Cloudera Runtime 7.2.14

Using Streams Replication Manager

Date published: 2019-09-13
Date modified: 2022-02-24

https://docs.cloudera.com/

Legal Notice

Contents

SRM Command Line Tools

Overview of the command line tools shipped with SRM.

Important: SRM ships with the srm-control, srm-driver, srm-service, and srm command line tools. Use the srm-control tool to the manage replication of topics and consumer groups. Do not use the srm-driver, srm-service, and srm tools, use Cloudera Manager instead to manage the SRM driver and service.

srm-control

Learn how to use the srm-control command line tool which is used to manage replication of topics and consumer groups.

The srm-control tool enables users to manage replication of topics and consumer groups. The tool has three subcommands topics, groups, and offsets. The topics subcommand is used to control which topics are replicated. The groups subcommand is used to control which consumer groups are replicated. The offsets subcommand is used to export translated offsets for a source->target cluster pair.

The srm-control command line tool is located in /opt/cloudera/parcels/CDH/bin. An alternative is provided for it by default, it is called srm-control. Simply typing srm-control will invoke the tool, you do not need to reference its full path.

Important: Do not use the srm-driver, srm-service, and srm command line tools which are also located in the bin directory. Instead, use Cloudera Manager to manage the SRM driver and service.

If any of the Kafka clusters that SRM connects to use any type of encryption or authentication, you must configure the srm-control tool before use. For more information, see *Configuring srm-control*.

A full list of the available options for the tool can be found in *srm-control Options Reference*.

Related Information

srm-control Options Reference

Configuring srm-control

Cloudera Manager automatically generates a configuration file for the srm-control tool. If both the co-located and external Kafka clusters are unsecured, the default configuration can be used, without making any changes. If however, any of the clusters taking part in the replication process use any type of security, additional configuration is required.

The srm-control tool functions similarly to any Kafka client. It reads data from and writes data to Kafka topics. More specifically, it manipulates Streams Replication Manager's (SRM) internal configuration topic (stored within Kafka) that contains the replication allow and deny lists.

The srm-control tool requires a properties file which specifies information about all clusters taking part in the replication process. That is, the properties file contains the cluster names (aliases), the bootstrap servers, and security related properties of each cluster.

This information, however, is not unique to the tool, as the SRM service (Driver and Service roles) also requires these properties to be set. Therefore, the configuration of the tool can be viewed as a subset of the SRM service's configuration. Because of this and because the SRM service is configured with Cloudera Manager, Cloudera Manager is capable of automatically creating a configuration file for the tool based on the configuration of the SRM service.

The configuration file is located at /etc/streams_replication_manager/conf/srm.properties. By default the tool uses this configuration file.

Important: Do not edit this configuration file directly. Only use it to verify your configuration or to troubleshoot issues. If you want to make changes manually, always create a copy, edit it as needed, and run the tool with the --config option to specify the location of the copy.

The default configuration generated by Cloudera Manager is dynamic. It is updated any time you deploy the client configuration for SRM. For example, if you add a new cluster for replication or change an existing one, the changes you made are automatically added to the default tool configuration once client configuration is deployed.

This automation simplifies the process of configuring the tool. If the Kafka clusters that SRM is connecting to are not secured, then no additional configuration is needed. The default will contain all necessary properties. In cases like this you only need to ensure that the SRM service is correctly configured.

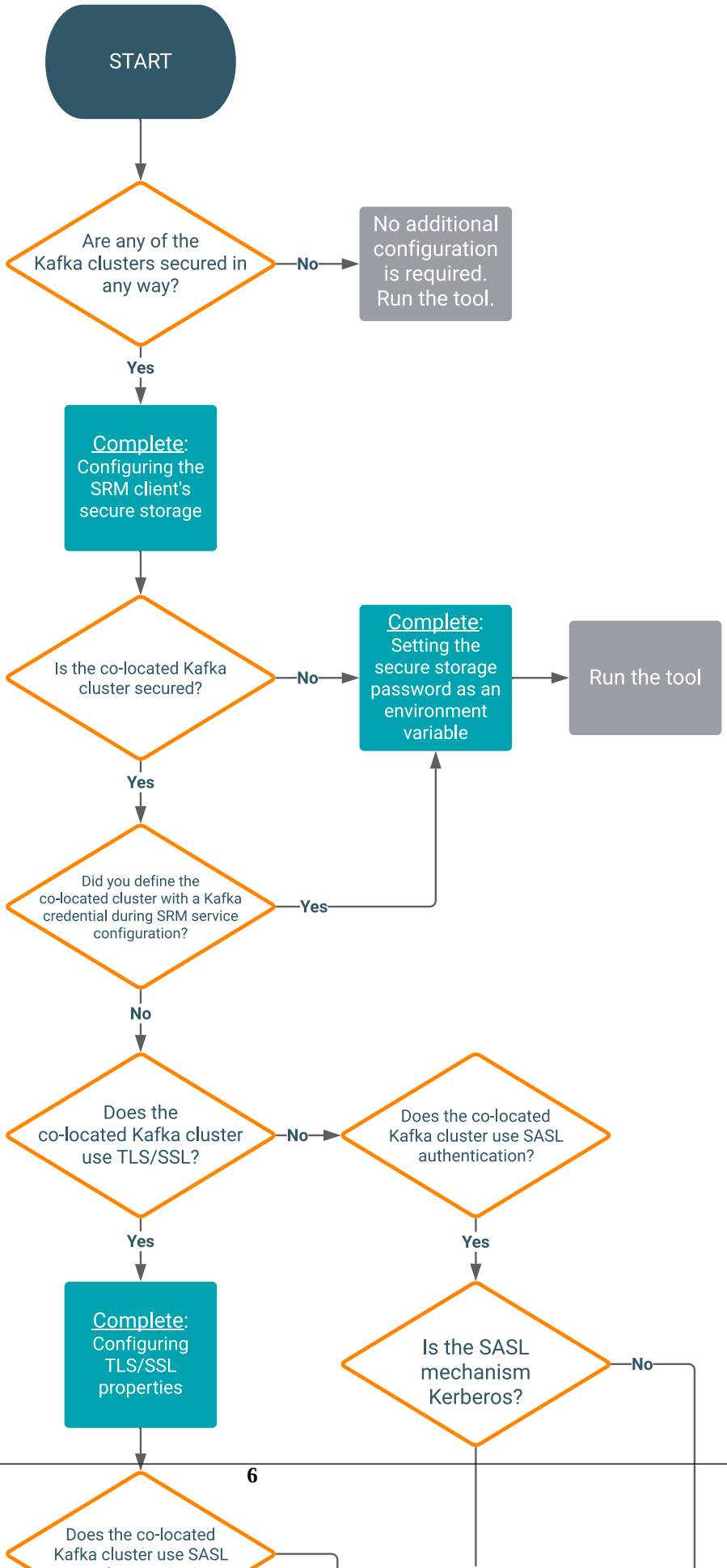
However, if any of the Kafka clusters use any type of encryption or authentication, additional configuration is required. This is because the generated configuration by default does not contain any sensitive TLS/SSL or SASL properties required to access the Kafka clusters. Providing this information to the tool requires additional configuration by the user.

There are five configuration tasks related to setting up sensitive properties for the tool. These are as follows:

- *Configuring the SRM client's secure storage*
- *Configuring TLS/SSL properties*
- *Configuring Kerberos properties*
- *Configuring properties for non-Kerberos authentication mechanisms*
- *Setting the secure storage password as an environment variable*

Which of these tasks you need to complete depends on the security configuration of the clusters that SRM connects to and the method used to define the co-located clusters. Review the following flowchart to see which of the tasks you need to complete and in what order.

Note: If there is no co-located Kafka cluster in your deployment, choose No where the chart asks about the co-located Kafka cluster.


START
Are any of the Kafka clusters secured in any way?
No
No additional configuration is required. Run the tool.
Yes
Complete: Configuring the SRM client's secure storage
Is the co-located Kafka cluster secured?
No
Complete: Setting the secure storage password as an environment variable
Run the tool
Yes
Did you define the co-located cluster with a Kafka credential during SRM service configuration?
Yes
No
Does the co-located Kafka cluster use TLS/SSL?
No
Does the co-located Kafka cluster use SASL authentication?
Yes
Yes
Complete: Configuring TLS/SSL properties
Is the SASL mechanism Kerberos?
No
Does the co-located Kafka cluster use SASL

Configuring the SRM client's secure storage

The SRM client's secure storage is an intermediary keystore used to store sensitive security properties that are required to access the Kafka clusters that SRM connects to. If any of these clusters are secured, you need to set up and configure the secure storage, otherwise the srm-control tool will not function.

About this task

Based on the SRM service's configuration, Cloudera Manager automatically generates a default configuration file that is used by the srm-control tool. The default configuration however does not contain any sensitive data (keystores, truststores, passwords) required to access the clusters. Instead, sensitive details are added to SRM's secure storage, which is an intermediary keystore used to store sensitive data. Specifically, this secure storage will contain all the sensitive information required to access the Kafka clusters that SRM connects to and replicates.

The SRM client's secure storage is not automatically created or populated with all the necessary information. As a result, before you can start using the tool, you must set up and configure the secure storage. This is done by configuring various configuration properties of the SRM service in Cloudera Manager.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the SRM service is complete:
  - The Kafka clusters that SRM connects to are defined and are added to the SRM service's configuration. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Click Gateway in the Filters pane.
4. Find and configure the following properties:

   SRM Client's Secure Storage Type

   The keystore type of the secure storage. Must be a valid Java keystore type. Cloudera recommends that you use the default, PKCS12.

   SRM Client's Secure Storage Password

   The password used to access the secure storage. Take note of the password you configure. You need to provide it in your CLI session before running the tool.

   Environment Variable Holding SRM Client's Secure Storage Password

   The name of the environment variable that stores the secure storage password. Take note of the name that you configure. You need to set it in your CLI session before running the tool.
5. Click Save Changes.
6. Re-deploy client configuration.

What to do next

If your co-located cluster is secured, you must manually configure the sensitive properties required to access the co-located cluster. Continue with one of the following tasks depending on the security configuration of the co-located cluster.

- *Configuring TLS/SSL properties*
- *Configuring Kerberos properties*
- *Configuring properties for non-Kerberos authentication mechanisms.*

If the co-located cluster is not secured, continue with:

- *Setting the secure storage password as an environment variable*

Related Information

Configuring TLS/SSL properties
Configuring Kerberos properties
Configuring properties for non-Kerberos authentication mechanisms
Setting the secure storage password as an environment variable

Configuring TLS/SSL properties

The SRM client's secure storage is automatically populated with the sensitive properties needed by the srm-control tool to access clusters defined with Kafka credentials. However, if the co-located cluster uses TLS/SS and was defined through a service dependency, the TLS/SSL properties of the co-located cluster must be configured and added to the secure storage manually.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the SRM service is complete:
  - The Kafka clusters that SRM connects to are defined and are added to the SRM service's configuration. This includes both external and co-located clusters.
  - Replications are configured.
- If TLS/SSL encryption is enabled on the co-located Kafka cluster:
  - Generate or acquire a truststore containing the certificate of the Certificate Authority that issued the co-located Kafka cluster's certificate.
  - Note down the password of the truststore. You will need to provide it during configuration.
- If both TLS/SSL encryption and authentication are enabled on the co-located Kafka cluster:
  - Generate or acquire a key and truststore which contain all necessary keys and certificates.
  - Note down the locations and passwords for the key and truststores. You will need to provide these during configuration.
  - If the Certificate Authority that issued the tool's certificate is different from the Certificate Authority of the co-located Kafka cluster's certificate, ensure the tool's Certificate Authority certificate is added to the truststore of the co-located Kafka cluster. Otherwise, the co-located Kafka cluster will not trust the certificates used by the tool and a trusted connection will not be established.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Click Gateway in the Filters pane.

4. Find and configure the following properties:

   If the co-located cluster uses SSL/TLS encryption only, and no authentication is required, you do not need to configure the keystore related properties from the following list.

   JKS Keystore file Location for the SRM client's TLS/SSL server

   The path to the TLS/SSL keystore file containing the client certificate and private key. Ensure that this file is available on all SRM hosts.

   SRM Client's TLS/SSL Server JKS Keystore File Password

   The password used to access the keystore file specified in the JKS Keystore file Location for the SRM client's TLS/SSL server property.

   SRM Client's TLS/SSL Server JKS Keystore Key Password

   The password used to access the key stored in the keystore file specified in the JKS Keystore file Location for the SRM client's TLS/SSL server property.

   Gateway TLS/SSL Trust Store File

   The path to the TLS/SSL truststore file containing the server (co-located cluster's) certificate and public key. Ensure that this file is available on all SRM hosts.

   Gateway TLS/SSL Truststore Password

   The password used to access the truststore file specified in the Gateway TLS/SSL Trust Store File property.

   Note: The Gateway TLS/SSL Trust Store File and Gateway TLS/SSL Truststore Password properties are automatically filled if you use Auto-TLS.

5. Click Save Changes.
6. Re-deploy client configuration.

What to do next

- If your co-located cluster uses Kerberos authentication, continue with *Configuring Kerberos properties*.
- If your co-located cluster uses a SASL authentication mechanism different from Kerberos, continue with *Configuring properties for non-Kerberos authentication mechanisms*.
- If the co-located cluster does not use SASL authentication continue with *Setting the secure storage password as an environment variable*.

Related Information

Configuring Kerberos properties

Configuring properties for non-Kerberos authentication mechanisms

Setting the secure storage password as an environment variable

Configuring Kerberos properties

The SRM client's secure storage is automatically populated with the sensitive properties needed by the srm-control tool to access clusters defined with Kafka credentials. However, if the co-located cluster uses Kerberos and was defined with a service dependency, its Kerberos properties must be configured and added to the secure storage manually.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the SRM service is complete:
  - The Kafka clusters that SRM connects to are defined and are added to the SRM service's configuration. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
2. Go to Configuration.
3. Click Gateway in the Filters pane.
4. Find and configure the following properties:

   SRM Client's Kerberos Principal Name

   The kerberos principal that the tool uses for authentication when connecting to the co-located Kafka cluster.

   SRM Client's Kerberos Keytab Location

   The path to the Kerberos keytab file that the tool uses for authentication when connecting to the co-located Kafka cluster.
5. Click Save Changes.
6. Re-deploy client configuration.

What to do next

Continue with *Setting the secure storage password as an environment variable*.

Related Information

Setting the secure storage password as an environment variable

Configuring properties for non-Kerberos authentication mechanisms

If your co-located Kafka cluster uses an authentication mechanism different from Kerberos, and you defined the co-located cluster through a service dependency, you need to copy and edit the srm-control tool's default configuration file generated by Cloudera Manager so that it includes the SASL properties of the co-located cluster.

About this task

The co-located Kafka cluster's properties for SASL mechanisms other than Kerberos cannot be added to the secure storage using Cloudera Manager. Instead, if your co-located Kafka cluster uses an authentication mechanism different from Kerberos, and you defined the co-located cluster through a service dependency, the SASL properties must be directly added to the configuration file that the srm-control tool uses.

This is done by creating a copy of the default configuration file that Cloudera Manager generates and editing the copy so that it contains the necessary SASL properties.

Important: After you complete the following steps, you must run the srm-control tool with --config option to specify the custom configuration file that you created. Otherwise, the tool uses the default configuration and fails when it tries to establish a connection with the co-located cluster.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the SRM service is complete:
  - The Kafka clusters that SRM connects to are defined and are added to the SRM service’s configuration. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. SSH into one of the SRM hosts in your cluster.

2. Create a copy of the default configuration file.
   For example:

```
cp /etc/streams_replication_manager/conf/srm.properties /tmp
```

   Note: The copy of the configuration file that you create contains sensitive information, Cloudera recommends that you protect the copy by setting appropriate system access rights.

   Important: Do not directly edit the default configuration found at /etc/streams_replication_manager/conf/srm.properties. Always create a copy when making changes.

3. Open the copy of the configuration file with an editor.
4. Add the necessary SASL properties to the file.
   The exact properties and values you add to the file depends on the type of SASL authentication used by the co-located Kafka cluster. The following example is for a cluster that uses PLAIN authentication:

```
[***ALIAS***].sasl.mechanism=PLAIN
[***ALIAS***].sasl.jaas.config=${custom_jaas_config}
```

   Replace *[***ALIAS***]* with the co-located cluster's alias. Additionally, Cloudera does not recommend storing the JAAS configuration directly in the file. Instead, store the JAAS configuration in an environment variable and specify the environment variable as the value of the [***ALIAS***].sasl.jaas.config property. This practice is recommended for any other sensitive data that you add to the configuration.

   In a case like this, the environment variable you use to store the sensitive data must be set in your CLI session before running the tool. For example:

```
export custom_jaas_config  = “org.apache.kafka.common.security.plain.PlainLoginModule
          required username="[***USER NAME***]" password="[***PASSWORD***]";”
```

5. Save the file.

What to do next

Continue with *Setting the secure storage password as an environment variable*.

Related Information

Setting the secure storage password as an environment variable

Setting the secure storage password as an environment variable

If any of the clusters that Streams Replication Manager (SRM) connects to are secured, and you have set up and configured the SRM client's secure storage, you must set the secure storage password as an environment variable in your CLI session before running the srm-control tool.

About this task

The SRM Client's secure storage is password protected. In order for the srm-control tool to access the secure storage, it must know the password protecting the secure storage. This password is specified by the user in Cloudera Manager, however it cannot be automatically passed to the tool. This is because the tool is not an entity that is managed by Cloudera Manager. As a result, if any of the Kafka clusters in your replication scenario are secured, and you have set up and configured the SRM client's secure storage, you must configure this password for the srm-control tool. This is done by setting the secure storage password as an environment variable in your CLI session before you run the tool.

Before you begin

- Ensure that you have reviewed the information available in Configuring srm-control and understand that the following step list is only one part of the full configuration workflow. Depending on your scenario, completing other configuration tasks might be required.
- Ensure that setup and configuration of the SRM service is complete:
  - The Kafka clusters that SRM connects to are defined and are added to the SRM service's configuration. This includes both external and co-located clusters.
  - Replications are configured.

Procedure

1. SSH into one of the SRM hosts in your cluster.
2. Set the secure storage password as an environment variable.

```
export [***SECURE STORAGE ENV VAR***]="[***SECURE STORAGE PASSWORD***]"
```

Replace *[***SECURE STORAGE ENV VAR***]* with the name of the environment variable you specified in Environment Variable Holding SRM Client's Secure Storage Password. Replace *[***SRM SECURE STORAGE PASSWORD***]* with the password you specified in SRM Client's Secure Storage Password. For example:

```
export SECURESTOREPASS="mypassword
```

What to do next

Run the srm-control tool. How you run the tool depends on how it was configured:

- If you did not create a copy of the default configuration file you can run the tool without the --config option.

  For example:

```
srm-control topics --source [***SOURCE_CLUSTER***] --targe
t [***TARGET_CLUSTER***] --add [***TOPIC1***],[***TOPIC2***]
```

- If a custom configuration file was created, you must run the tool using the --config option.

  For example:

```
srm-control topics --config [***PATH TO CUSTOM CONFIGURATION FILE***]
 --source [***SOURCE_CLUSTER***] --target [***TARGET_CLUSTER***] --a
dd [***TOPIC1***],[***TOPIC2***]
```

For more information and command examples, see the documentation on the topics, groups, and offsets subcommands.

Related Information

Topics and Groups Subcommand

Offsets Subcommand

Topics and Groups Subcommand

Learn how to use the topics and groups subcommand of the srm-control command line tool.

Important: If any of the Kafka clusters that SRM connects to use any type of encryption or authentication, you must configure the srm-control tool before use. For more information, see Configuring srm-control.

The topics and groups subcommands are used to manipulate the topic or group allowlist (whitelist) and denylist (blacklist). Both subcommands support the same set of command options.

If you are modifying allow and denylists which target newly created topics or consumer groups, changes made with the srm-control tool may not be instantaneous A topic or group needs to be discovered by SRM before it can be added to or removed from allow or denylists. New topic and group discovery happens every 10 minutes by default. As a result, you may need to wait up to 10 minutes until you can see the changes made.

Add topics or groups to an allowlist:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd [GROUP1],[GROUP2]
```

Remove topics or groups from an allowlist:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove [GROUP1],[GROUP2]
```

Add topics or groups to a denylist (blacklist):

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd-blacklist [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --a
dd-blacklist [GROUP1],[GROUP2]
```

Remove topics or groups from a denylist:

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove-blacklist [TOPIC1],[TOPIC2]
```

```
srm-control groups --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --r
emove-blacklist [GROUP1],[GROUP2]
```

In addition to adding or removing items, you can also use the tool to look at the contents of a deny or allowlist.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --l
ist
```

Using Regular Expressions

Specifying topics or groups is also possible with regular expressions. This is done by specifying a regular expression instead of topic names in the --add, add-blackilist, or remove-blacklist options.

For example, assume that you want to replicate all topics from the source cluster to the target cluster. In a case like this, you must add all topics to the allowlist. This can be done with a single srm-control command using the .* regular expression.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --add
 ".*"
```

You can also use regular expressions to specify a set of topics. For example, assume that you have many topics in your Kafka cluster, but only want to add the following topics to the allowlist.

```
testTopic1
testTopic2
testTopic3
```

In a case like this, you can use the testTopic.* or ^test.* regular expressions to add the topics.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --add
 "testTopic.*"
```

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --add
 "^test.*"
```

Note: The regular expression must match the whole topic name. For example, testTopic1 can be matched using test.*, testTopic.*, and .*test.*, but cannot be matched with test, ^test, and so on.

Client Override Options

The topics and groups subcommands support a number of client override options. Client override options allow users to temporarily specify or override configuration properties used for replication. These options also enable users to issue srm-control commands even if the SRM's configuration file is not available on the host that the command is being issued from. While it is possible to specify a range of properties with the client override options, and they can prove to be a powerful tool in certain scenarios, Cloudera recommends that you use Cloudera Manager to manage client configuration options.

The following client override options are available:

- --bootstrap-servers: Specifies the bootstraps servers.
- --producer-props: Specifies producer configuration properties.
- --consumer-props: Specifies consumer configuration properties.
- --props: Specifies client configuration properties.

Note:

Client override options always take precedence over the configuration set in Cloudera Manager. Additionally, the --producer-props and --consumer-props options take precedence over the --props option.

A simple example of using client override options is when you want to change the bootstrap server. This can be done in two ways.

You can specify the bootstrap server with the --bootstrap-servers option.

```
srm-control --bootstrap-servers localhost:9092 topics --sour
ce [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --list
```

Alternatively, you can also use the --props option together with the bootstrap.servers Kafka property to define the bootstrap server.

```
srm-control --props bootstrap.servers=localhost:9092 topics --so
urce [SOURCE_CLUSTER] --list
```

Related Information

New Topic and Consumer Group Discovery

Offsets Subcommand

Learn how to use the offsets subcommand of the srm-client command line tool.

Important: If any of the Kafka clusters that SRM connects to use any type of encryption or authentication, you must configure the srm-control tool before use. For more information, see Configuring srm-control.

SRM automatically translates consumer group offsets between clusters. The offset mappings are created by SRM, but are not applied to the consumer groups of the target cluster directly. Consumers can be migrated from one cluster to another without losing any progress by using the offsets subcommand on the target cluster to export the translated offsets of the source cluster. For example:

```
srm-control offsets --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --g
roup [GROUP1] --export > out.csv
```

Exported offsets can then be applied to the consumer groups in the target cluster with the kafka-consumer-groups tool. Alternatively, this process can be automated by enabling automatic group offset synchronization. If this feature is enabled, the translated offsets are automatically exported from the source cluster and applied to the target cluster.

For detailed steps on cluster migration, see *Migrating Consumer Groups Between Clusters*. For more information regarding automatic group offset synchronization, see *Configuring automatic group offset synchronization*.

Client Override Options

The offset subcommand supports client override options. Client override options allow users to temporarily specify or override configuration properties. These options also enable users to issue srm-control commands even if the SRM's configuration file is not available on the host that the command is being issued from. While it is possible to specify a range of properties with the client override options, and they can prove to be a powerful tool in certain scenarios, Cloudera recommends that you use Cloudera Manager to manage client configuration options.

The following client override options are available:

- --bootstrap-servers: Specifies the bootstraps servers.
- --props: Specifies client configuration properties.

Note:

Client override options always take precedence over the configuration set in Cloudera Manager.

A simple example of using client override options is when you want to change the bootstrap server. This can be done in two ways.

You can specify the bootstrap server with the --bootstrap-servers option.

```
srm-control --bootstrap-servers localhost:9092 offsets --sourc
e [SOURCE_CLUSTER] --group [GROUP] --export > out.csv
```

Alternatively, you can use the --props option together with the bootstrap.servers Kafka property to define the bootstrap server.

```
srm-control --props bootstrap.servers=localhost:9092 offsets --sourc
e [SOURCE_CLUSTER] --group [GROUP] --export > out.csv
```

Related Information

How to Set up Failover and Failback

Migrating Consumer Groups Between Clusters

Configuring automatic group offset synchronization

Monitoring Replication with Streams Messaging Manager

Learn about monitoring SRM replication with Streams Messaging Manager.

Users have the ability to connect SRM with Streams Messaging Manager (SMM) and monitor replications through the SMM UI.This is achieved with the Kafka Streams application and the REST API that come bundled with SRM. The Kafka Streams application calculates and aggregates replication metrics, the REST API exposes these metrics. SMM uses the REST API to display aggregated metrics to the end users, enabling monitoring as a result. Monitoring replication flows in SMM is available starting with version 2.0.0.

For more information regarding the requirements and setup of SRM with SMM, see Monitoring Kafka Cluster Replication using SMM in the SMM guide.

Related Information

Monitoring Cluster Replications Overview

Replicating Data

A step by step guide on how to start replicating data between Kafka clusters with SRM.

About this task

Installing and starting SRM on your cluster does not automatically start data replication. In order to start the replication process, you need to update the allowlists with the srm-control tool.

Before you begin

In Cloudera Manager, verify that the SRM driver role is started and is in good health.

Verify that SRM is configured correctly. Make sure that connection information for each Kafka cluster is added as well as at least one source->target replication is specified and enabled.

Procedure

1. Update the topics allowlist to start data replication.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --
add [TOPIC1],[TOPIC2]
```

Note: If required, instead of listing the topics that you want to add, you can also use regular expressions to add multiple topics with one command.

2. Verify that the topics have been added to the allowlist.

```
srm-control topics --source [SOURCE_CLUSTER] --target [TARGET_CLUSTER] --
list
```

Results

The topics you specify with the --add option are added to the topic allowlist and are replicated to the specified target cluster.

How to Set up Failover and Failback

Learn how to prepare for failover and failback scenarios with Streams Replication Manager (SRM).

If a primary Kafka cluster is temporarily unavailable, you can migrate mission-critical workloads to a backup Kafka cluster (failover). When the primary cluster is restored, you can migrate back (failback). To prepare for this scenario, ensure SRM is configured with bidirectional replication of mission-critical consumer groups and topics. Then in the case of a disaster scenario you can migrate consumer groups between clusters.

Related Information

Offsets Subcommand

Configure SRM for Failover and Failback

Learn how to configure Streams Replication Manager (SRM) for failover and failback.

About this task

To prepare for a failover or failback scenario you must set up SRM with bidirectional replication. Additionally, you must ensure that all mission critical topics and consumer groups are added to the allowlist on both the primary and backup clusters. Optionally, you can also choose to enable automatic group offset synchronization, which can simplify the steps you need to take when migrating consumer groups.

Before you begin

- Ensure that the SRM service is configured and all clusters are taking part in the replication process are defined and added to SRM's configuration. For more information, see Defining and adding clusters for replication.
- Ensure that the srm-control tool is configured. For more information, see Configuring srm-control on page 4.
- If you plan on enabling automatic group offset synchronization, ensure that you review Configuring automatic group offset synchronization. Although the basic steps are provided here, the feature has some limitations that you must be aware of.
- The following steps and examples are for a replication scenario with two clusters. These are referred to as primary and secondary.

Procedure

1. Set up bidirectional replication between clusters:
   a) In Cloudera Manager, go to Clusters and select the Streams Replication Manager service.
   b) Go to Configuration
   c) Find the Streams Replication Manager's Replication Configs property.
   d) Click the add button and add new lines for each unique replication you want to add and enable.
   e) Add and enable your replications.

   For example, assume that you have two clusters, primary and secondary. To enable bidirectional replication between these clusters, you must add two unique replications. One that replicates from primary to secondary and another that replicates from secondary to primary.

   ```
   primary->secondary.enabled=true
   secondary->primary.enabled=true
   ```

2. Enable and configure automatic group offset synchronization.

   This can be done by enabling sync.group.offsets.enabled. Optionally, if you want to customize the frequency of offset synchronization, you can also set sync.group.offsets.interval.seconds. Both properties are configured by adding them to Streams Replication Manager's Replication Configs. For example:

   ```
   sync.group.offsets.enabled = true
   sync.group.offsets.interval.seconds = [***TIME IN SECONDS***]
   ```

3. Enter a Reason for change, and then click Save Changes to commit the changes.
4. Restart Streams Replication Manager.

5. Add the required consumer groups and topics to the allowlist on the primary cluster.
   - Add groups:

     ```
     srm-control groups --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --add [GROUP1],[GROUP2]
     ```

   - Add topics:

     ```
     srm-control topics --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --add [TOPIC1],[TOPIC2]
     ```

6. Add the required consumer groups and topics to the allowlist on the secondary cluster.

   Important:

   If remote topics and consumer groups are not added to the allowlist on the secondary cluster, a failback operation will be impossible to carry out.

   - Add groups:

     ```
     srm-control groups --source [SECONDARY_CLUSTER] --targe
     t [PRIMARY_CLUSTER] --add [GROUP1],[GROUP2]
     ```

   - Add topics:

     ```
     srm-control topics --source [SECONDARY_CLUSTER] --target [PRIMARY_CLUSTE
     R] --add [PRIMARY_CLUSTER.TOPIC1],[PRIMARY_CLUSTER.TOPIC2]
     ```

7. Verify that all required topics and consumer groups are added to the allowlist.
   - Verify consumer groups:

     ```
     srm-control groups --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --list
     ```

     ```
     srm-control groups --source [SECONDARY_CLUSTER] --targe
     t [PRIMARY_CLUSTER] --list
     ```

   - Verify topics:

     ```
     srm-control topics --source [PRIMARY_CLUSTER] --targe
     t [SECONDARY_CLUSTER] --list
     ```

     ```
     srm-control topics --source [SECONDARY_CLUSTER] --targe
     t [PRIMARY_CLUSTER] --list
     ```

Results

SRM is set up with bidirectional replication and all mission critical topics and consumer groups are added to the allowlist on both the primary and secondary clusters.

Related Information

Configuration Examples

Migrating Consumer Groups Between Clusters

Learn how to migrate consumers between clusters.

About this task

If a primary Kafka cluster is temporarily unavailable, you can migrate mission-critical workloads to a secondary Kafka cluster (failover). When the primary cluster is restored, you can migrate back (failback). The steps for migrating consumers in a failover or failback scenario are identical. However, depending on the scenario, your source and target clusters will be different. During failover you migrate consumers from primary to secondary, while during failback you migrate consumers from secondary to primary.

Before you begin

- Make sure that the clusters that you are migrating consumers between are set up with bidirectional replication.
- Verify that all mission critical consumer groups and topics, including the ones on the secondary cluster are whitelisted.
- The steps you need to take differ depending on whether automatic group offset synchronization is enabled. Check SRM's configuration to see if the feature is enabled. For more information, see Configuring automatic group offset synchronization

Procedure

1. Migrate consumer group offsets.

   If automatic group offset synchronization is disabled follow the steps provided under Manual migration. If automatic group offset synchronization is enabled, follow the steps under Automatic migration.

   #### For Manual migration

   a. Export the translated consumer group offsets of the source cluster.

   ```
   srm-control offsets --source [SOURCE_CLUSTER] --targe
   t [TARGET_CLUSTER] --group [GROUP1] --export > out.csv
   ```

   b. Reset consumer offsets on the target cluster.

   ```
   kafka-consumer-groups --bootstrap-server [TARGET_BROKER:PORT] --reset
   -offsets --group [GROUP1] --execute --from-file out.csv
   ```

   #### For Automatic migration

   If automatic group offset synchronization is enabled, the translated offsets of the source cluster are already applied on the target cluster. Continue with Step 2 on page 19.

   Important: Although translated offsets are applied, depending on the offset group synchronization frequency configured (sync.group.offsets.interval.seconds and emit.checkpoints.interval.seconds), you might have outdated offsets applied. Cloudera recommends that you follow the steps for manual migration if you are unsure whether the latest offsets were applied or if having the latest offsets is critical for your use case.

2. Start consumers on the target cluster.

Results

Consumers automatically resume processing messages on the target cluster where they left off on the source cluster.

Related Information

Offsets Subcommand