluster. All actions on ADS Control available in the ADCM interface are described in the articles:

- Cluster actions
- Service actions

Connect to ADS Control

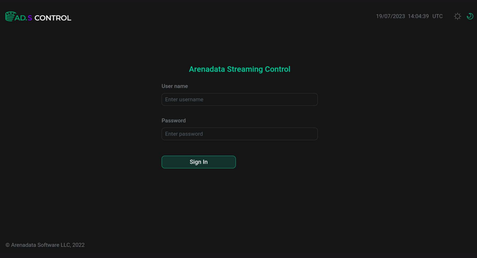
After installation, in the address bar of the browser, enter the address of the host where the ADS Control cluster is deployed. As the port number, specify the value of the ***server.http.port*** parameter from the ***General*** section, defined at the configuration stage of the ADS Control service when installing the ADS Control cluster.

Another way to connect to the ADS Control interface is to follow one of the links provided on the ***Info*** tab of the ADS Control service menu.



Link to connect to ADS Control

Next, in the login window that opens, enter your credentials (login and password) in the appropriate fields. The password for `admin` login is set on the configuration page of the ADS Control service in the ***Basic authentication*** group when installing the ADS Control cluster. After entering your credentials, click ***Sign in***.

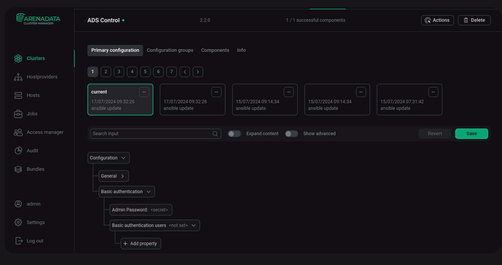


Login window

Manage users

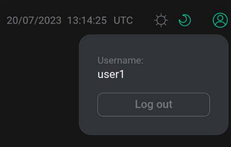
In order to create a user, do the following:

1. On the configuration page of the ADS Control service, in the ***Basic authentication*** group, click ***+ Add property*** to create a new user.



Creating a user for ADS Control

2. In the window that opens, enter the user name (field name), password (secret), confirm the password, and click ***Apply***.
   Usernames may include lowercase letters, numbers, periods, underscores, and/or hyphens.
3. Save the configuration by clicking ***Save***.
4. Restart the ADS Control service. To do this, apply the ***Restart*** action by clicking in the ***Actions*** column of the service.
   This gives the new user access to the ADS Control user interface. After authorization in ADS Control, the name of the new user is displayed in the authorization section.



User name display in ADS Control interface

In order to delete a user, on the configuration page of the ADS Control service in the ***Basic authentication*** group click next to the user you want to delete and click ***Save***.

To change user data, enter new data for the user and click ***Save***.

After deleting or each change made, restart the ADS Control service. To do this, apply the ***Restart*** action by clicking on the in the ***Actions*** column of the service.

Use ADS Control

The ADS Control user interface features are described in the ADS Control UI overview article.

At the moment, ADS Control allows you to configure Kafka Connect connectors, which are included in ADS by default for the 3.7.2.1.b1 version and above:

- ***MirrorCheckpointConnector***, ***MirrorHeartbeatConnector***, ***MirrorSourceConnector*** — connectors designed to replicate topics using the Mirror Maker 2 mechanism. For more information on working with these connectors in ADS Control, see Mirror Maker 2 in ADS Control.
- ***SqlServerConnector*** — connector designed to collect changed data in SQL Server databases and transfer it to Kafka. For more information, see Debezium connector for MS SQL Server.
- ***PostgresConnector*** — connector designed to collect changed data in PostgreSQL Server databases and transfer it to Kafka. For more information, see Debezium connector for PostgreSQL server.
- ***IcebergSinkConnector*** — connector designed for writing data from Kafka to Iceberg tables. For more information, see Iceberg connector.

General information on working with connectors in ADS Control is provided in the Work with Kafka Connect in ADS Control article.

LDAP authentication

Olga Semme

Contents

To Table of Contents

Overview

Enable LDAP authentication

Overview

Starting with 2.4.0 version of ADS Control, for connection to user interface, in addition to basic authentication users (required user `admin` and created users), users with an account in MS Active Directory can be used.

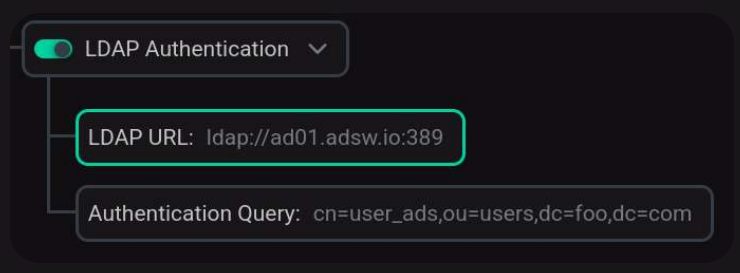
To configure authentication in the user interface, **distinguished name** (**DN**) — the user account in Active Directory — is used (for example, `cn=user_ads,ou=users,dc=foo,dc=com`). At the same time, **common name** (**CN**) is used as a login for authentication in the user interface (for this example, `user_ads`).

To configure authentication of several users, a DN for searching users in the `cn={0},ou=users,dc=foo,dc=com` format can be specified, where the index `{0}` is used to replace the CN of users. In this case, the CN of any user matching this DN is used as a login for authentication.

Enable LDAP authentication

To enable authentication for LDAP users, follow these steps on the configuration page of the ADS Control service:

1. Activate the ***LDAP Authentication*** switch and fill in the parameters in the drop-down list of parameters:
   - ***LDAP URL*** — URL of the LDAP server. The URL must begin with `ldap://` or `ldaps://` and include a port.
   - ***Authentication Query*** — query for searching users. Can be a full DN of a user (`cn=user_ads,ou=users,dc=foo,dc=com`) or DN for searching users (`cn={0},ou=users,dc=foo,dc=com`).



Enabling LDAP authentication

2. Save the configuration by clicking ***Save***.
3. Restart the ADS Control service. To do this, apply the ***Restart*** action by clicking on the icon in the ***Actions*** column of the service.

   As a result, the LDAP user or user group has access to the user interface of ADS Control. To authenticate in the authorization window, enter the user's CN and password for their account in MS Active Directory. After authentication in ADS Control, the user's CN is displayed in the authorization section.

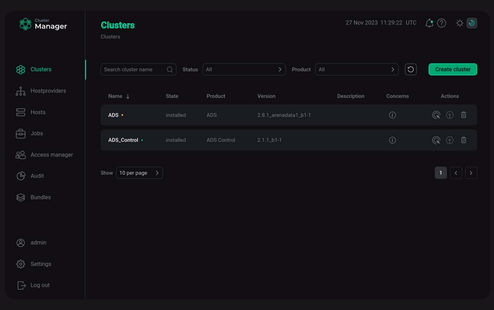
Cluster actions

Olga Semme

To Table of Contents

This article describes the main actions available in the ADCM interface for working with an ADS Control cluster.

Actions with the cluster are performed on the ***Clusters*** page.



Clusters page

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

The set of available actions depends on the cluster state:

- For new cluster (cluster state `created`) the ***Install*** action is available, which installs the cluster service.
- After cluster installation (cluster state `installed`) the following actions are available:
  - ***Check*** — checks the health of the cluster.
  - ***Restart*** — stops and then starts the cluster.
  - ***Start*** — starts the cluster services.
  - ***Stop*** — stops cluster services.
  - ***Manage Kerberos*** — allows you to enable, reconfigure, and disable Kerberos authentication.

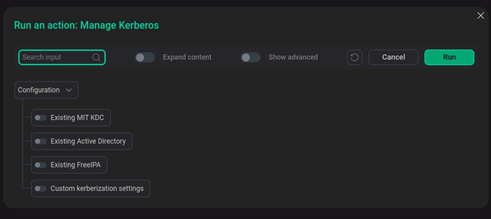
A description of how to enable and disable Kerberos for ADS Control, and all prerequisites for working with different Kerberos options, correspond to the descriptions of working with Kerberos in ADS:

- Existing MIT KDC
- Existing Active Directory

Every option of the ***Manage Kerberos*** action can be combined with the Custom kerberization settings option.

If Kerberos in ADS Control and ADS is configured with the participation of different key distribution centers (KDCs), for the two clusters to interact correctly, it is necessary to configure mutual trust between the KDCs.

After enabling Kerberos on ADS Control in the user interface, access to those ADS clusters on which Kerberos has already been configured appears. Before Kerberos is enabled on ADS Control, ADS clusters with authentication enabled are displayed in the interface in the offline status.



Manage Kerberos options

- ***Manage SSL*** — allows you to enable, reconfigure, and disable channel security via SSL protocol.

A description of how to enable and disable SSL channel security for ADS Control, and all prerequisites for working with the SSL protocol, correspond to the description of working with SSL in ADS, and ADS Control host certificates must trust the certificates of ADS hosts, that is, the *truststore* of each host in one cluster must contain the certificate **.crt* of each host in the other cluster.

Disable and re-enable SSL on a cluster using the ***Manage SSL*** action, if SSL on ADS was enabled later than on ADS Control.

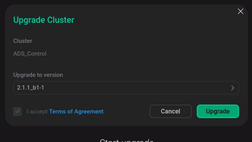
After enabling SSL on ADS Control in user interface, access to those ADS clusters on which SSL has already been configured appears. Before enabling SSL on ADS Control, already protected ADS clusters are displayed in the interface in the offline status.

- ***Reinstall status-checker*** — initiates reconfiguration and restart of statuschecker for all cluster services. Used when migrating a cluster under the control of a new ADCM server.

- For a cluster prepared for upgrade (the cluster state is `upgrade from X.X.X`), the ***Upgrade*** action is available, which upgrades the cluster version.

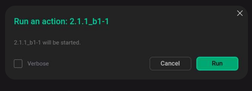
To update the cluster version:

1. Download ADS Control bundle new version — see step ***Upload a cluster bundle to ADCM*** in the Create a cluster article.
2. Open the ***Clusters*** page. After downloading the bundle, the icon becomes active.
3. Click on the icon . In the ***Upgrade Cluster*** window that opens, select the version available for upgrade and click ***Upgrade***.



Start upgrade

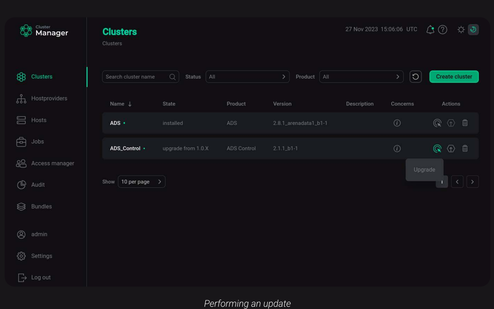
4. In the dialog box that opens, click ***Run***.



Confirmation of preparation for update

As a result, the status in the ***State*** column changes to `upgrade from XXX` (depending on the cluster version) — the cluster is ready for upgrade.

5. Click the icon in the ***Actions*** column and run the ***Upgrade*** action.



Performing an update

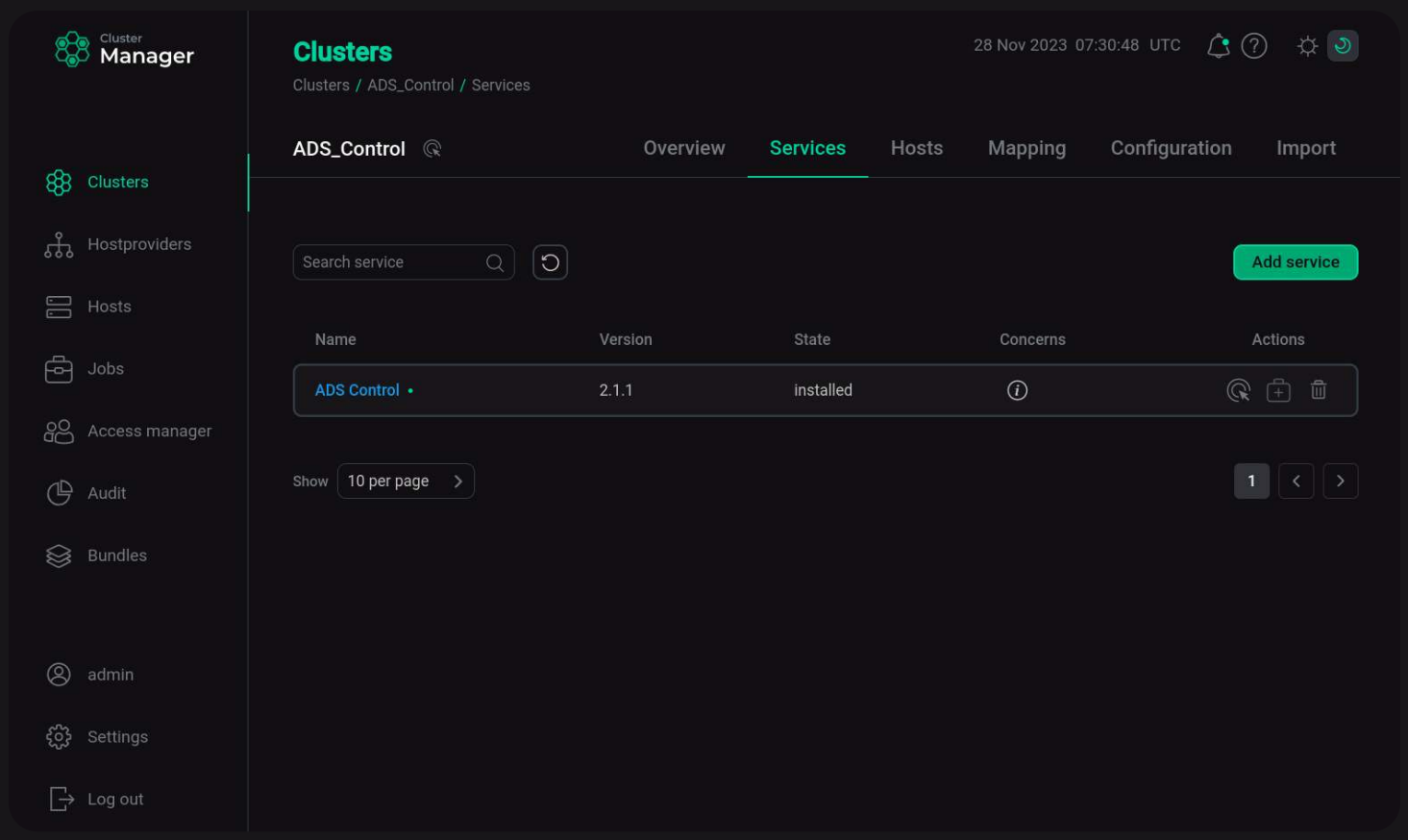
In the dialog box that opens, click ***Run*** to confirm the update. Once the upgrade process is complete, ADCM will change the cluster status to `installed`.

Service actions

Olga Semme

To Table of Contents

ADCM UI provides actions to manage service of ADS Control. These actions are available on the ***Services*** tab. To open this tab, click an ADS Control cluster name on the ***Clusters*** page and switch to the ***Services*** tab.



ADS Control cluster service in the ADCM interface

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

Actions in the ADS Control service:

- ***Check*** — checks the functionality of the service: the availability of the service on the port and the status of the service.
- ***Restart*** — stops and then starts the service.
- ***Start*** — starts the service.
- ***Stop*** — stops the service.
- ***Manage Ranger plugin*** — initiates the activation of the Ranger ADS Control plugin to enable authorization.

ADS Control UI overview

Olga Semeniv

Contents

To Table of Contents

Login window

Main window

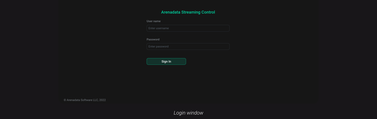
- Authorization section
- Information section
- Search section
- Cluster management section

Cluster management

Below you can find the overview of the features contained in the ADS Control UI.

Login window

This is the first window the user sees after connecting to ADS Control. In order to access to ADS Control, you are prompted to enter your credentials (login and password) in the corresponding fields. After you entered your credentials, click **Sign in**.

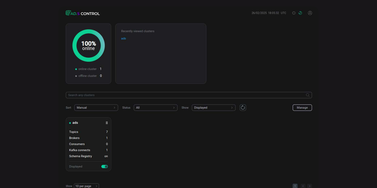


Login window

Main window

This is the window that the user sees after logging into ADS Control. The main window contains four sections:

- **Authorization section** is located at the top of the screen.
- **Information section** is located in the upper half of the screen.
- **Search section** is located in the middle of the screen.
- **The cluster management section** is located in the lower half of the screen.



Main window

All subsections and their contents are described in detail below.

Authorization section

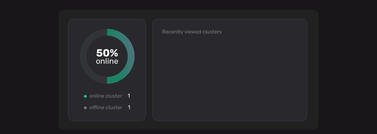
The authorization section appears after clicking on the icon located in the upper right corner. The authorization section contains:

- username under which the session was logged in;
- ***Log out*** button to exit ADS Control or change user.

Authorization section

Information section

The information section contains data on the percentage of the number of clusters in the online and offline state. The data is presented in graphical form. Also in this section is available a list of the latest clusters with which work was carried out.



Information section

Search section

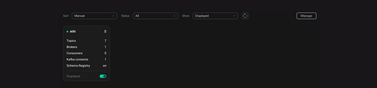
In this section, you can search for clusters by the entered designation. You can also set filters to search – cluster states.

Search section

As a result of the search, clusters matching the search conditions will appear in the cluster management section.

Cluster management section

This section selects a cluster for management.

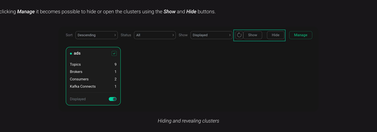


Cluster management section

In the cluster management section, it is possible to set filters for sampling:

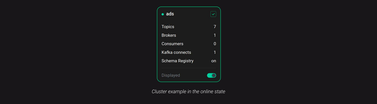
- ***Sort*** – defines the sorting method:
  - ***Ascending*** – sorting by name in ascending alphabetical order;
  - ***Descending*** – sorting by name in descending alphabetical order;
  - ***Manual*** – manual sorting.
- ***Status*** – defines the cluster status for the sample:
  - ***Online*** – clusters in the online state;
  - ***Offline*** – clusters in the offline state;
  - ***All*** – all clusters.
- ***Show*** – indicates which clusters with which visibility should be included in the selection:
  - ***Hidden*** – hidden clusters;
  - ***Displayed*** – visible clusters
  - ***All*** – all clusters.

After clicking ***Manage*** it becomes possible to hide or open the clusters using the ***Show*** and ***Hide*** buttons.

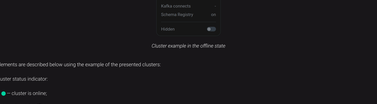


Hiding and revealing clusters

At the bottom of the cluster management section there are graphical displays of the clusters in use.



Cluster example in the online state



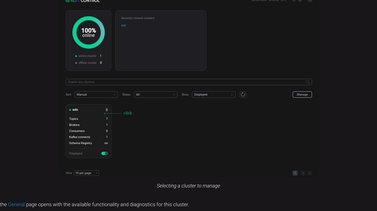
Cluster example in the offline state

The elements are described below using the example of the presented clusters:

- Cluster status indicator:
  - ● – cluster is online;
  - ● – cluster is offline.
- **ads** – cluster name.
- Flag that enables the ability to control the visibility of the cluster using the ***Show*** and ***Hide*** buttons:
  - ☑ – the cluster is available for visibility control;
  - ☐ – the cluster is not available for visibility control.
    Displayed only when ***Manage*** is enabled.
- ⠿ – icon, dragging which it is possible to move the cluster image in the cluster management section. Available only when ***Manage*** is disabled.
- Cluster data showing cluster elements:
  - number of topics;
  - number of brokers;
  - number of consumers;
  - number of Kafka Connect instances;
  - a valid instance of the Schema Registry service.
- switch to change cluster visibility:
  - ⬬ – cluster is displayed ;
  - ⬭ – cluster hidden.

Cluster management

To manage a cluster, in the cluster management section, click on the graphical display of the cluster.



Selecting a cluster to manage

Next, the *General* page opens with the available functionality and diagnostics for this cluster.

After opening the ***General*** window, you can switch to the following tabs for diagnosing and managing the cluster:

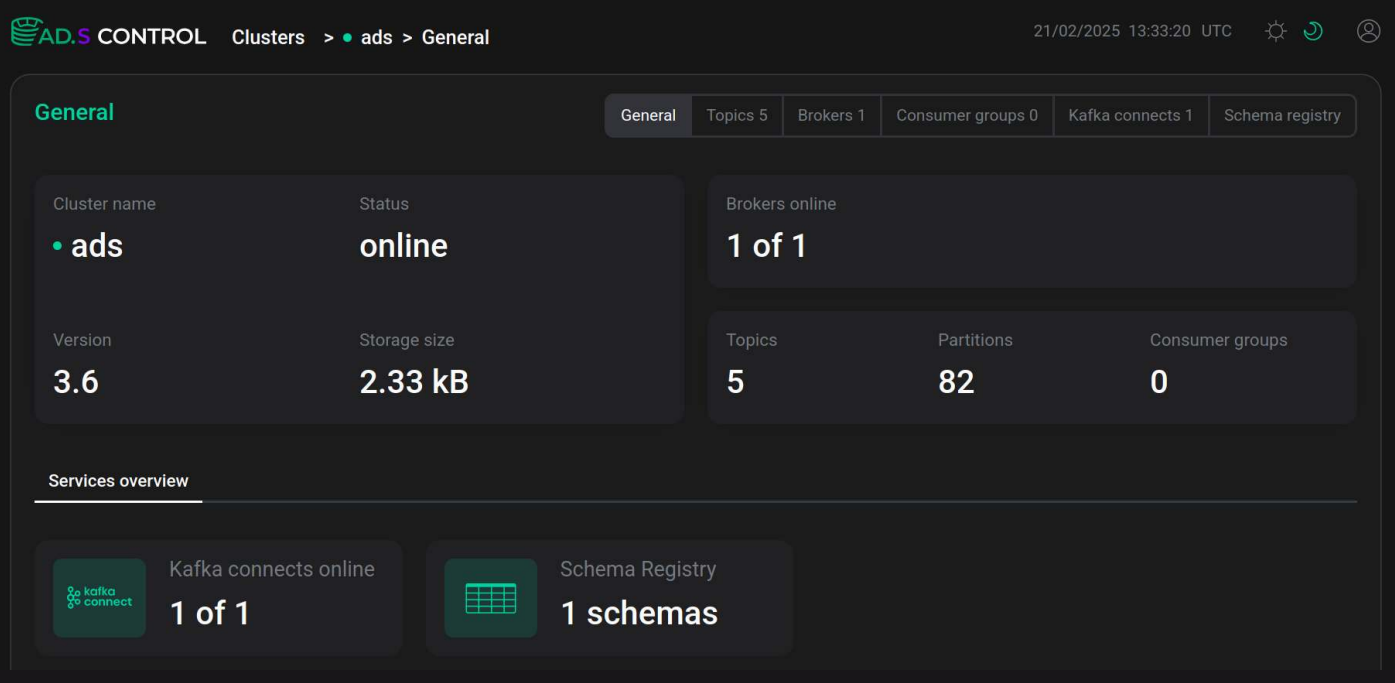
- Topics
- Brokers
- Kafka Connect
- Schema registry

General page in ADS Control

Olga Semme

To Table of Contents

The ***Clusters*** → ***<cluster name>*** → ***General*** page in the ADS Control web interface is designed to display general information about the cluster and becomes available after selecting a cluster in the cluster management section.



General page

On the ***General*** page, you can switch to the following tabs for diagnostics and cluster management:

- Topics
- Brokers
- Consumer groups
- Kafka Connects
- Schema registry

Also the ***General*** page contains the following sections:

- Section with the main cluster parameters:
  - ***Cluster name*** — the name specified when creating the cluster.
  - ***Status*** — the current state of the cluster (`online`, `offline`, or `degraded`).
  - ***Version*** — the version of the product bundle that was used to install the cluster.
  - ***Storage size*** — the total size of data stored in all brokers of the cluster.
- ***Brokers online*** section. The section shows the number of Kafka brokers that are online at the moment. After clicking within the section, you go to the Brokers page.
- Section containing information about the topics that currently exist in the cluster:
  - ***Topics*** — the total number of topics in the cluster, including internal topics. After clicking within the record about the number of topics, you go to the Topics page.
  - ***Partitions*** — the total number of partitions, including partitions of internal topics, excluding replicas.
  - ***Consumer groups*** — the total number of connected consumer groups. After clicking within the record about the number of consumers, you go to the Consumer groups page.
- ***Services overview*** section — displays connected Kafka Connect services if this service is installed in the cluster. By clicking on the service card inside the section, you can go to the page that manages this service in ADS Control:
  - Kafka Connect
  - Schema-Registry

Contents

Topics page overview
Topics tabs
Topic overview
Messages tab
Produce a message
Partitions tab
Configuration tab
Consumer groups tab
Schemas tab

To Table of Contents

Topics page overview

The **Topics** page, under **Topics** > **topic name**, lists all the ADS Control topics and allows you to search for information about topics and manage topics in the ADS cluster. The **Topics** page has two tabs: **Topics** and **Topic groups**. Topic management and monitoring is grouped by the tabs on the Topics page.

The top of the page contains:

- Search: Search for information about topics.
- **Create topic** button: Create a new topic in the cluster, including internal topics.

The table contains the following columns:

Field	Description
Name	Name of the topic. By clicking the name, you can go to the topic overview page.
Messages	Number of messages in the topic.
Internal	Indicates whether the topic is internal.
Size	Size of data contained in the topic.
Retention.ms	Maximum time in milliseconds for the retention policy.
Retention.bytes	Maximum size that the topic can grow to before deleting old segments to retain storage space.
Partitions	Number of partitions for the topic.
Replicas	Number of replicas for the topic. A **CORRECTIVE** action will be shown if any replica does not sync with the partition leader.
Topic storage	Indicates whether the topic is compacted. Compaction is a process that scans the tail of closing closed segments in the log and removes any records that have duplicate keys.
Actions	Delete topic: Delete the topic.

The bottom of the page contains:

- Rows per page: Limit the number of rows shown on the page.
- Navigation: Navigate between pages.

Create a topic

After clicking **Create topic** from the main topics page opens:

At the top of the page there are:

- **Name**: Name of the topic.
- **Partitions**: Number of partitions for the topic.
- **Replication factor**: Number of replicas for the topic.

At the bottom of the page there are:

- **Cancel**: Cancel operation and return to the topics page.
- **Create**: Create the topic.

Topic overview

After clicking the topic name in the topic table, the topic overview page opens.

At the top of the page there is a summary of the topic as well as topic management icons in accordance with the table given above.

The tabs are as described below:

- Messages
- Partitions
- Configuration
- Consumer groups
- Schemas

Messages tab

The **Topics** > **cluster name** > **Topics** > **topic name** > **Messages** tab opens by default when you go to the topic overview page. The tab provides information about the messages contained in the topic and allows you to produce messages to the topic.

Reading messages is divided into two tabs – **Batch** and **Streaming**.

Batch

The **Batch** tab contains:

- Read from dropdown:
  - Read the specified number of messages from the beginning.
  - Read the last number of messages from the end.
- **Start offset**: Text field offset of the partition from which messages will be read. Available values: **Earliest**, **Latest**, **Number**. When you select **Number**, you can enter the offset number where the reading starts.
- **Max messages read**: The maximum number of messages that will be read.
- **Apply** button: Update the offset based on the parameters entered.

- **Filters message** button: After you click it, a window opens for setting messages in a table type. The window structure is given in point.
- Table with information about topic messages. The table columns are described below.

Field	Description
Partition	Partition number
Offset	Offset in the partition under which the message is written
Timestamp	Timestamp
Key	Message key
Value	Message value

The **Streaming** tab contains:

- Read from dropdown:
  - Read the specified number of messages from the beginning.
  - Read the last number of messages from the end.
- **Start offset**: Text field offset of the partition from which messages will be read. Available values: **Earliest**, **Latest**, **Number**. When you select **Number**, you can enter the offset number where the reading starts.
- **Max messages read**: The maximum number of messages that will be read.
- **Apply** button: Update the offset based on the parameters entered.

Values in the fields for filtering messages will automatically apply after the streaming is stopped.

- Buttons for managing message streaming:
  - – start reading messages. When you click this button, the message streaming mode changes to .
  - – end reading messages. When you click this button, the message streaming mode changes to .

Values in the fields for filtering messages will automatically apply after the streaming is stopped.

- Table with information about sent partitions. The table columns are described below.

Field	Description
Partition ID	Partition number
Lag	Amount offset of messages in the topic partition
Offset	Amount offset of messages in the topic partition
Messages	Number of messages in the partition
Actions	Actions with the partition: you can reset the partition to its original state and delete it. After clicking on the button, the offset for you can reset to the beginning or to the end.

Configuration tab

The **Topics** > **cluster name** > **Topics** > **topic name** > **Configuration** tab opens from the topic overview page. The tab allows you to view and edit parameters that can be changed after creating a topic.

On the **Configuration** tab there are:

- Button to switch to viewing parameters by groups.
- – Expand / collapse all parameter groups.
- Table with information about parameters. The table columns are described below.
- Edit button on the row for editing topic parameters. After editing the parameter, click **Save** to save the changes or **Cancel** to cancel the changes.

To view parameters divided into groups, click the button. Below is a brief description of the main parameters that are combined in the **Configuration** tab.

Replication

Parameter	Description
follower.replication.throttled.replicas	List of replicas for which log replication should be regulated on the side of the user.
leader.replication.throttled.replicas	List of replicas for which log replication should be regulated on the leader side.
min.insync.replicas	Minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.
unclean.leader.election.enable	Indicates whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though they may not be in sync.

Storage

Parameter	Description
cleanup.policy	Log cleanup policy for this topic. Valid values are delete and compact.
compression.type	Compression type for topic.
delete.retention.ms	Length of time to retain tombstone markers for log compacted topics. This setting also gives a bound on the time in which a consumer must complete a read if they begin from offset 0 to ensure they get a valid snapshot of the final stage (otherwise tombstone messages may be collected before the consumer completes their scan).
file.delete.delay.ms	Time to wait before deleting a file from the filesystem.
max.compaction.lag.ms	Maximum time a message will remain ineligible for compaction in the log. Only applicable for logs that are being compacted.
max.message.bytes	Maximum record batch size allowed by Kafka (after compression if compression is enabled). If this value is increased and there are consumers older than 0.10.2, the consumers' fetch size must also be increased so that they can fetch record batches this large. In the latest message format version, records are always grouped into batches for efficiency. In previous message format versions, uncompressed records are not grouped into batches and this limit only applies to a single record in that case. This can be set per topic with the topic level max.message.bytes config.
message.format.version	Specify the message format version the broker will use to append messages to the logs. The value should be a valid ApiVersion. Some examples are: 0.8.2, 0.9.0.0, 0.10.0, check ApiVersion for more details. By setting a particular message format version, the user is certifying that all the existing messages on disk are smaller or no older than that version. The message format version is used by producers to determine which version of the client API is supported by the broker for the messages they send. Setting this value incorrectly will cause consumers with older versions to break as they will receive messages with a format that they don't understand.
message.timestamp.difference.max.ms	Maximum difference allowed between the timestamp when the broker receives the message and the timestamp specified in the message. If message.timestamp.type=CreateTime, a message will be rejected if the difference in timestamp exceeds this threshold. This configuration is ignored if message.timestamp.type=LogAppendTime. The maximum timestamp difference allowed should be no greater than log.retention.ms to avoid unnecessarily frequent log rolling.
message.timestamp.type	Define whether the timestamp in the message is message create time or log append time. The value should be either CreateTime or LogAppendTime.
min.cleanable.dirty.ratio	This configuration controls how frequently the log compactor will attempt to clean the log (assuming log compaction is enabled). By default we will avoid cleaning a log where more than 50% of the log has been compacted. This ratio bounds the maximum space wasted in the log by duplicates (at most 50% of the log could be duplicates). A higher ratio will mean fewer, more efficient cleanings, but will mean more wasted space in the log. If the max.compaction.lag.ms or the min.compaction.lag.ms configurations are also specified, then the log compactor considers the log to be eligible for compaction as soon as either: (i) the dirty ratio threshold has been met and the log has had dirty (uncompacted) records for at least the min.compaction.lag.ms duration, or (ii) the log has had dirty (uncompacted) records for at most the max.compaction.lag.ms period.
retention.bytes	Size of the data that topic partitions can store.
segment.bytes	Segment size for the topic.
segment.ms	Period of time after which Kafka forces to use the log for the possibility of starting a compacting/deleting, even if the segment file is not full.

Storage Performance

Parameter	Description
flush.messages	Number of messages of data which has been stored in the topic before synchronization of application data with data on the hard drive.
flush.ms	Time interval at which to force fsync of logged data.
index.interval.bytes	Frequency of adding an index entry to the Kafka offset index.
message.downconversion.enable	Enable down-conversion of message formats to support consumption requests.
min.insync.replicas	Minimum number of replicas that must acknowledge a write for the write to be considered successful. If this minimum cannot be met, then the producer will raise an exception (either NotEnoughReplicas or NotEnoughReplicasAfterAppend). When used together, min.insync.replicas and acks allow you to enforce greater durability guarantees. A typical scenario would be to create a topic with a replication factor of 3, set min.insync.replicas to 2, and produce with acks of "all". This will ensure that the producer raises an exception if a majority of replicas do not receive a write.
preallocate	Indicates whether to preallocate a file on disk for a new segment in the log.

Others

Parameter	Description
local.retention.bytes	Maximum local log segment size.
local.retention.ms	Maximum time for local log segments to be stored before deleting them.
message.timestamp.after.max.ms	Acceptable difference between the message timestamp and the broker timestamp. The message timestamp can be later than or equal to the broker timestamp, with the maximum acceptable difference determined by this value. Only applicable if message.timestamp.type=CreateTime.
message.timestamp.before.max.ms	Acceptable difference between the broker timestamp and the message timestamp. The message timestamp can be earlier than or equal to the broker timestamp, with the maximum acceptable difference determined by this value. Only applicable if message.timestamp.type=CreateTime.
unclean.leader.election.enable	Specifies whether to enable replicas not in the ISR set to be elected as leader as a last resort, even though they may not be in sync. Note: Default: Server uses the value specified in the broker's default.replication.factor.
segment.index.bytes	Segment index size for the topic.

Disable the "Read only" option for the topic.

If the cleanup.policy parameter is set to compact, then the segment.bytes parameter is set to the segment.ms parameter.

Consumer groups tab

The **Topics** > **cluster name** > **Topics** > **topic name** > **Consumer Groups** tab opens from the topic overview page. The tab displays consumer groups that are connected to the topic.

The **Consumer groups** tab contains a table with information about connected consumer groups. The table columns are described below.

Field	Description
Consumer group ID	Consumer group name. By clicking on the name, you can go to the consumer group overview page.
Reading rates	Reading status
Lag	Delay in consumer processing

Work with consumer groups is carried out on the Topic groups tab.

Schemas tab

The **Topics** > **cluster name** > **Topics** > **topic name** > **Schemas** tab displays the schema for the topic message key and value.

The **Schemas** tab contains a table with information about the latest versions of schemas for the topic. The description of the table columns is given below.

Field	Description
Type	Schema type (key or message)
Name	Schema name (the is schema)
Schema ID	Schema ID
Actions	Latest schema version

Work with schemas (creating and editing) is carried out on the Schemas tab.

Work with brokers in ADS Control

Olga Semme

Contents

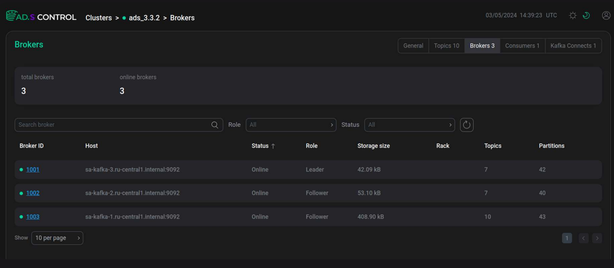
To Table of Contents

Brokers page overview

Broker overview

Brokers page overview

The ***Clusters*** → ***<cluster name>*** → ***Brokers*** page in the ADS Control web interface is designed to display information about Kafka brokers in the ADS cluster. The ***Brokers*** page becomes available after selecting a cluster in the cluster management section and going to the desired tab on the General page.



Brokers page

The top of the page contains:

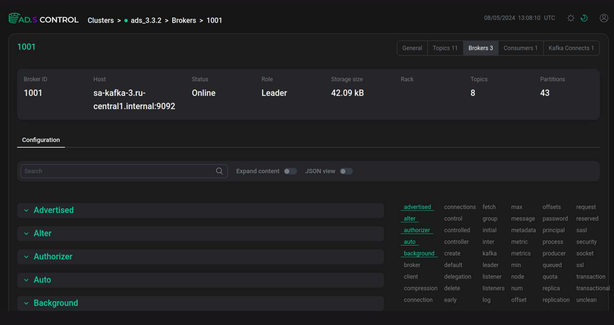
- Section of general information about brokers:
  - ***total brokers*** field — the total number of brokers in the cluster;
  - ***online brokers*** field — the total number of brokers in online status.
- Search and filtering section:
  - field for searching a broker by ID;
  - ***Role*** field for filtering brokers by broker role in the cluster: ***Leader*** or ***Follower***;
  - ***Status*** field for filtering brokers by broker status in the cluster: ***Online*** or ***Offline***;
  - button to reset the search and filtering results.

The ***Brokers*** page contains a table with information about each broker. The table columns are described below.

Field	Description
Broker ID	Broker ID. By clicking the identifier, you can go to the broker overview page
Host	Broker host name
Status	Broker status
Role	Role of the broker in the cluster
Storage size	Data size on the broker
Rack	Rack accessory (if configured)
Topics	Total number of topics storing their partitions on the broker
Partitions	Total number of stored partitions

Broker overview

After clicking the broker ID in the brokers table, the broker overview ***Clusters*** → ***<cluster name>*** → ***Brokers*** → ***<broker ID>*** page opens.



Broker overview page

At the top of the page there is a section displaying the main parameters of the broker in accordance with the table given above.

On the broker overview page there is a ***Configuration*** tab containing drop-down lists with broker parameters.

Kafka broker parameters

- Broker parameters cannot be changed in ADS Control and are read-only.
- For detailed information about all Kafka configuration parameters available in the ADCM interface, see ADS configuration parameters (under ***Kafka***).
- The broker parameters are configured on the configuration page of the ***Kafka*** service in the ADCM interface. Broker parameters that are not available in the ADCM interface can be specified in a special field. To do this, set the ***Show advanced*** switch to active and expand the ***server.properties*** node in the configuration settings tree. Using the ***Add key,value*** field, select ***Add property*** and enter the name of the parameter and its value.
- For detailed information about all broker parameters, you can refer to Apache Kafka documentation.

Work with consumer groups in ADS Control

Olga Semme

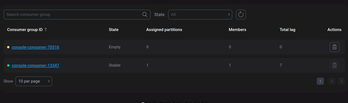
Contents

To Table of Contents

Consumer groups page overview
Consumer group overview
Change the message offset

Consumer groups page overview

The ***Clusters → <cluster name> → Consumer groups*** page in the ADS Control web interface is designed to display information about connected consumer groups in the ADS cluster. The ***Consumer groups*** page becomes available after selecting a cluster in the cluster management section and going to the desired tab on the General page.



Consumer groups page

The top of the page contains:

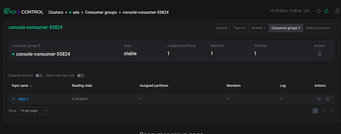
- Section with general information about consumers:
  - ***Total groups*** field – the total number of consumer groups in the cluster;
  - ***Stable groups*** field – the number of consumer groups that include an active consumer;
  - ***Empty groups*** field – the number of consumer groups that do not contain active consumers.
- Search and filtering section:
  - field for searching a consumer group by name;
  - ***State*** field for filtering connectors by status:
    - `Stable` – there is an active participant in the consumer group;
    - `Empty` – there are no active participants in the consumer group;
    - `Dead` – the consumer group has no members and its metadata has been deleted;
    - `Preparing rebalance` – reading messages has been stopped by participants so that partitions can be reassigned;
    - `Completing rebalance` – reassigning sections to group members;
  - button to reset the search results.

The ***Consumer groups*** page contains a table with information about each consumer group. The table columns are described below.

Field	Description
Consumer group ID	Consumer group name. By clicking the name, you can go to the consumer group overview page
State	Consumer group status
Assigned partitions	Number of partitions assigned to the consumer group
Members	Number of group members
Total lag	Total lag of all consumers is the delay in the time it takes for a message to travel from the producer (which generates the messages) to the consumer (which receives them)
Actions	Column contains an icon to delete a consumer group. A group can only be deleted if there are no active members in it. Action requires confirmation

Consumer group overview

After clicking on the group name in the consumer groups table, the consumer group page ***Clusters → <cluster name> → Consumer groups → <consumer group name>*** opens.



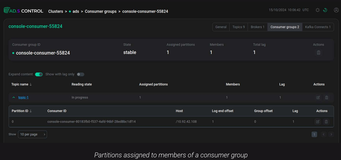
Consumer group page

At the top of the page, there is a section displaying the main parameters of the consumer group and an icon for deleting the group in accordance with the table given above.

On the consumer group page, there is a table with information about the topics to which members of the consumer group are connected. The table columns are described below.

Field	Description
Topic name	Topic name. By clicking the name, you can go to the topic overview page
Reading state	Reading status
Assigned partitions	Number of assigned partitions
Members	Number of group members
Lag	Lag in consumer processing
Actions	Column contains icons for controlling reading from the topic: – when you click the icon, the Change partition offsets window opens. – when you click the icon, the topic is removed from the connections of the consumer group. This action removes the offset for the topic. With a new connection with the same group of consumers, reading from this topic will begin at offset zero. The action requires confirmation. Actions are only available if there are no active group members

Enabling the ***Expand content*** switch, like clicking on the line with the desired topic, opens a table with information about the partitions that are assigned to group members. The ***Show with lag only*** switch leaves in the table only those topics that have a processing lag by the consumer.



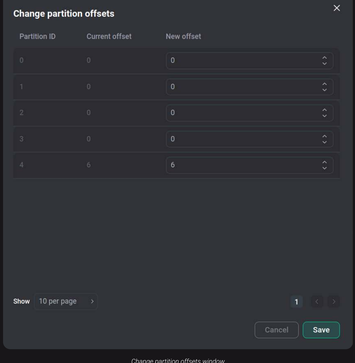
Partitions assigned to members of a consumer group

The table columns are described below.

Field	Description
Partition ID	Number of the partition assigned to the group member
Consumer ID	Consumer Group Member ID
Host	IP host to connect
Log end offset	Last offset in the log
Group offset	Group offset
Lag	Lag in consumer processing
Actions	Column contains icons for controlling reading from the partition: – when you click the icon, the Change partition offsets window opens for a separate partition. – when you click the icon, the partition is removed from the connections of the consumer group. This action removes the offset for a specific partition. With a new connection to the same consumer, reading from this topic will begin at offset zero. The action requires confirmation. Actions are only available if there is no active connected consumer for the partition

Change the message offset

You can change the messages offsets in the ***Change partition offsets*** window, which appears after clicking the icon in actions for a topic connected to a consumer group or separately for each connected partition on the consumer group overview page.



Change partition offsets window

To change the offset, change the ***New offset*** field for the partition with the required ***Partition ID*** and save the new offset by clicking ***Save***.

Work with Kafka Connect in ADS Control

A Tips Tricks

ADS CONTROL

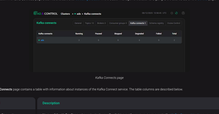
Contents

To Table of Contents

Kafka Connects page overview
Kafka Connect instance overview
Create a connector
Connector overview
- Overview tab
- Configuration tab
- Offset management tab

Kafka Connects page overview

The **Clusters** → **<cluster name>** → **Kafka Connects** page in the ADS Control web interface is designed to display information about available instances of the [Kafka Connect](#) service in the ADS cluster. The **Kafka Connects** page includes the [overview panel](#) in the cluster management section and opening by the desired tab in the [menu](#) page.



Kafka Connects page

The **Kafka Connects** page contains a table with information about instances of the Kafka Connect service. The table columns are described below:

Field	Description
Kafka Connects	Name of the Kafka Connect service instance. By clicking on the name, you can go to the Kafka Connect instance overview page
Running	Number of running connectors
Paused	Number of paused connectors
Stopped	Number of stopped connectors
Degraded	Number of connectors in `degraded` status
Failed	Number of connectors with error
Total	Total number of connectors

Kafka Connect instance overview

After clicking on the name, the Kafka Connect instance overview page (**Clusters** → **<cluster name>** → **Kafka Connects** → **<cluster name> connector** opens.

Kafka Connect instance page

The **Kafka Connects** service overview page contains:

- Search and filtering section:
  - field for searching a connector by name;
  - **Type** field for filtering connectors by type: `Source` or `Sink`;
  - **Status** field for filtering connectors by status:
    - `Running`
    - `Stopped`
    - `Paused`
    - `Unassigned`
    - `Failed`
    - `Degraded`
  - button to reset the search and filtering results;
  - **Create Connector** button (for [creating](#) a new connector).
- Table with information about created Kafka Connect connectors. The table columns are described below:

Field	Description
Name	Connector name. By clicking on the name, you can go to the connector instance overview page
Type	Connector type
Plugin	Name of the connector plugin
Topics	Topics related to the operation of the connector
Broker	Broker on which the connector is running
Tasks	Number of connector tasks
Actions	Column contains icons for managing the connector: - restart the connector. - pause the connector. When the connector is paused, its tasks continue to exist in the cluster configuration. - connector shutdown. - run the connector. - delete the connector. All actions require confirmation

Stopped status

Starting from [ADS 2.0.0](#), Control Center has a new state for connectors — `Stopped`, which is the Kafka initial state after creation. The initial state of a connector can be changed using the `initial.state` parameter in the configuration file when creating the connector.

When a stopped connector is resumed, `Running` or paused, `Paused`, the connector starts on the assigned worker node and creates a set of tasks according to the configuration.

After switching an existing connector to the `Stopped` state:

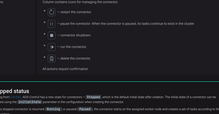
- The connector stops working.
- The connector appears in ADS Control with the `Stopped` state. This will happen even if the connector was in the `Failed` state before the stop request or if it encountered an error stopping (see [KIP-875](#)).
- All connector tasks are completely terminated. If running in distributed mode, a set of empty tasks is published to the connector configuration topic.
- When starting a stopped connector, a previous set of tasks in the configuration topic will receive a tombstone, and tasks across the cluster will not be assigned to the connector. As a result of this rebalance, any information about the connector provided through the REST API will show it with an empty set of tasks.
- All resources allocated to the connector or its tasks (sockets and buffers) will be released.

After changing the status to `Stopped`, it is possible to modify or reset offsets for the connector on the Offset management tab.

Create a connector

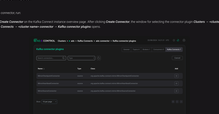
To create a connector, run:

- Click **Create Connector** in Kafka Connects overview page. After clicking **Create Connector** the window for selecting the connector plugin (**Clusters** → **<cluster name>** → **Kafka Connects** → **<cluster name> connector** → **Kafka connector plugin**) opens.

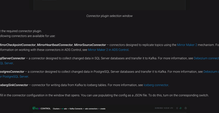


Connector plugin selection window

2. Select the required connector plugin.
   The following connectors are available for use:
   - **MirrorSourceConnector**, **MirrorCheckpointConnector**, **MirrorHeartbeatConnector** – connectors designed to replicate topics using the [MirrorMaker 2](#) mechanism. For more information on working with these connectors in ADS Control, see [Mirror Maker 2 in ADS Control](#).
   - **KpiServerConnector** – a connector designed to collect statistics in the KSL Server database and transfer it to Kafka. For more information, see [database connector for KSL Server](#).
   - **PostgreSqlConnector** – a connector designed to collect changes data in PostgreSQL Server databases and transfer it to Kafka. For more information, see [Debezium connector for PostgreSQL Server](#).
   - **IndexConnector** – a connector for writing data from Kafka to Solany indexes. For more information, see [Solany connector](#).
3. Next, fill in the connector configuration in the window that opens. You can use populating the config as a JSON file. To do this, turn on the corresponding switch.



Connector configuration

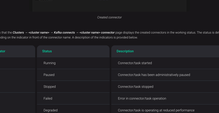


Connector JSON configuration file

4. After filling, click **Save** and get a message about the successful creation of the connector.

Message about the successful creation of the connector

5. Make sure that the page of the created connector is opened with the display of tasks.



Created connector

6. Check that the **Clusters** → **<cluster name>** → **Kafka connects** → **<cluster name> connector** page displays the created connectors in the working status. The status is determined depending on the indicator in front of the connector name. A description of the indicators is provided below.

Indicator	Status	Description
	Running	Connector/task started
	Paused	Connector/task has been administratively paused
	Stopped	Connector/task stopped
	Failed	Error in connector/task operation
	Degraded	Connector/task is operating at reduced performance
	Unassigned	Connector/task has not yet been assigned to a worker

Connector overview

After clicking on the connector name in the table with information about the created connectors, the connector instance overview page opens.
At the top of the page, there is a section displaying the main parameters of the connector, as well as connector management icons according to the table provided [above](#).

Main connector parameters

Tabs are displayed at the bottom:

- [Overview](#)
- [Configuration](#)
- [Offset management](#)

Overview tab

The **Clusters** → **<cluster name>** → **Kafka connects** → **<cluster name> connector** → **<connector name>** → **Overview** tab opens by default when navigating to the connector overview page. The tab provides information about the connector's [tasks](#) and their status.

Overview tab

On the **Overview** tab, you will find:

- Search and filter section:
  - field to search for a task by ID;
  - **Status** field to filter tasks by status:
    - `Running`
    - `Failed`
    - `Paused`
    - `Stopped`
    - `Unassigned`
    - `Degraded`
  - button to reset search results.
- Table with information about each task. The description of the table columns is provided below:

Field	Description
Task ID	Task ID
Status	Task status. When the status is `Failed`, clicking on the icon opens a description of the error that occurred
Actions	Column contains the icon for restarting the task

Configuration tab

The **Clusters** → **<cluster name>** → **Kafka connects** → **<cluster name> connector** → **<connector name>** → **Configuration** tab opens from the connector overview page. The tab allows you to view and edit the connector's configs.

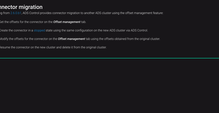
Configuration tab

The **Configuration** tab contains:

- Display switch key parameters by name.
- **JSON** / **Parameter** switch, which means hidden content.
- **JSON** / **JSON view** switch, which opens the content in JSON format.
- **Edit** button to enter connector parameter editing mode. After editing the parameters, click **Save** to save changes or **Cancel** to discard them.
- Connector parameters, divided into dropdown lists. The set of connector configuration parameters depends on the connector plugin and generally includes the main parameters of the [Kafka Connect](#) and [Kafka brokers](#), as well as parameters defined by the plugin vendor. Some parameters are described in articles dedicated to [plugin connectors](#) in ADS Control.

Offset management tab

The **Clusters** → **<cluster name>** → **Kafka connects** → **<cluster name> connector** → **<connector name>** → **Offset management** tab is opened from the connector instance overview page. The tab allows you to view and modify offsets for the connector.



Offset management tab

The **Offset management** tab contains:

- **Edit** button to enter the connector offset editing mode. After editing offsets, click **Save** to save changes or **Cancel** to discard them. The button is available only if the connector is in the [Paused](#) status.
- Current connector offsets as a JSON file, which is the response to the offsets request in [KIP-875 Connect offsets management, update or reset](#).

Below is an example of a JSON file for different instances of the connector.

Offsets for IotBenchConnector

```
{
  "offsets": [
    {
      "partition": {
        "kafka_partition": 0,
        "kafka_topic": "topic-test"
      },
      "offset": {
        "kafka_offset": 1
      }
    }
  ]
}
```

Offsets for PostgreSqlConnector

```
{
  "offsets": [
    {
      "partition": {
        "server": "postgres"
      },
      "offset": {
        "last_snapshot_record": true,
        "lsn": 3179624,
        "txId": 1892,
        "ts_usec": 1654555619291,
        "snapshot": "INITIAL",
        "snapshot_completed": true
      }
    }
  ]
}
```

Connector migration

Starting from [ADS 2.0.0](#), ADS Control provides connector migration to another ADS cluster using the offset management feature.

1. Set the offsets for the connector on the **Offset management** tab.
2. Create the connector on a [target](#) stable using the same configuration on the new ADS cluster via ADS Control.
3. Modify the offsets for the connector on the **Offset management** tab using the offsets obtained from the original cluster.
4. Resume the connector on the new cluster and delete it from the original cluster.

Work with Schema-Registry in ADS Control

Olga Sarmina

Contents

To Table of Contents

Schema registry page overview

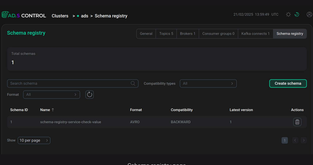
Create a schema

Schema overview

- Schema tab
- Version history tab

Schema registry page overview

The ***Clusters*** → ***<cluster name>*** → ***Schema registry*** page in the ADS Control web interface is designed to display information about schemas and manage schemas in ADS clusters with the Schema-Registry service installed. The ***Schema registry*** page becomes available after selecting a cluster in the cluster management section and going to the desired tab on the General page.



Schema registry page

On the ***Schema registry*** page there are:

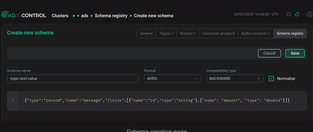
- ***Total schemas*** field — displays the number of schemas created for the cluster.
- Search and filtering section:
  - field for searching for a schema by name;
  - ***Compatibility types*** field for filtering schemas by compatibility type:
    - `NONE`
    - `FULL`
    - `FORWARD`
    - `BACKWARD`
    - `FORWARD_TRANSITIVE`
    - `BACKWARD_TRANSITIVE`
    - `FULL_TRANSITIVE`
  - ***Format*** field for filtering schemas by format:
    - `AVRO`
    - `PROTOBUF`
    - `JSON`
  - button to reset search and filtering results;
- ***Create schema*** button for creating new schemas.

The ***Schema registry*** page contains a table with information about each schema. The table columns are described below:

Field	Description
Schema ID	Schema unique identifier
Name	Subject name of the schema. For schemas created automatically when recording messages using the producer, it is filled in depending on the selected strategy naming. By clicking on the name, you can go to the schema overview page
Format	Used schema format
Compatibility	Compatibility type set for this schema
Latest version	Latest schema version
Actions	After clicking on the icon, the schema is deleted. Action requires confirmation

Create a schema

After clicking the ***Create schema*** button, the schema creation page opens.



Schema creation page

The page contains a field for entering the contents of the schema and fields with the parameters of the new schema, described in the table given above:

- ***Schema name*** — name of the schema subject.
- ***Format*** — schema format.
- ***Compatibility type*** — schema compatibility type.

When the ***Normalize*** flag is activated, a semantic check for schema equivalence is performed. This activation sets the value `true` to the serializer parameter `normalize.schemas`.

After creation, the schema can be saved using the ***Save*** button. The ***Cancel*** button cancels the creation of the schema.

Schema overview

Clicking on the schema name in the table opens the schema overview page.

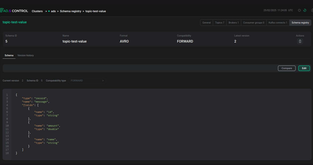
At the top of the page there is a section displaying the main parameters of the schema, described in the table given above, and the icon to delete the schema.

Tabs are displayed at the bottom:

- Schema
- Version history

Schema tab

The ***Clusters*** → ***<cluster name>*** → ***Schema registry*** → ***<schema name>*** → ***Schema*** tab opens by default when you go to the schema overview page. The tab displays the current version of the schema and information about it.

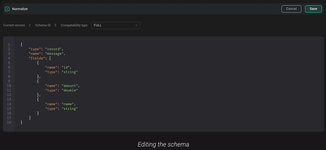


Schema tab

The ***Schema*** tab contains the schema and fields with schema parameters:

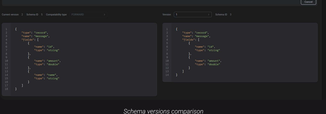
- ***Current version*** — number of the current version.
- ***Schema ID*** — identifier of the current version.
- ***Compatibility type*** — schema compatibility type.

After clicking the ***Edit*** button, the ability to edit the schema opens. In addition to the contents of the schema, the ***Normalize*** and ***Compatability type*** fields can also be changed. Clicking the ***Save*** button saves the changes. Any change to the schema content will result in a new version of the schema being created. The ***Cancel*** button cancels the change to the schema.



Editing the schema

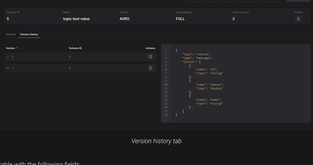
After clicking the ***Compare*** button, it becomes possible to compare the current version of the schema and any previous one. In this case, in the ***Version*** field that appears, the desired version of the schema is selected, and the ***Schema ID*** field displays the identifier of the selected schema version. The ***Cancel*** button closes the ability to compare schemas.



Schema versions comparison

Version history tab

The ***Clusters*** → ***<cluster name>*** → ***Schema registry*** → ***<schema name>*** → ***Version history*** tab opens from the schema overview page. The tab allows you to view all created schema versions.



Version history tab

The ***Version history*** tab contains a table with the following fields:

- ***Version*** — schema version.
- ***Schema ID*** — schema version identifier.

The schema version can be deleted using

The contents of the schema for each version can be viewed after clicking on the row.

Mirror Maker 2 in ADS Control

Olga Semina

Contents

To Table of Contents

Create connectors for Mirror Maker 2

Topic replication check

This article describes setting up topic replication to a remote cluster via the Mirror Maker 2 mechanism using the ADS Control.

The main concepts used in this article:

- **Source cluster** – the cluster from which topics are replicated.
- **Target cluster** – the cluster to which topics are replicated.
- **Source topic** – a topic created on the source cluster and intended to be replicated to the target cluster.
- **Replica topic** – a topic automatically created on the target cluster and having the same content and metadata as the source topic.

NOTE

- Replication is shown based on the Active/Standby architecture when the target cluster is in passive mode (has no consumers and producers connected to it) and receives only replicated data.
- Before configuring Mirror Maker 2, the Kafka Connect service must be installed and configured on the source and destination clusters. Kafka Connect is available for installation starting from ADS version 1.7.1.b1.

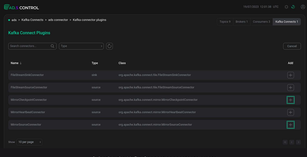
Create connectors for Mirror Maker 2

To perform a simple replication of a topic to a remote cluster, use the following connectors created on the basis of Kafka Connect:

- MirrorSourceConnector
- MirrorCheckpointConnector

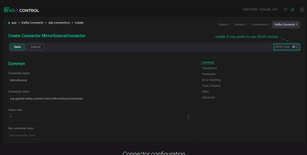
To create connectors using the ADS Control, you need to:

1. Go to the Kafka Connects page in the ADS Control web interface. The **Kafka Connects** page becomes available after selecting a cluster in the cluster management section and going to the desired tab on the General page.
2. Select the desired cluster and go to the Kafka Connect instance overview page.
3. Click **Create Connector** on the Kafka Connect instance overview page. After clicking **Create Connector**, the window for selecting the connector plugin (**Clusters → <cluster name> → Kafka Connects → <cluster name> connector → Kafka connector plugins** opens.
4. Select the desired connector to create.

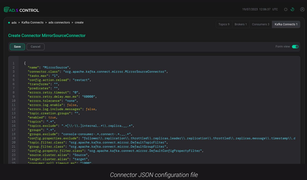


Selecting the Kafka Connect connector to create

5. Fill in the connector configuration parameters. If necessary, refer to the Kafka Connect service configuration information in ADS configuration parameters. You can use filling in the configuration in the form of a JSON file. To do this, enable the corresponding switch.



Connector configuration



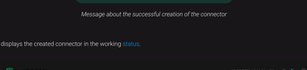
Connector JSON configuration file

Example JSON file content for simple connector configuration for Active/Standby architecture

```
{
    "connector.class": "org.apache.kafka.connect.mirror.MirrorSourceConnector",
    "source.cluster.alias": "source",
    "target.cluster.alias": "dest",
    "name": "<connector_name>",
    "target.cluster.bootstrap.servers": "<dest_cluster_bootstrap>",
    "tasks.max": "4",
    "topics": "<test_topic_name>",
    "source.cluster.bootstrap.servers": "<source_cluster_bootstrap>"
}
```

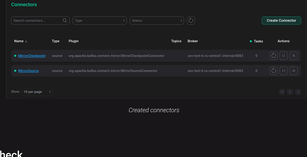
Attribute	Description
connector.class	Name of the class for this connector. Must be a subclass of `org.apache.kafka.connect.connector.Connector`
source.cluster.alias	Name of the source cluster to be used in the replication
target.cluster.alias	Name of the target cluster to be used in the replication
name	Name of the connector to be used in the replication
target.cluster.bootstrap.servers	Bootstrap-servers of the target cluster, used with port 9092
tasks.max	Maximum number of tasks that can be used for a connector
topics	Comma-separated list of source topics to use
source.cluster.bootstrap.servers	Bootstrap-servers of the source cluster, used on port 9092

6. After filling, click ***Save*** and get a message about the successful creation of the connector.



Message about the successful creation of the connector

7. Check that the opened page displays the created connector in the working status.



Created connectors

Topic replication check

After successful completion of the connector configuration, a service topic `mm2-offset-syncs.<target_cluster_alias>.internal` is created on the side of the source cluster, where `<target_cluster_alias>` corresponds to the target cluster designation entered during the connector configuration.

The `mm2-offset-syncs` topic is populated by the MirrorSourceConnector and then used by the MirrorCheckpointConnector to convert consumer group offsets.

In order to check for the presence of the service topic `mm2-offset-syncs`, enter the command on any broker of the source cluster to display the list of topics:

```
$ /usr/lib/kafka/bin/kafka-topics.sh --list --bootstrap-server hostname:9092
```

The list of topics that appears should contain the desired topic:

```
__consumer_offsets
mm-connect-configs
mm-connect-offsets
mm-connect-status
mm2-offset-syncs.dest.internal
```

The creation of a replicated source topic and the writing of messages to it occurs on the side of the source cluster.

To create an source topic, enter the command:

```
$ /usr/lib/kafka/bin/kafka-topics.sh --create --topic <test_topic_name> --bootstrap-server hostname:9092
```

where `<test_topic_name>` – the name of the source topic, as entered when configuring the connector.

To write messages to the original topic, enter the command:

```
$ /usr/lib/kafka/bin/kafka-console-producer.sh --topic <test_topic_name> --bootstrap-server hostname:9092
```

After writing messages to the source topic on the ***Connectors for ads*** page in the ADS Control user interface, the name of the replica topic appears in the line of the MirrorSourceConnector connector in the `<source_cluster_alias>.<test_topic_name>` format, where:

- `<source_cluster_alias>` – the corresponding source cluster alias entered during connector configuration.
- `<test_topic_name>` – the name of the source topic entered during connector configuration.

List of connectors indicating the replica topic

The replica topic also appears in the list of topics on the target cluster. In order to read messages from the replica topic on the target cluster, enter the command:

```
$ /usr/lib/kafka/bin/kafka-console-consumer.sh --topic <source_cluster_alias>.<test_topic_name> --from-beginning --bootstrap-server hostname:9092
```

Read messages in the target cluster completely repeat those written in the source cluster.

Debezium connector for PostgreSQL server

Olga Semmar

Contents

To Table of Contents

Prerequisites

Create a Debezium connector for PostgreSQL server

Use the Debezium connector for PostgreSQL server

snapshot.mode

The article shows an example of running and using the Debezium connector for PostgreSQL server via ADS Control. The connector is designed to record events about row-level changes in PostgreSQL database tables to Kafka topics.

For this example, the PostgreSQL server is Arenadata Postgres (ADPG).

Prerequisites

The following environment was used to create the connector:

- The ADS cluster is installed according to the Online installation guide. The minimum ADS version is 3.6.2.b1.
- Kafka and Kafka Connect services are installed in the ADS cluster.
- To automatically create a Kafka topic, the *auto.create.topics.enable* parameter is enabled in the *server.properties* group when configuring the Kafka service.
- The ADS Control cluster is installed according to the guide Get started with Arenadata Streaming Control and integrated with the ADS cluster being used.
- The ADPG cluster is installed according to the Online installation guide.
- PostgreSQL server IP address (a host with the ADPG service) is `10.92.40.129`. For incoming connections, the default port number is `5432`.
- On the installed PostgreSQL server (ADPG cluster), the following preparatory steps have been completed:
  - The `books_store` database exists in the ADPG cluster.
  - A user with the `user1` name, privilege `SUPERUSER`, and `password1` password are created in the `books_store` database.
  - The `book` and `author` tables were created in the database and several rows of data were added to them.
  - The pg_hba.conf file is configured to provide access to the user and the host on which the Kafka Connect service of the ADS cluster is installed. To do this, an entry about the host number and user is added to the *PG_HBA* field on the configuration page of the ADPG service:

```
host    books_store  user1        10.92.43.204/32     trust
```

  - The *wal_level* parameter value in the *postgresql.conf* file is changed from `replica` to `logical` – this value adds information required to support logical decoding, which is necessary for the connector to work. To do this, an entry about the parameter and the new value has been added to the *postgresql.conf custom section* field on the configuration parameters page of the ADPG service.

Set permissions for the Debezium connector

Configuring PostgreSQL server to run the Debezium connector when using the `pgoutput` plugin requires a database user who has the following privileges:

- `REPLICATION`
- `LOGIN`
- `CREATE` in the database to add publications.
- `SELECT` on tables to copy the original table data.

The user with the `SUPERUSER` role in this example has all the necessary privileges, but it is recommended that you use this role only in test mode. For server security, it is recommended to configure individual privileges for the user as described above.

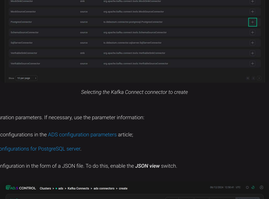
More information about setting up user privileges for the Debezium connector can be found in the Setting up permissions section of the Debezium connector for PostgreSQL server documentation.

Create a Debezium connector for PostgreSQL server

To create a Debezium connector for PostgreSQL server via ADS Control, use the *PostgresConnector* connector plugin.

To create connectors using the ADS Control, you need to:

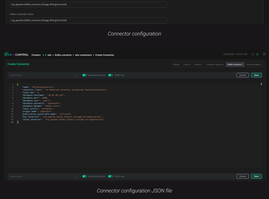
1. Go to the Kafka Connects page in the ADS Control web interface. The *Kafka Connects* page becomes available after selecting a cluster in the cluster management section and going to the desired tab on the General page.
2. Select the desired cluster and go to the Kafka Connect instance overview page.
3. Click *Create Connector* on the Kafka Connect instance overview page. After clicking *Create Connector*, the window for selecting the connector plugin (*Clusters → <cluster name> → Kafka Connects → <cluster name> connector → Kafka connector plugins* opens).
4. Select the desired connector to create.



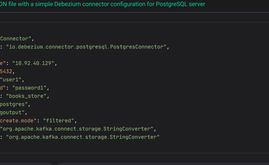
Selecting the Kafka Connect connector to create

5. Fill in the connector configuration parameters. If necessary, use the parameter information:
   - Kafka Connect service configurations in the ADS configuration parameters article;
   - Debezium connector configurations for PostgreSQL server.

   You can use filling in the configuration in the form of a JSON file. To do this, enable the *JSON view* switch.



Connector configuration



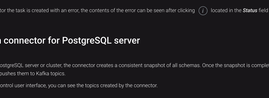
Connector configuration JSON file

Example contents of a JSON file with a simple Debezium connector configuration for PostgreSQL server

```
{
    "name": "PostgresConnector",
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "tasks.max": 1,
    "database.hostname": "10.92.40.129",
    "database.port": 5432,
    "database.user": "user1",
    "database.password": "password1",
    "database.dbname": "books_store",
    "topic.prefix": "postgres",
    "plugin.name": "pgoutput",
    "publication.autocreate.mode": "filtered",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.storage.StringConverter"
}
```

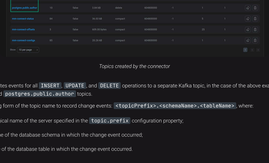
Attribute	Description
name	Name of the connector that will be used in the Kafka Connect service
connector.class	Class name for the connector
tasks.max	Maximum number of created tasks
database.hostname	PostgreSQL server instance address
database.port	PostgreSQL server instance port number
database.user	PostgreSQL server username
database.password	Password for PostgreSQL server user
database.names	Name of the database from which changes will be captured
topic.prefix	Topic prefix for a PostgreSQL server instance/cluster that forms a namespace and is used in all Kafka topic names that the connector writes to, Kafka Connect schema names, and the corresponding Avro schema namespaces when using the Avro Converter
plugin.name	Name of the PostgreSQL logical decoding module installed on the PostgreSQL server
publication.autocreate.mode	Indicates whether and how the connector creates a publication. This option only applies if the connector streams are modified using the `pgoutput` plugin

6. After filling, click *Save* and get a message about the successful creation of the connector.



Message about the successful creation of the connector

7. Check that the opened page displays the created connector in the working status.



Created connector

If after creating the connector the task is created with an error, the contents of the error can be seen after clicking ⓘ located in the *Status* field of the task.

Use the Debezium connector for PostgreSQL server

The first time you connect to a PostgreSQL server or cluster, the connector creates a consistent snapshot of all schemas. Once the snapshot is complete, the connector continuously captures row-level changes and pushes them to Kafka topics.

On the Topics page of the ADS Control user interface, you can see the topics created by the connector.

Topics created by the connector

For each table, the connector writes events for all `INSERT`, `UPDATE` and `DELETE` operations to a separate Kafka topic, in the case of the above example – the `postgres.public.book` and `postgres.public.author` topics.

The connector uses the following form of the topic name to record change events: `<topicPrefix>.<schemaName>.<tableName>`, where:

- `<topicPrefix>` – the logical name of the server specified in the `topic.prefix` configuration property;
- `<schemaName>` – the name of the database schema in which the change event occurred;
- `<tableName>` – the name of the database table in which the change event occurred.

NOTE

More detailed information about the topic names created by the Debezium connector can be found in the section Topic names of the Debezium for PostgreSQL server connector documentation.

snapshot.mode

Using the snapshot.mode parameter, you can configure the moment at which the connector takes snapshots.

Debezium connector for MS SQL Server

Olga Semmie

AD.S CONTROL

Contents

To Table of Contents

Prerequisites
Create a Debezium connector for MS SQL Server
Use the Debezium Connector for MS SQL Server
- Topic for recording change events
- Topic for recording schema changes
- snapshot.mode

The article shows an example of running and using the Debezium connector for MS SQL Server via ADS Control. The connector is designed to create records in Kafka topics about events of changes against database tables on Microsoft SQL Server.

Prerequisites

The following environment was used to create the connector:

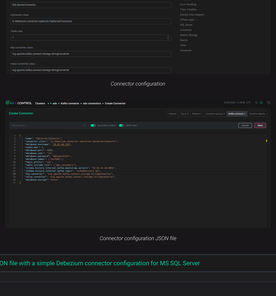
- The ADS cluster is installed according to the Online installation guide. The minimum ADS version is 3.6.2.2.b1.
- Kafka and Kafka Connect services have been installed to the ADS cluster.
- To automatically create a Kafka topic, the ***auto.create.topics.enable*** parameter is enabled in the ***server.properties*** group when configuring the Kafka service.
- The ADS Control cluster is installed according to the guide Get started with Arenadata Streaming Control and integrated with the ADS cluster being used.
- MS SQL Server is installed on the host with the Ubuntu 20.04 operating system. `p@ssword123!` password is set for the `sa` user. `test001` database and `dbo.customers` table are created under `sa` user.
- MS SQL Server IP address is `10.92.40.239`. For incoming connections, the default port number is `1433`.
- To allow the Debezium connector to capture change event records for database operations, enable change data capture (CDC):
  - for SQL Server database;
  - for table, changes to which should be captured by the connector.

Create a Debezium connector for MS SQL Server

To create a Debezium connector for MS SQL Server via ADS Control, use the ***SqlServerConnector*** connector plugin.

To create connectors using the ADS Control, you need to:

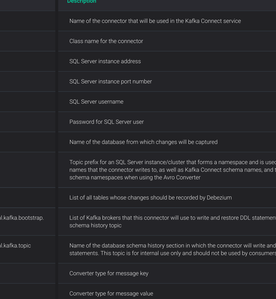
1. Go to the Kafka Connects page in the ADS Control web interface. The ***Kafka Connects*** page becomes available after selecting a cluster in the cluster management section and going to the desired tab on the General page.
2. Select the desired cluster and go to the Kafka Connect instance overview page.
3. Click ***Create Connector*** on the Kafka Connect instance overview page. After clicking ***Create Connector***, the window for selecting the connector plugin ***Clusters → <cluster name> → Kafka Connects → <cluster name> connector → Kafka connector plugins*** opens.
4. Select the desired connector to create.



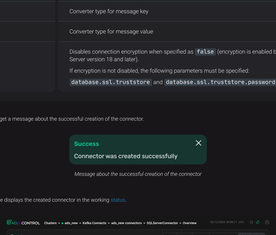
Selecting the Kafka Connect connector to create

5. Fill in the connector configuration parameters. If necessary, use the parameter information:
   - Kafka Connect service configurations in the ADS configuration parameters article;
   - Debezium connector configurations for MS SQL Server.

   You can fill in the configuration in the form of a JSON file. To do this, enable the ***JSON view*** switch.



Connector configuration



Connector configuration JSON file

Example contents of a JSON file with a simple Debezium connector configuration for MS SQL Server

```
{
    "name": "SQLServerConnector",
    "connector.class": "io.debezium.connector.sqlserver.SqlServerConnector",
    "database.hostname": "10.92.40.239",
    "tasks.max": 1,
    "database.port": 1433,
    "database.user": "sa",
    "database.password": "p@ssword123!",
    "database.names": ["test001"],
    "topic.prefix": "sql1",
    "table.include.list": ["dbo.customers"],
    "schema.history.internal.kafka.bootstrap.servers": "10.92.42.28:9092",
    "schema.history.internal.kafka.topic": "schemahistory.sql",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.storage.StringConverter",
    "database.encrypt":"false"
}
```

Attribute	Description
name	Name of the connector that will be used in the Kafka Connect service
connector.class	Class name for the connector
database.hostname	SQL Server instance address
database.port	SQL Server instance port number
database.user	SQL Server username
database.password	Password for SQL Server user
database.names	Name of the database from which changes will be captured
topic.prefix	Topic prefix for an SQL Server instance/cluster that forms a namespace and is used in all Kafka topic names that the connector writes to, as well as Kafka Connect schema names, and the corresponding Avro schema namespaces when using the Avro Converter
table.include.list	List of all tables whose changes should be recorded by Debezium
schema.history.internal.kafka.bootstrap.servers	List of Kafka brokers that this connector will use to write and restore DDL statements to the database schema history topic
schema.history.internal.kafka.topic	Name of the database schema history section in which the connector will write and restore DDL statements. This topic is for internal use only and should not be used by consumers
key.converter	Converter type for message key
value.converter	Converter type for message value
database.encrypt	Disables connection encryption when specified as `false` (encryption is enabled by default in MS SQL Server version 18 and later). If encryption is not disabled, the following parameters must be specified: `database.ssl.truststore` and `database.ssl.truststore.password`

6. After filling, click ***Save*** and get a message about the successful creation of the connector.

Message about the successful creation of the connector

7. Check that the opened page displays the created connector in the working status.

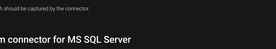
Created connector

If after creating the connector the task is created with an error, the contents of the error can be seen after clicking located in the **Status** field of the task:

Use the Debezium Connector for MS SQL Server

The first time the Debezium connector connects to an SQL Server database, it creates a consistent snapshot of the database schemas. After the initial snapshot is taken, the connector continuously commits row-level changes for `INSERT`, `UPDATE`, or `DELETE` operations. The connector creates events for each data change operation and transmits them to Kafka topics.

On the Topics page of the ADS Control user interface, you can see the topics created by the connector.



Topics created by the connector

Topic for recording change events

For each table, the connector writes events for all `INSERT`, `UPDATE`, and `DELETE` operations to a separate Kafka topic, in the case of the example above — the `sql.test001.dbo.customers` topic. The connector uses the following form of the topic name to record change events: `<topicPrefix>.<schemaName>.<tableName>`, where

- `<topicPrefix>` – the logical name of the server specified in the `topic.prefix` configuration property;
- `<schemaName>` – the name of the database schema in which the change event occurred;
- `<tableName>` – the name of the database table in which the change event occurred.

Topic for recording schema changes

For each table that has CDC enabled, the Debezium connector maintains a history of schema change events that are applied to the tables in the database. The connector writes schema change events to a separate Kafka topic, in the case of the example given — `sql`. This name is the same as the `topicPrefix` parameter specified when creating the connector.

NOTE

More detailed information about the topic names created by the Debezium connector can be found in the Topic names section of the Debezium Connector for MS SQL Server documentation.

snapshot.mode

Using the snapshot.mode parameter, you can configure the moment at which the connector takes snapshots.

Contents

Iceberg Sink Connector overview
Prerequisites
ADS and ADS Control
ADPS
ADH
Create a Debezium connector for PostgreSQL server
Create an iceberg connector
PostgreSQL database table data in ADS Control
Iceberg tables
UPSERT mode

Iceberg Sink Connector overview

Sink for loading Data Connector - data connectors for writing data from Kafka topics. Apache Iceberg tables (Data Connector) and Data Lake (Object Storage).

- Message consolidation for activated loading controlling.
- Supporting [UPSERT](#) operations.
- Ability to flow schema-related tables.
- Automatic table maintenance.
- Real-time mapping using catalog tables changing the records.

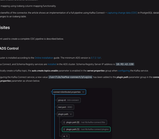
Iceberg Sink Connector is a Kafka Connect connector for writing the data from Kafka topics to Apache Iceberg tables (Data Connector) and Data Lake (Object Storage) tables. Iceberg Sink Connector is developed for PostgreSQL database tables with a primary key and for Debezium connector for PostgreSQL server.

Prerequisites

The environment used to create a complete CDC pipeline is described below.

ADS and ADS Control

- The ADS cluster is installed according to the [Arenadata Installation](#) guide. The minimum ADS version is [3.2.7.0](#).
- Kafka, Kafka Connect, and Zookeeper services are [activated](#) in ADS cluster. Connecting Kafka to Kafka Connect: **10.192.40.238**.
- The ADS Control cluster is installed according to the [ADS Control Installation](#) guide. The minimum ADS Control version is [2.2.7.0](#).
- Add a configuration file to the Kafka Connect cluster and create an [Export Kafka Connect configuration](#) task. The task creates the [export.properties](#) configuration file in the [Kafka Connect directory](#) on the [Kafka Connect hosts](#) and [activates](#) it.



Adding the plugin path is required only for [HDFS 3.3.2](#).

To configure the Kafka Connect cluster, open the configuration file and specify the path to the plugin directory in the [plugin.path](#) parameter. The path must match the [Target path](#) value in the [Create configuration](#) task for the name of the ADH cluster and click the [Import](#) button.

Import of Kafka Connect configuration

As a result of the import, the configuration file is displayed in the [export.properties](#) file editor. Click the [Save](#) button to save the changes.

Editing Kafka Connect configuration

As a result of the import, the configuration file is created in the [/usr/hdp/kafka-connect/](#) directory on the [Kafka Connect hosts](#) and [activated](#) in ADS Control.

ADPS

- The ADP cluster is installed according to the [Arenadata Installation](#) guide.
- Postgresql_conn of Kafka is activated with the ADPS service. Set [wal_level](#) to [logical](#) for streaming connections, the default port number is [5432](#).
- To create a Debezium connector for PostgreSQL server, you must create a database user with the [REPLICATION](#) role and the [SELECT](#) or [ALL PRIVILEGES](#) access to the database from the Kafka Connect host in the [pg_hba.conf](#) file and the [wal_level](#) parameter in the [postgresql.conf](#) file.

Create a test database

Create a user with the [REPLICATION](#) role:

```
CREATE USER db_replication PASSWORD 'Password';
```

Create the replication:

```
ALTER ROLE db_replication WITH REPLICATION;
```

Create a user with [SUPERUSER](#) role:

```
CREATE USER db_user WITH SUPERUSER PASSWORD 'Password';
```

> **Caution**
> The [SUPERUSER](#) can be used for testing purposes only. For more information, see [Creating a user for the Debezium connector](#).

Create a test table:

```
CREATE TABLE db_table (
    id SERIAL NOT NULL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description VARCHAR(512),
    weight FLOAT
);
```

Configure replica identity for a table to enable the use of [REGIONAL_PRIVACY](#) clause in the connector rule for it to the topic:

```
ALTER TABLE db_table REPLICA IDENTITY FULL;
```

Fill the table with data (generate an example, several rows are needed for tests):

```
INSERT INTO db_table(name, description) VALUES ('Jane Doe', 'Jane''s description');
```

ADH

- The Kerberos server is installed according to the [Kerberos Installation](#) guide.
- The ZOOKEEPER service is installed according to the [ADH Installation](#) guide. The minimum ADH version is [3.3.4.0](#).
- The HDFS service is installed according to the [ADH Installation](#) guide, which must have rights to connect to the HDFS storage from the [Kafka Connect](#) group.

Iceberg Sink Connector with Kerberos

Setup for the [HDFS 3.3.2](#) connector. The Kerberos settings are described in the [HDFS 3.3.2 Sink Connector](#) configuration parameters table.

Also, when Kerberos authentication is activated in the Hadoop cluster for the sink task, the [principal](#) is required.

To setup, add the [core-site.xml](#) and [hdfs-site.xml](#) files. Corresponding settings are placed into the [hdfs.conf](#) file:

- [HDFS 3.3.2 hdfs.conf](#) is used as an example.
- [HDFS 3.3.2 hdfs-site.xml](#) is also used.

Put both of these files into the [/etc/](#) directory on all Kafka Connect hosts.

To configure the Kerberos connector, add the following properties to the [connect-distributed.properties](#) file:

```
sasl.kerberos.service.name=hdfs
sasl.jaas.config=com.sun.security.auth.module.Krb5LoginModule required \
    useKeyTab=true \
    storeKey=true \
    keyTab="/etc/security/keytabs/hdfs.headless.keytab" \
    principal="hdfs-adh@ADH.LOCAL";
```

Where:

- [useKeyTab=true](#) - the property specifies that the connector will use the keytab file.
- [storeKey=true](#) - the property specifies that the keytab file will store the key.
- [keyTab="/etc/security/keytabs/hdfs.headless.keytab"](#) - the property specifies the path to the keytab file.
- [principal="hdfs-adh@ADH.LOCAL"](#) - the property specifies the principal to be used.

Create a Debezium connector for PostgreSQL server

To create a Debezium connector for PostgreSQL server, you must create a [pgoutput](#) logical replication stream and configure the user and access permissions for the connector. Refer to the [Debezium connector for PostgreSQL server](#) documentation for more information.

Below is an example of configuration parameters for the Debezium connector for PostgreSQL server.

Example of the contents of the Debezium connector configuration JSON file for PostgreSQL server

```
{
    "name": "cdc-to-ceph",
    "tasks.max": "1",
    "plugin.name": "pgoutput",
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "database.server.name": "pgtest",
    "database.hostname": "10.92.3.34",
    "database.port": "5432",
    "database.user": "db_user",
    "database.password": "Q!1a2z",
    "database.dbname" : "Research",
    "table.include.list": "public.(.*)",
    "heartbeat.interval.ms": "5000",
    "publication.autocreate.mode": "filtered",
    "tombstones.on.delete":"false",
    "transforms": "AddPrefix",
    "transforms.AddPrefix.type": "org.apache.kafka.connect.transforms.RegexRouter",
    "transforms.AddPrefix.regex": "([^.]+)\\.([^.]+)\\.([^.]+)",
    "transforms.AddPrefix.replacement": "$3",
    "key.converter": "org.apache.kafka.connect.storage.StringConverter",
    "value.converter": "org.apache.kafka.connect.storage.StringConverter"
}
```

Attribute	Description
plugin.name	Schema flagging output options
key.converter & schema.registry.url	Schema flagging output options
transforms.unwrap.type	Specifies the class used for the [UNWRAP](#) transformation
transforms.unwrap.add.fields	Specifies the fields to add to the message header
transforms.unwrap.delete.handling.mode	Defines how to handle delete events. [DROP](#) the whole flow when [delete.handling.mode=drop](#) is set
table.include.list	Specifies the tables for the connector to track. Use a comma-separated list of the table names, prepend each of the names with the schema name (for example, [public.table1,public.table2](#)). To change only from the specified tables, specify [table.include.list](#) with the [public.(.*)](#) value
slot.name	Name of the PostgreSQL [replication slot](#) that the Debezium connector uses for streaming changes to a specific database. The default value is [debezium](#). The connector creates a slot for each database that it follows. The connector will fail if the slot is not unique for the server
publication.name	Name of the PostgreSQL [publication](#) created for streaming changes when using [pgoutput](#). This plugin and publication name will be used if the publication does not exist when the connector is started. If the specified publication does not exist, the connector tries to create a publication using the database user account that has sufficient rights. If the publication already exists, either the connector user must have sufficient rights to execute the [CREATE PUBLICATION xxx](#) command or the user must be the owner of the publication. Otherwise, the connector stops with an error. If [publication.autocreate.mode](#) is set to [filtered](#), the connector does not use this configuration parameter. The connector uses the configuration that is specified for the [table.include.list](#) parameter instead
publication.autocreate.mode	Applies only when streaming changes by using the [pgoutput](#) plug-in. Determines how creation of a publication should work. Specify the following values: • [all_tables](#) - Default value. The connector runs the [CREATE PUBLICATION <publication_name> FOR ALL TABLES](#) statement if the publication does not exist. • [disabled](#) - The connector does not attempt to create a publication. A database administrator or the user configured to create publications must create it. Otherwise, the connector stops with an error. • [filtered](#) - The connector creates a publication with the tables that are specified in the [table.include.list](#) configuration property. If the publication exists, the connector updates the publication to include the tables that are specified in the [table.include.list](#) configuration property. If the publication exists and the [table.include.list](#) configuration property is not specified, the connector updates the publication to include all tables in the database.
snapshot.mode	Specifies the criteria for performing a snapshot when the connector starts. Possible values are: • [initial](#) - The connector performs a snapshot only when no offsets have been recorded for the logical server name. • [always](#) - The connector performs a snapshot each time it starts. After the snapshot completes, the connector continues to stream changes from the WAL. • [never](#) - The connector never performs snapshots. When a connector is configured this way, its behavior when it starts for the first time is defined by the [snapshot.new.tables](#) configuration property.
snapshot.new.tables	Specifies how the connector should handle data from tables that are added to the list of tables that are configured for the [table.include.list](#) property after the initial snapshot was taken. Possible values are: • [parallel](#) - Default value. The connector performs a snapshot of new tables in parallel with streaming from the WAL. • [off](#) - The connector does not perform snapshots for new tables.
tombstones.on.delete	Controls whether a tombstone event is generated after a delete event. Possible values are: • [true](#) - Default value. After a delete event is generated, a tombstone event is also generated. • [false](#) - A tombstone event is not generated.
slot.name	Name of the PostgreSQL [logical decoding slot](#) created for streaming. The value can be any lowercase letter, number, and the underscore character. The default value is [debezium](#). Each connector must have a unique name. When a connector is restarted, it can resume from the last point where it stopped streaming changes.
schema.exclude.list	Specifies the schemas for the connector to exclude. Use a comma-separated list of the schema names. Specify [schema.exclude.list](#) or [schema.include.list](#), but not both.
schema.include.list	Specifies the schemas for the connector to track. Use a comma-separated list of the schema names. Specify [schema.include.list](#) or [schema.exclude.list](#), but not both.

Create an iceberg connector

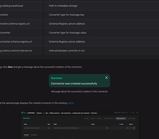
To create an iceberg connector via ADS Control, the connector of type [IcebergConnector](#) is used.

Create a connector configuration file:

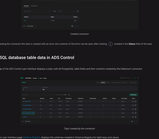
1. Go to the [Connectors](#) page in the ADS Control user interface. The [Kafka Connect](#) tab shows available clusters. After clicking a [Create](#) in the cluster management action panel, the [Create Connector](#) window opens.
2. Specify the connector configuration in the [Add configuration manually](#) window.
3. Click the [Create Connector](#) button.

Creating a connector

You can fill in the configuration in the form, or in JSON file. To fill in the form, enable the [Edit mode](#) switch.



Connector configuration



Connector configuration, JSON file

Example contents of a JSON file with an Iceberg connector configuration

```
{
  "connector.class": "org.apache.iceberg.connect.IcebergSinkConnector",
  "tasks.max": "2",
  "topics": "pgtest.public.db_table",
  "iceberg.tables": "db_table",
  "iceberg.catalog.s3.endpoint": "http://10.92.3.53:9000",
  "iceberg.catalog.s3.access-key-id": "ADMIN",
  "iceberg.catalog.s3.secret-access-key": "ADMIN123",
  "iceberg.catalog.client.region": "us-east-1",
  "iceberg.catalog.s3.path-style-access": "true",
  "iceberg.catalog.uri": "thrift://10.92.3.53:9083",
  "iceberg.catalog.warehouse": "s3a://warehouse",
  "iceberg.catalog.type": "hive",
  "iceberg.control.commit.interval-ms": 1000,
  "iceberg.control.commit.timeout-ms": 1000,
  "iceberg.catalog.s3.pathstyle": "true",
  "iceberg.catalog.io-impl": "org.apache.iceberg.aws.s3.S3FileIO",
  "iceberg.catalog.s3.ssl.enabled": "false",
  "iceberg.control.topic": "cdc-control-topic",
  "iceberg.catalog.s3.preload-client": "true",
  "iceberg.catalog.client.region": "us-east-1",
  "iceberg.catalog.s3.connection-timeout-ms": 5000,
  "iceberg.catalog.s3.socket-timeout-ms": 5000,
  "iceberg.catalog.s3.checksum-enabled": "false",
  "iceberg.catalog.s3.chunked-encoding-enabled": "false"
}
```

Attribute	Description
name	Name of the connector that will be used in the Kafka Connect service
connector.class	Class name for the connector
tasks.max	Maximum number of tasks to create
topics	Name of the topic with data for the connector. Specify the topics or [topics.regex](#), but not both. Example: [topic1,topic2,topic3](#)
topics.regex	Regular expression that specifies the topics with data for the connector. Specify the topics or [topics](#), but not both. Example: [topic.*](#)
iceberg.tables	Comma-separated list of destination tables
iceberg.tables.dynamic-enabled	Default file format for a new table: [parquet](#), [avro](#), or [orc](#)
iceberg.tables.auto-create-enabled	Set to [true](#) to auto-create a new target table
iceberg.tables.evolve-schema-enabled	Set to [true](#) to add columns to a table when necessary
iceberg.tables.upsert-mode-enabled	Set the value to [true](#) to enable [UPSERT mode](#)
iceberg.tables.cdc-field	Specifies the name of the field that contains the type of operation ([INSERT](#), [UPDATE](#), or [DELETE](#)) for the record
iceberg.tables.route-field	Setting specifies the name of the field whose value routes a record to a particular table
iceberg.tables.default-commit-branch	Commit branch for new snapshots. Default is [main](#)
iceberg.tables.default-id-columns	List of columns that uniquely identify rows
iceberg.tables.default-partition-by	Applies to a record if the catalog and Iceberg table's [partition.by](#) field is not set. Use [hour(ts)](#) for hourly partitions
iceberg.tables.auto-create-properties	Properties to add to newly created tables
iceberg.tables.write-properties	Properties to add to newly written tables
iceberg.control.commit.interval-ms	The value is the interval in milliseconds that a commit is initiated. The default value is [300000](#) (5 minutes)
iceberg.control.commit.timeout-ms	The value is the commit timeout interval in milliseconds. The default value is [30000](#) (30 seconds)
iceberg.control.commit.threads	The value is the number of threads to use for commits. The default value is the number of cores multiplied by [2](#), but with a minimum value of [5](#)
iceberg.control.group-id	Name of the consumer group to store offsets. The default value is [cg-control-{connectorName}](#)
iceberg.control.topic	Name of the topic used to manage commits. The default value is [control-iceberg](#)
iceberg.control.admin.timeout-ms	The value is the timeout interval in milliseconds for admin operations. The default value is [60000](#) (1 minute)
iceberg.control.consumer.poll.timeout-ms	The value is the timeout interval in milliseconds for consumer polling. The default value is [1000](#) (1 second

SMT for Iceberg Sink Connector

by ADS CONTROL

Contents

Overview of SMT
CopyValue
DmsTransform
DebeziumTransform
JsonToMapTransform
KafkaMetadataTransform
MongoDebeziumTransform
Example of SMT

To Table of Contents

Overview of SMT

Single Message Transform (SMT) - a ready-made component with simple logic for a single message transformation.

SMT transformations are part of the Kafka Connect architecture, used to modify data before writing it to Kafka and to modify data from Kafka, simplifying event processing and routing.

SMT transformations are not mandatory.

The configuration of transformations is performed as part of the instance configuration of the connector via the following parameters:

```
{
"transforms": "value1",
"transforms.value1.type": "value.type",
"transforms.value1.transformConfig": "valueTransformConfig"
}
```

where

- `value1` - the name of the transformation. names can be listed separated by commas in the order the transformations are applied.
- `value.type` - the full class name for the transformation.
- `transformConfig` - the configuration parameter for a specific transformation. Some transformations may not have parameters.
- `valueTransformConfig` - the value of the transformation's configuration parameter.

Starting with ADS 3.5.1.b1, in ADS Control, several types (classes) of SMT are available for configuration by default for the Iceberg Sink Connector:

- io.tabular.iceberg.connect.transforms.CopyValue
- io.tabular.iceberg.connect.transforms.DmsTransform
- io.tabular.iceberg.connect.transforms.DebeziumTransform
- io.tabular.iceberg.connect.transforms.JsonToMapTransform
- io.tabular.iceberg.connect.transforms.KafkaMetadataTransform
- io.tabular.iceberg.connect.transforms.MongoDebeziumTransform

CopyValue

`CopyValue` copies a value from one field to a new field.

The configuration parameters for the `CopyValue` transformation are described below.

Parameter	Description
source.field	Name of the field from which the value is copied
target.field	Name of the field into which the value is copied

DmsTransform

`DmsTransform` converts a message into the format used in AWS Database Migration Service (DMS), for ingesting data to a target store using the CDC feature. Applying the transformation produces the fields `before` for the key-level and adds metadata fields `_cdc_op`, `_cdc_ts`, and `_cdc_source`.

There are no configuration parameters for the `DmsTransform` transformation.

DebeziumTransform

`DebeziumTransform` converts a message into the format used in the Debezium platform for ingesting data to the target storage using the CDC function.

The application of the transformation performs:

- moving `before` or `after` fields associated with data collections to the top level (if they were previously added using the partition.payload.field parameter).
- adding metadata fields `_cdc_op`, `_cdc_ts`, `_cdc_offset`, `_cdc_source`, `_cdc_target`, and `_cdc_key`.

The configuration parameters for `DebeziumTransform` are described below.

Parameter	Description	Default value
cdc.target.pattern	Template used to set the value of the CDC target field	«db».«table»

JsonToMapTransform

`JsonToMapTransform` converts data of type `String` into JSON objects for schema creation.

Iceberg Sink Connector can be used to transform data with a Map structure without a schema into iceberg structures.

For poorly structured JSON objects with dynamically changing keys, using the Iceberg Sink Connector can lead to an unlimited increase in the number of columns in an Iceberg table due to schema evolution. In such a situation, JsonToMapTransform can be used so that data can be loaded into Iceberg and then processed with queries.

Using the `JsonToMapTransform` transformation provides:

- Conversion of nested objects into a Map and subspace as a Map type schema.
- Creation of Iceberg tables with Iceberg Map (String) columns for JSON objects according to the schema.

> **NOTE**
>
> - When using `JsonToMapTransform`, `value.converter` should be set to `StringConverter` not `JsonConverter`.
> - Message keys, tombstones, and headers are not transformed and are passed to the SMT in their original form.
> - Keys with empty arrays and empty objects are excluded from the resulting schema.
> - If JSON arrays contain elements of different types, they are converted into arrays of strings.

The configuration parameters for `JsonToMapTransform` are described below.

Parameter	Description	Default value
json.root	Specifies the type of structure being created: • `false` — (**Strict**) will be created with inferred schemas for primitive and array fields. Nested objects will become fields of type `MapString, String>`. • `true` — intended for the most inconsistent data. It will create a (**Strict**) with a single field named `payload` with a schema structure of type `MapString, String>`.	false

Below is the difference between the modified messages with different values of the `json.root` parameter.

Original message

```
{
"key": 1,
"array": [1,"two",3],
"empty_obj": {},
"nested_obj": {"some_key": {"one","two"}}
}
```

Modified message with `json.root=true`

```
{jsonRecord schema:
  "payload": (Optional) Map<String, String>

$jsonRecord value (Struct):
{payload: Map
  "key": "1"
  "array": "[1,"two",3]"
  "empty_obj": "{}"
  "nested_obj": "{"some_key":{"one","two"}}"
}
```

Modified message with `json.root=false`

```
{jsonRecord schema:
  "key": (Optional) Int32,
  "array": (Optional) Array<String>,
  "nested_object": (Optional) Map<String, String>

$jsonRecord value (Struct):
{key: 1
  "array": ["1", "two", "3"]
  "nested_object" Map {"some_key" : {"one", "two"}}
}
```

KafkaMetadataTransform

`KafkaMetadataTransform` adds Kafka message metadata fields `topic`, `partition`, `offset`, `timestamp`.

The configuration parameters for the `KafkaMetadataTransform` transformation are described below.

Parameter	Description	Default value
field_name	Prefix for fields	_kafka_metadata
nested	Determines the level for adding metadata: • `false` — metadata is added at the top level with the prefix `_kafka_metadata_topic`, `_kafka_metadata_partition`, `_kafka_metadata_offset`, `_kafka_metadata_timestamp`. • `true` — metadata is added with a structure `_kafka_metadata_topic`, `_kafka_metadata_partition`, `_kafka_metadata_offset`, `_kafka_metadata_timestamp`.	false
external_field	Adds a permanent key/value pair to the metadata (for example, the cluster name)	–

MongoDebeziumTransform

`MongoDebeziumTransform` transforms a message received via the connector Debezium from the format used in the MongoDB database (with BSON strings and `before` or `after` fields) into `before` or `after` fields in JSON structures, which can be modified using the `KafkaMetadata` transformation.

The configuration parameters for `MongoDebeziumTransform` are described below.

Parameter	Description
array_handling_mode	An array or a BSON document for setting the array processing mode

Example of SMT

> **NOTE**
>
> The example provided in the article is based on the pipeline described in the Iceberg connector article.

Below is the `my_table` iceberg table, displayed in the HUE interface as a result of running the Iceberg Sink Connector without using SMT transformations. The table shows only the fields contained in the `sdc.dms_public.my_table` topic

Iceberg table

To implement SMT transformations, create the `my_newest_table` iceberg table for the `sdc.dms_public.my_table` topic

When creating a new table in the connector configuration, add parameters for implementing the CopyValue and KafkaMetadataTransform transformations

The new parameters are shown below.

```
{
"transforms": [
  "copyName",
  "kafkaMetaData"
],
"transforms.copyName.type": "io.tabular.iceberg.connect.transforms.CopyValue",
"transforms.copyName.source.field": "name",
"transforms.copyName.target.field": "name_copy",
"transforms.kafkaMetaData.type": "io.tabular.iceberg.connect.transforms.KafkaMetadataTransform",
"transforms.kafkaMetaData.nested": "true"
}
```

where

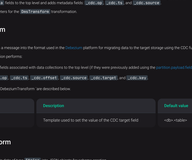
- The `copyName` transformation creates a `name_copy` field, which fully copies the `name` field.
- The `kafkaMetaData` transformation creates a `_kafka_metadata` metadata group.

Below is an example of a connector configuration JSON file using the new parameters.

Example of a JSON file of an Iceberg connector configuration with SMT

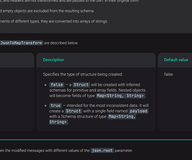
```
{
  "connector.class": "org.apache.iceberg.connect.IcebergSinkConnector",
  "iceberg.tables.cdc-field": "_cdc",
  "name.ser": 1,
  "transforms": [
    "copyName",
    "kafkaMetaData"
  ],
  "iceberg.tables.upsert-mode-enabled": true,
  "iceberg.tables.auto-create-enabled": true,
  "transforms.copyName.type": "io.tabular.iceberg.connect.transforms.CopyValue",
  "iceberg.tables": [
    "default.my_newest_table"
  ],
  "transforms.kafkaMetaData.type": "io.tabular.iceberg.connect.transforms.KafkaMetadataTransform",
  "value.converter.value.subject.name.strategy": "io.confluent.connect.avro.AvroConverter",
  "key.converter": "io.confluent.connect.avro.AvroConverter",
  "transforms.copyName.target.field": "name_copy",
  "topics": [
    "sdc.dms_public.my_table"
  ],
  "iceberg.control.commit.interval-ms": 30000,
  "iceberg.catalog.s3.endpoint": "http://10.92.42.153:8001",
  "value.converter.schema.registry.url": "http://10.92.42.153:8081",
  "transforms.kafkaMetaData.nested": "true",
  "transforms.copyName.source.field": "name",
  "name": "iceberg-sink-cdc-newestconnector",
  "iceberg.tables.schema-case-insensitive": true,
  "iceberg.catalog.warehouse": "s3a://warehouse/hive/warehouse",
  "iceberg.tables.default-id-columns": "id",
  "iceberg.control.topic": "ctl-test",
  "key.converter.schema.registry.url": "http://10.92.42.153:8081",
  "iceberg.write.format.default": "parquet",
  "iceberg.hadoop-conf-dir": "/usr/lib/hadoop/config"
}
```

As a result of running, `iceberg-sink-cdc-newestconnector`, the created `my_newest_table` table displays the `name_copy` and `_kafka_metadata` fields in the HUE interface



Iceberg table with new fields

On the *Table Browser* - *Databases* - *default* - *my_newest_table* - *Overview* HUE page, there is a table schema containing the new fields.



Iceberg table schema

Hovering over the `_kafka_metadata` field opens a structure that contains the record's metadata fields.

Kafka_metadata field structure

ADS Control configuration parameters

Ray Mitchell

To Table of Contents

This article describes the parameters that can be configured for ADS Control version 4x.XX4. To read about the configuration process, refer to the *Configure ADS Control* article.

NOTE

Some of the parameters described in the ADCM UI offer the **Show advanced** flag being set.

The configuration descriptions use the concepts of the *Eclipse Vert.x* framework.

- **Verticle** – a piece of code that is an event loop which can be deployed in the Vert.x environment.
- **Worker verticle** – a verticle that runs using a thread from the pool of already created Vert.x worker threads, rather than using a new event loop.

General

Parameter	Description	Default value
server.http.port	Port to connect to ADS Control	8888
server.vertx.pools.eventLoopPoolSize	Maximum number of Event Loop verticles in the pool	8
server.vertx.pools.workerPoolSize	Maximum number of working verticles in the pool	20
server.vertx.config.clusterResponseTimeoutMs	Time interval to request getting metrics from Kafka cluster (in ms)	500
server.vertx.event.bus.requestTimeoutMs	Time interval for requesting the Event Bus of the Vert.x server (in ms)	30000
server.vertx.event.bus.consumer.deliveryOptions.timeout	Maximum number of attempts for consumer requests on the Event bus before the connection is terminated by the circuit breaker pattern	3
server.vertx.event.bus.consumer.circuitBreaker.maxRetries	Time interval of parameters for delivering consumer requests to the Event Bus	7200000
server.vertx.event.bus.producer.circuitBreaker.maxRetries	Maximum number of attempts to request manufacturer requests on the Event bus before the connection is terminated by the circuit breaker pattern	0
server.vertx.event.bus.producer.deliveryOptions.timeout	Time interval of parameters for delivery of manufacturer requests to the Event Bus	7200000
server.file-extensions	Possible server file extensions	json,yml,cnt,mpng
server.session-consumer-cache-enabled	Enables caching for consumer sessions	true
server.session-consumer-cache.capacity	Consumer session cache data size	1000
server.session-consumer-cache.expireTime	Consumer session cache expiration time	20000
server.session-consumer-cache.cleanDelay	Consumer session cache clear delay time	10000
server.producer-cache.capacity	Producers cache data size	1000
server.producer-cache.expireTime	Producers data cache expiration time	60000
server.streaming.consumer.group.offPaths	Paths that is added to the Kafka consumer group identifier	adscc
server.streaming.consumer.group.inactiveSTmoutCheckInterval	Amount of time a data consumer requests can remain inactive (i.e. with no consumer signals) before the broker initiates a rebalance (in ms)	600000
server.streaming.consumer.group.inactiveGroupsCheckInterval	Interval between consumer group activity checks (in ms)	120000
server.streaming.consumer.messagePollTimeout	Timeout between polls for new messages from the Kafka server (in ms)	1000
server.streaming.consumer.maxPollRecords	The maximum number of records the consumer will retrieve in a single poll call	10

Kafka Workers

Parameter	Description	Default value
server.vertx.verticle.kafka-common-offset-info.producer.instances	Number of processes of **kafka-invalid-offset-info-producer** verticles for receiving information about the end of the **__consumer_offsets** topic (data about the last message read by the consumer	1
server.vertx.verticle.application-state-verticle.instances	Number of **application-state** verticle instances collecting application state data	5
server.vertx.verticle.application-state-verticle.updatePeriodOffsetsMs	Period between requests for application state data (in ms)	5050
server.vertx.verticle.application-state-verticle.updatePeriodOffsetsMaxSeconds	Period between requests for data on the latest offsets (in ms)	30000

Kafka Consumer

Parameter	Description	Default value
server.vertx.kafka.lastOffsetNotTimeoutMs	Timeout of the last poll of the topic offset **__consumer_offsets** (data about the last message read by the consumer) (in ms)	200

Kafka Settings

Parameter	Description	Default value
server.kafka.timeout	Timeout waiting for a response to a request to the Kafka server	300000
server.kafka.describeTopicsTimeout	Timeout waiting for a response to a request for a topic description to the Kafka server	10000
server.kafka.messagePollTimeout	Timeout between polling new messages from the Kafka server (in ms)	2000
server.kafka.systemGroupId	Groupid for connecting to Kafka server	adscc

Basic authentication

Parameter	Description	Default value
Admin Password	ADS Control administrator password	--
Basic authentication users	Form for filling out data for new users • Enter field name – enter the user name. • Enter secret – enter a password for the user. • Confirm – password confirmation. For more information, see *Manage users*.	--

LDAP Authentication

Parameter	Description	Default value
LDAP URL	LDAP server URL to enable *user synchronization* during authentication	--
Authentication Query	Query to find users in LDAP directory to enable *user synchronization* during authentication	--

Logging

The **logback_template** parameter defines the log templates.

Default value:

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

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

    <appender name="FILE-ROLLING" class="ch.qos.logback.core.rolling.RollingFileAppender">

        <rollingPolicy class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>logs/archived/aap.%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{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
        </encoder>
    </appender>
    <root level="INFO">
        <appender-ref ref="STDOUT" />
    </root>
    <logger name="ru.arenadata.adscc" level="DEBUG">
        <appender-ref ref="FILE-ROLLING" />
    </logger>
    <logger name="ru.arenadata.adscc" level="DEBUG">
        <appender-ref ref="STDOUT" />
    </logger>
    <logger name="org.apache.kafka" level="ERROR">
        <appender-ref ref="STDOUT" />
    </logger>
</configuration>
```

Monitoring

Parameter	Description	Default value
server.monitoring.openTelemetry.enable	Enables the monitoring system "OpenTelemetry"	ON
server.monitoring.prometheus.enable	Enables the monitoring system "Prometheus"	ON

Client

Parameter	Description	Default value
client.http.webClientOptions.maxPoolSize	Maximum pool size for connections	100
client.http.webClientOptions.verifyHost	Enables client host verification	false
client.http.circuitbreaker.maxFailures	Number of failures at which the Circuit Breaker pattern disables the execution of the operation	3
client.http.circuitbreaker.timeout	Value above which the request increases the value of the number of failures (in ms)	500
client.http.circuitbreaker.fallbackOnFailure	Sets whether or not the fallback is executed on failure, even when the circuit is closed	false
client.http.circuitbreaker.resetTimeout	Time after which the circuit breaker pattern will try to close the circuit again (in ms)	150

Kafka Clusters

After clicking **Set** the **clusters** field appears for integration with external Kafka Clusters.

Group	Parameter	Description
--	name	Kafka cluster name
--	bootstrapServers	Bootstrap Kafka cluster servers
--	schemaRegistryUrl	URL to connect to Schema Registry
--	jmxClientProperties	JMX port
kafkaConnectClusters	name	Kafka Connect cluster name
	url	URL to connect to Kafka Connect
	jmxClientProperties	JMX port

JAAS template file

The custom jaas.conf file template is intended for specifying user data for connecting clients of other services to the current service (paths to keytab files, the **useTicketCache** parameter and others). For more information, see *Configure a custom jaas.conf*.

Default value:

```
{% if cluster_config.kerberos_client and cluster_config.kerberos_client.enable_kerberos %}
KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule required
        useKeyTab=true
        storeKey=true
        useTicketCache=false
        serviceName="kafka"
        keyTab="{{ cluster_config.kerberos_client.keytab_dir }}/adscc.service.keytab"
        principal="adscc/{{ ansible_fqdn }}@{{ cluster_config.kerberos_client.realm }}";
};
{% endif %}
```

Advanced docker settings

Parameter	Description	Default value
Install docker package	Enables installation of the Docker package when installing the service	true
Disable seccomp	Disables the seccomp mechanism when installing a service	false

Parameter	Description	Default value
ADS Control Ranger plugin enabled	Indicates whether Ranger ADS Control plugin is enabled (auto-populated)	false

ranger-adscc-security.xml

Apache Ranger parameters

Parameter	Description	Default value
ranger.plugin.adscc.policy.rest.url	URL to Ranger Admin	--
ranger.plugin.adscc.service.name	Name of the Ranger Service containing policies for this ADS Control instance	--
ranger.plugin.adscc.policy.rest.ssl.config.file	Location of the file containing SSL data for connecting to Ranger Admin	--
ranger.plugin.adscc.policy.cache.dir	Directory where Ranger policies are cached after successful retrieval from the server	/tmp
ranger.plugin.adscc.policy.pollIntervalMs	How often to poll for changes in policies	--
ranger.plugin.adscc.policy.rest.client.connection.timeoutMs	ADS Control plugin RangerRestClient connection timeout in milliseconds	--
ranger.plugin.adscc.policy.rest.client.read.timeoutMs	ADS Control plugin RangerRestClient read timeout in milliseconds	--
ranger.plugin.adscc.use.rangerGroups	Enables the use of group information when evaluating user permissions	true
Add key, value	Parameters and their values entered in this field override the parameters specified in the ADCM user interface. This field also allows you to set values for all user parameters that are not displayed in the interface, but are allowed in the ranger-adscc-audit.xml configuration file	--

ranger-adscc-audit.xml

Parameter	Description	Default value
xasecure.audit.is.enabled	Enables audit for Ranger plugins	true
xasecure.audit.destination.solr	Enables audit in Solr for Ranger plugins	true
xasecure.audit.destination.solr.urls	Defines Solr URL	--
xasecure.audit.destination.solr.batch.filespool.dir	Directory for Solr audit spool	/tmp
Add key, value	Parameters and their values entered in this field override the parameters specified in the ADCM user interface. This field also allows you to set values for all user parameters that are not displayed in the interface, but are allowed in the ranger-adscc-audit.xml configuration file	--

ranger-policymgr-ssl.xml

Parameter	Description	Default value
xasecure.policymgr.clientssl.keystore	Location of the keystore file	--
xasecure.policymgr.clientssl.keystore.password	Keystore password	--
xasecure.policymgr.clientssl.truststore	Location of the truststore file	--
xasecure.policymgr.clientssl.truststore.credential.file	Truststore password	--
xasecure.policymgr.clientssl.keystore.credential.file	Location of the keystore credential file	/etc/adscc/conf/ssl/adscc.keystore.jceks
xasecure.policymgr.clientssl.truststore.credential.file	Location of the truststore credential file	/etc/adscc/conf/ssl/adscc.truststore.jceks
Add key, value	Parameters and their values entered in this field override the parameters specified in the ADCM user interface. This field also allows you to set values for all user parameters that are not displayed in the interface, but are allowed in the ranger-policymgr-ssl.xml configuration file	--

ADS Control releases

Olga Sarmina

ADS.CONTROL

Contents

To Table of Contents

2.6.2
2.6.0
2.5.0
2.4.0
2.3.0
2.2.0
2.1.1
1.0.0

2.6.2

2.6.2.b1

Date: 22.12.2025

Improvements

Removed the ability to compare schemas when only one version of the schema is available

Fixed sorting for large numerical values in tables

Bug fixes

Fixed an issue that occurred when sending a message in a format other than JSON

Fixed issues that occurred when reading a message if the cluster does not have the Schema-Registry service

Fixed an issue that occurred when applying a Cruise Control proposal — rebalance was not performed due to the default dry-run mode being enabled for requests

Fixed an issue that occurred when sending a message to a specified partition

Fixed a crash of `FlatStreamSinkConnector` that occurred when publishing messages to a topic

Misc/Internal

Release 2.6.1 was skipped due to critical vulnerabilities

2.6.0

2.6.0.b1

Date: 28.11.2025

New features

To provide support for Cruise Control, a ***Clusters → «cluster name» → Cruise Control*** page has been implemented in the ADS Control UI. The page contains the following tabs:

- ***Proposals***
- ***Tasks execution History***

Added support for Astra Linux 1.7 "Voronezh"

Added support for ARLinux 12 SP

By default, Java 21 is pre-installed from the `arenadata` repository for installing the ADS Control service

A utility has been implemented for explicitly selecting Java for the service

Docker is no longer required to install the ADS Control service

Added the ability to change the number of partitions on the ***Clusters → «cluster name» → Topics → Partitions*** tab

Added the ability to write messages in the Avro format

Improved display of message keys and large messages when reading

Added the ability to use a schema template when writing a message

Messages in the Avro format are now converted to a readable format when integrating with Schema-Registry

New connector management features have been added to Kafka Connect:

- Support for the `Stopped` status for connectors
- Support for selecting the target status when creating a connector
- Support for viewing and managing connector offsets (for detailed information, refer to the KIP-875 description).

Bug fixes

Fixed incorrect response status code when updating topics in some cases

Misc/Internal

This version adds the ability to automatically install Java from an Arenadata repository when installing a cluster. However, Arenadata is not an official JDK provider, and this feature is provided to ensure a fully automatic installation on systems that do not have the necessary dependencies. In production environments, we recommend using a JDK from official providers.

Minimum ADCM version — 2.6.0

In this version, the ***Cruise Control*** tab requires Kerberos authentication and SSL to be enabled on the ADS cluster. This issue will be fixed in the next version

2.5.0

2.5.0.b1

Date: 30.06.2025

New features

Added the ability to view a list of schemas associated with a topic

Added schema verification during creation of schemas

Implemented support for SASL_PLAINTEXT authentication in ADS

Improvements

Improved the Messages tab for a topic:

- added scrolling to the list of values for the ***Partition*** filter;
- message reading is divided into two tabs – ***Batch*** and ***Streaming***
- stream reading is optimized;
- messages (values and keys) in JSON format are now displayed in a human-readable form.

Bug fixes

Fixed incorrect usage of the `keystore` and `truststore` variables

Fixed a consumer group was not deleted after inactivityTimeout expired

Fixed an issue when creating a consumer group with `offsetTypeLatest`

Fixed creating schemas with the same ID

Fixed incorrect operation of filters on the connector plugin selection page

Fixed an issue with hiding the error stack trace panel after a successful restart of a task

Fixed an issue that occurred when setting `predicates.hasDateValue.type` and `transforms.dateToString.type` parameters for newly created Kafka connectors

Misc/Internal

Minimum ADCM version — 2.5.0

2.4.0

2.4.0.b1

Date: 17.03.2025

New features

Implemented the AuthZ plugin (for Ranger)

Implemented the Schema Registry page for viewing and managing schemas

Support for synchronizing users with an LDAP directory

Support for setting up Tiered Storage when creating a topic on the Topics page

Bug fixes

After the ADS Control SSL interface, all Kafka topic enable metadata (size, number of messages, etc.) was displayed, but the messages were not loaded

Incorrect behavior after deleting the last topic or shifting the consumer group partition

Page size settings combobox was not available in the consumer group offset editing form

Filtering messages by ***Start offset*** did not work correctly

Password field was not displayed when creating the Debezium MSSQL/Postgres connector

Request for consumer groups for a non-existent topic did not fail with a 404 error

2.3.0

2.3.0.b1

Date: 25.10.2024

New features

New tabs have been implemented for the Topics page:

- Messages, which allows you to read messages from a topic, as well as write messages to a topic
- Consumer groups, displaying consumer groups connected to the topic

Implemented the Consumer groups overview page for reviewing and managing consumer groups

Implemented support for ReOS

Improvements

Added the ***Advanced docker settings*** group of configuration parameters to control the installation of the Docker package before installing the ADS Control cluster

Bug fixes

Fixed issue with updating the broker state when stopping/starting the Kafka service

Fixed incorrect display of values in the input data for editing topic parameters after changing the configuration via the CLI

Fixed issue with updating connector parameters in the UI after an external change

2.2.0

2.2.0.b1

Date: 27.06.2024

New features

Implemented the General cluster overview page

Implemented the Brokers overview page

Implemented the Topics overview page that allows:

- topics creation, cleaning and deletion operations;
- switching to the topic overview page.

Added new *cluster actions* ***Manage Kerberos*** and ***Manage SSL***

Improvements

Implemented the ability to work with secure (SSL, Kerberos) Kafka clusters

Misc/Internal

After running cluster upgrades, re-enable SSL on the cluster via the ***Manage SSL*** action if it was enabled before the upgrade

2.1.1

2.1.1.b1

Date: 02.08.2023

New features

Implemented support for Astra Linux

Revised user management to use secretmap entries in the ADCM interface

Misc/Internal

Changed product name to ADS Control, made appropriate changes to ADS Control UI

Changed container version to 2.1.1

Updated user agreement

1.0.0

1.0.0.b2

Date: 10.01.2023

Bug fixes

Fixed incorrect display of the cluster status in the ADCM interface after installation

1.0.0.b1

Date: 23.12.2022

New features

The first version of Arenadata Streaming Command Center (ADSCC) has been released with the following functionality:

- creating, editing, and deleting Kafka connectors for exchanging data and creating replicas of Kafka topics;
- support for managing multiple ADS clusters that have Kafka and Kafka Connect services installed.

Version compatibility matrix

Olga Semme

To Table of Contents

ADSC version	ADS min version	ADPS min version	ADCM min version
2.6.0	3.6.2.2	1.2.1	2.8.0
2.5.0	3.6.2.2	1.2.1	2.5.0
2.4.0	3.6.2.2	1.2.1	2.4.0
2.3.0	3.6.2.1	—	2.2.0
2.2.0	2.8.1.1	—	2.0.0
2.1.0	1.8	—	2023.10.10.08