Cloudera Streaming Analytics 1.18.0

Installation & Upgrade

Date published: 2019-12-17
Date modified: 2026-09-16

CLOUDERA

https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Installation 4
Deployment scenarios 4
Cluster service layout with Flink 5
Cluster service layout with Cloudera SQL Stream Builder 6
Installing CSD and Parcel 7
Adding Flink as a Service 9
Setting up your HDFS Home directory 10
Setting the Java executable for the Flink client 11
Configuring Databases for Cloudera SQL Stream Builder 11
Setting up MySQL/MariaDB database for Cloudera SQL Stream Builder 12
Setting up PostgreSQL database for Cloudera SQL Stream Builder 13
Setting up Oracle database for Cloudera SQL Stream Builder 14
Adding Cloudera SQL Stream Builder as a Service 16
Enabling High Availability for Cloudera SQL Stream Builder 18

Upgrade 19
Before upgrading your cluster 19
Stopping Flink applications 19
Stopping SQL stream jobs 20
Exporting SQL projects 21
Upgrading Cloudera Streaming Analytics artifacts and services 22
After upgrading your cluster 25
Repairing the MariaDB Admin Database 25
Updating Flink job dependencies 26
Configuring the Python environment for SQL Stream Builder 26
Resuming Flink applications 27
Importing a project 28
Resuming SQL jobs 29

Migration 30
Migrating Flink service to a different host 31
Migrating SQL jobs 31

Installation

You can review the different deployment scenarios for Cloudera Streaming Analytics. After you have decided by one of the deployment scenarios, you can start the installation process by adding the CSD and Parcel in Cloudera Manager, then adding and configuring Flink and Cloudera SQL Stream Builder services.

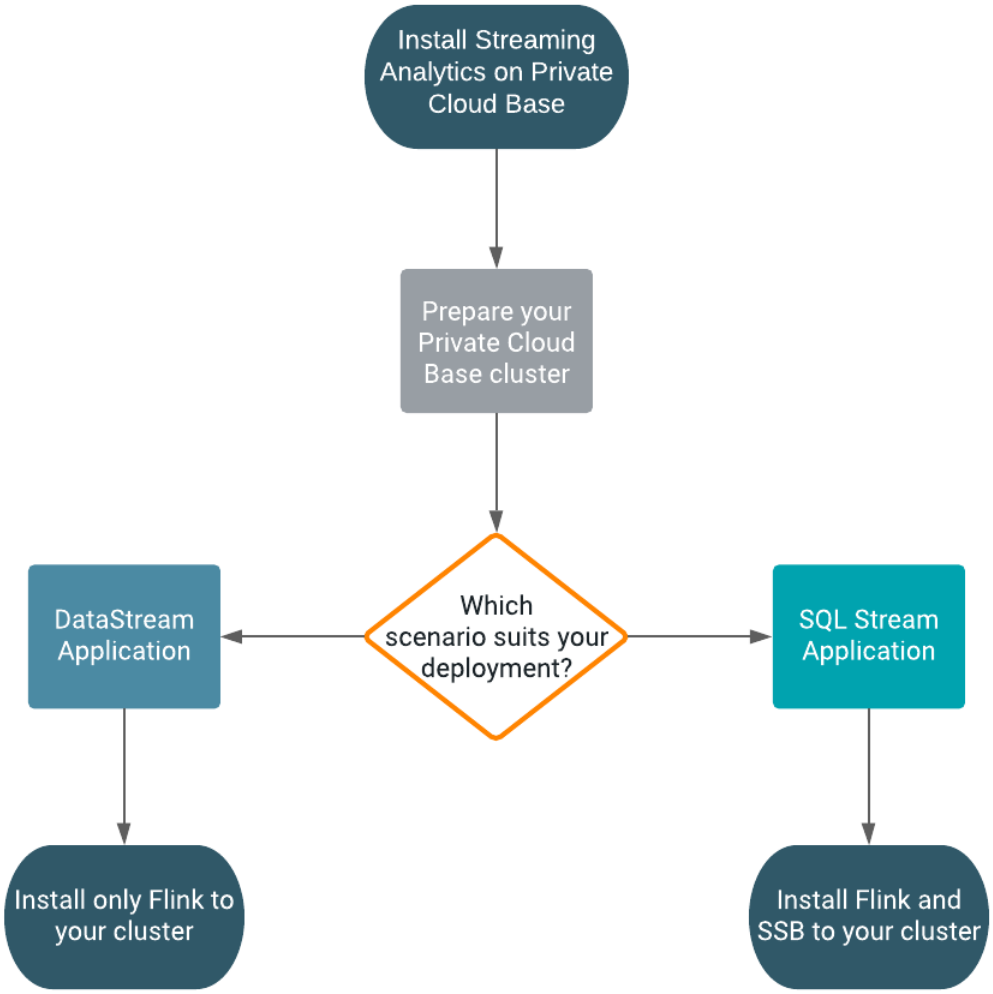
Deployment scenarios

The application you want to build determines the Cloudera Streaming Analytics deployment process on Cloudera Base on premises. You can choose between building either a DataStream application or a SQL streaming application.

You can deploy Cloudera Streaming Analytics on Cloudera Base on premises depending on the application you want to build.

- DataStream application using only Flink. In this case, you need to create a Flink application cluster.
- SQL Streaming application using Flink with Cloudera SQL Stream Builder. In this case, you need to create a Streaming SQL cluster.

You can use the following workflow to have an understanding of the deployment process:



Cluster service layout with Flink

In Cloudera Streaming Analytics, Flink has mandatory dependencies with HDFS, YARN, and Zookeeper. You need to assign the Flink Gateway and HistoryServer roles to the host, based on the mandatory dependencies.

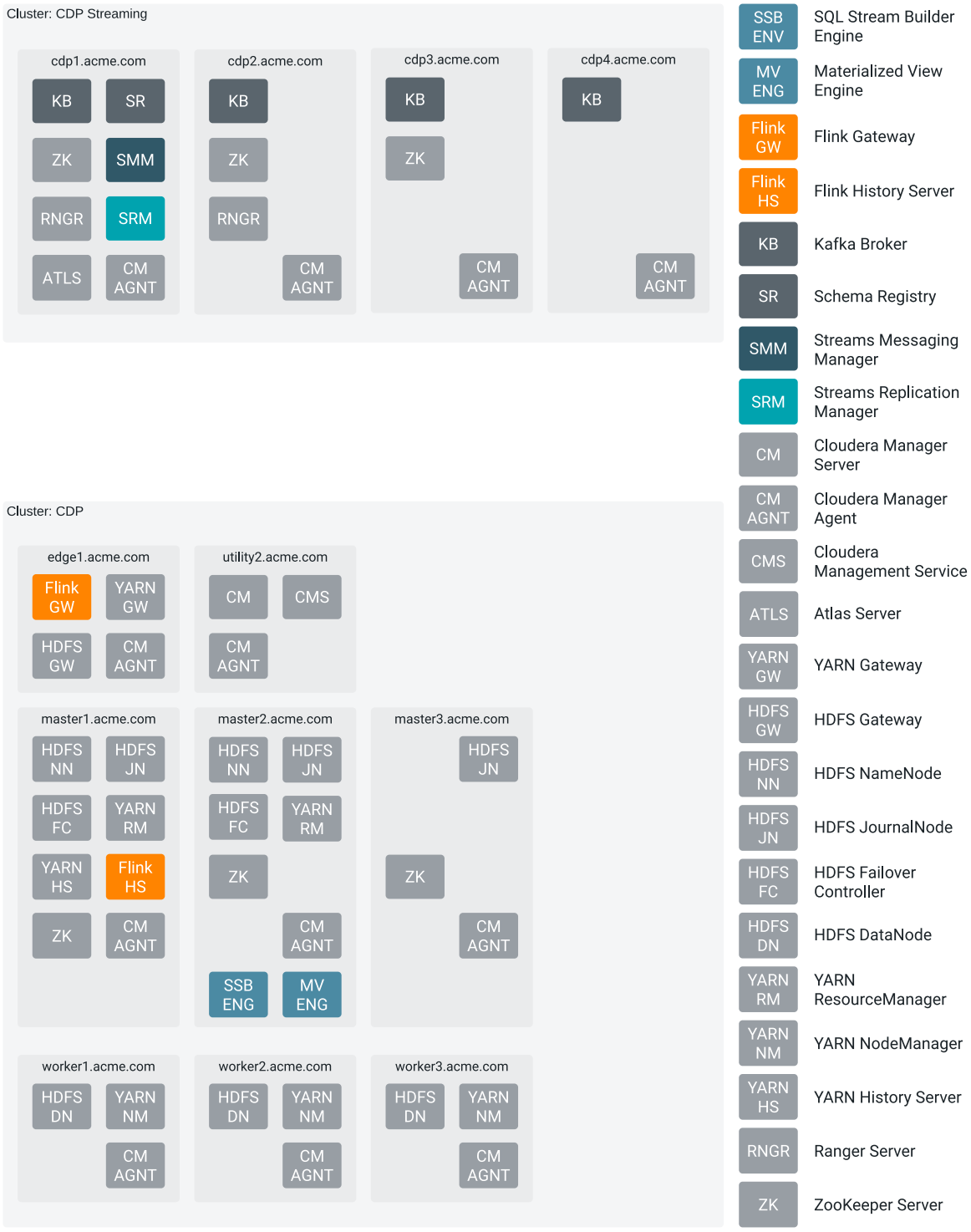
Flink jobs are executed as YARN applications. HDFS is used to store recovery and log data, while ZooKeeper is used for high availability coordination for jobs. In a standard layout, an Apache Kafka cluster is often located close to the YARN cluster executing the Flink cluster.

The Flink Gateway is collocated with YARN and HDFS Gateways. The Flink HistoryServer is collocated with an HDFS role, which can be either an active role or a Gateway. Use the following general service layout when collocating Flink roles and dependencies.



Cluster service layout with Cloudera SQL Stream Builder

In Cloudera Streaming Analytics, Cloudera SQL Stream Builder) has mandatory dependencies with Flink and Kafka. But due to its dependency with Flink, you also need to add YARN, HDFS and Zookeeper as a mandatory service on your cluster. You need to assign the Cloudera SQL Stream Builder roles in the same way as you assign the Flink roles.



Installing CSD and Parcel

For installing Cloudera Streaming Analytics, you need to upload the downloaded Flink and Cloudera SQL Stream Builder Custom Service Descriptor (CSD) files to the default CSD directory, and add the Cloudera Streaming Analytics parcel to your cluster using Cloudera Manager.

Before you begin

- Download the Flink and Cloudera SQL Stream Builder CSD and parcel files.

  For more information about download Flink and Cloudera SQL Stream Builder artifacts, see the Download location section.
- Install Cloudera Base on premises.

  For more information about installing Cloudera Base on premises and Cloudera Manager, see the Cloudera Base on premises documentation.
- If you access Cloudera SQL Stream Builder through Knox, select the Knox Gateway - Async Operations Supported checkbox in the Knox configuration before you install Cloudera SQL Stream Builder so that sampling works correctly.

Procedure

1. Place the CSD files in the /opt/cloudera/csd/ folder (default CSD directory).

   ```
   wget -P /opt/cloudera/csd/ https://USER:PASSWORD@archive.cloudera.com/p/
   csa/1.18.0.0/csd/FLINK-1.20.5-csa1.18.0.0-82229597.jar
   ```

   ```
   wget -P /opt/cloudera/csd/ https://USER:PASSWORD@archive.cloudera.com/p/
   csa/1.18.0.0/csd/SQL_STREAM_BUILDER-1.20.5-csa1.18.0.0-82229597.jar
   ```

   Cloudera Manager automatically detects the CSD files.
2. Change the ownership of the CSD files.

   ```
   chown cloudera-scm:cloudera-scm /opt/cloudera/csd/FLINK-1.20.5-
   csa1.18.0.0-82229597.jar
   ```

   ```
   chown cloudera-scm:cloudera-scm /opt/cloudera/cs
   d/SQL_STREAM_BUILDER-1.20.5-csa1.18.0.0-82229597.jar
   ```
3. Restart Cloudera Manager and CMS services for the changes to take effect.

   ```
   systemctl restart cloudera-scm-server
   ```
4. Log into Cloudera Manager.
5. Select Parcels on the Home > Hosts tab in the main navigation bar.
6. Click on Parcel Repositories & Network Settings tab.
7. Add the new Remote Parcel Repository URL for CSA.

   ```
   https://USER:PASSWORD@archive.cloudera.com/p/csa/1.18.0.0/parcels/
   ```

   Note: Make sure that the Remote Parcel Repository URL uses HTTPS link. To install a different version of the parcel, you can change the URL as needed.
8. Enter your download credentials to HTTP authentication username override for Cloudera Repositories and HTTP authentication password override for Cloudera Repositories.
9. Click Save & Verify Configuration to commit the change.
10. Click Close.
    You are redirected to the Parcels page.
11. Search for Flink, and click Download to download the parcel to the local repository.
12. After the download is completed, click Distribute to distribute the parcel to all clusters.
13. After the parcel is distributed, click Activate to activate the parcel.
14. Click OK when confirmation is required.

For more details, follow the standard procedure from the GUI or the API.

Results
You have added the Flink and Cloudera SQL Stream Builder CSD files, and the parcel to your cluster.

What to do next
Add Flink as a service in Cloudera Manager. After adding Flink as a service, you are able to add Cloudera SQL Stream Builder service to your cluster.

Related Information
Installing Cloudera Private Cloud Base

Add Flink as a Service

Adding Flink as a Service

You need to use the Add Service wizard in Cloudera Manager to include the Flink service on your cluster. When assigning roles, you must install Flink, HDFS, and YARN roles on the same node from where the Flink jobs are submitted.

Before you begin

- Make sure that the Flink CSD file is in the /opt/cloudera/csd folder.
- Make sure that the Cloudera Streaming Analytics parcel is added to Cloudera Manager.
- Check that the following components are installed on your cluster:

Mandatory components	Optional components
YARN	Kafka
HDFS	HBase
Zookeeper	Schema Registry
	Streams Messaging Manager
	Kudu
	Hive
	Atlas

Procedure

1. Open Cloudera Manager.
2. On the Home screen, select the drop-down menu to the right of your cluster.
3. Select Add Service.
4. From the list, select Flink as the type of service, then click Continue.
   The Add Service wizard is displayed.
5. Choose HBase and Hive as Optional dependency if needed for the source and sink solution, then click Continue.
6. Assign the Flink Dashboard role to one of the hosts, and the Flink Gateway role to every host.

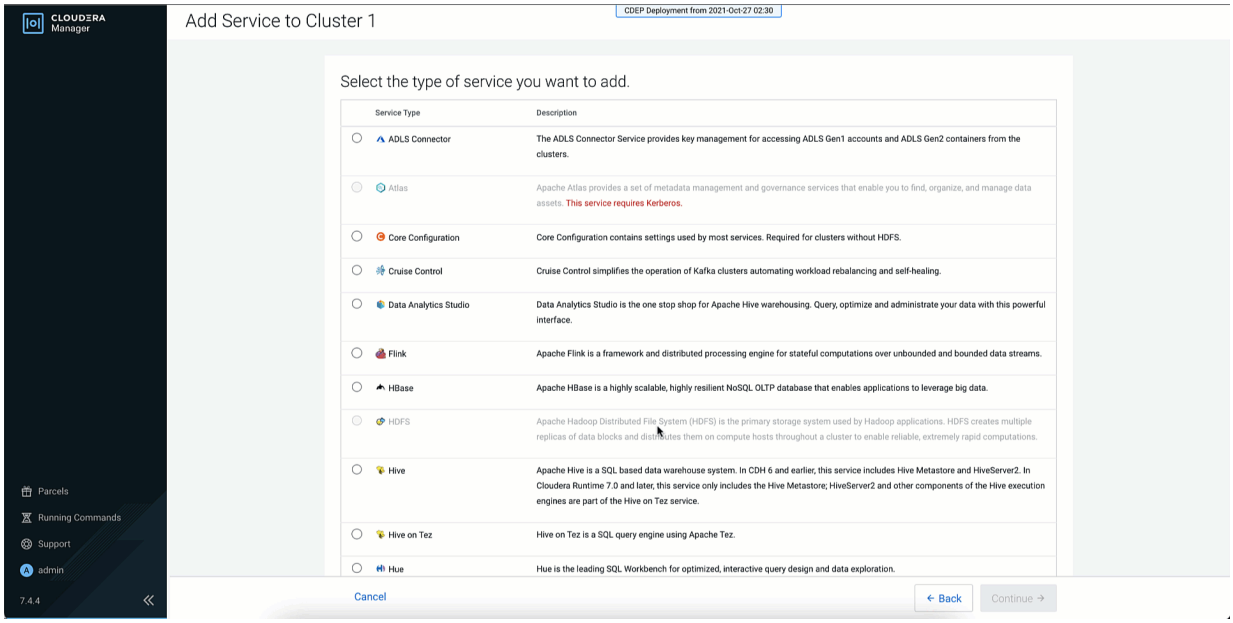


Note:

Make sure that you installed Flink, HDFS, and YARN Gateway roles on the same machine that will be used to submit Flink jobs. The Flink Dashboard role also depends on having HDFS client configurations on the same machine. The HDFS client configurations can either be provided by an HDFS daemon role implicitly or can be deployed by an HDFS Gateway role explicitly. Furthermore, the YARN Gateway role must be added to the same host as the Flink Dashboard role to resolve the YARN Resource Manager URL.

7. Click Continue.
8. Review the changes needed for your service.

   You can leave this page blank as the settings can be configured after the Flink service is added. You can configure the security settings at Flink > Configuration > Security .
9. Click Continue and wait until the first run of the Flink service is completed.
10. Click Continue and then Finish.



Results

You have added Flink as a service in Cloudera Manager.

Related Information

Installing CSD and parcel

Add Cloudera SQL Stream Builder as a Service

Setting up your HDFS Home directory

You need an HDFS Home directory to store temporary logs and data of your application to run a Flink job. You must set up the HDFS Home directory for your user to avoid error when using Flink.

About this task

To run a Flink job, your HDFS Home directory has to exist. If it does not exist, you receive an error message similar to the following:

```
Permission denied: user=$USER_NAME, access=WRITE, inode="/user".
```

Procedure

Create HDFS Home directory. Ask your HDFS administrator to perform the following (or obtain HDFS administrator role).

Options	Command
Kerberos enabled	`kinit hdfs`
	`hdfs dfs -mkdir /user/$USER_NAME`
	`hdfs dfs -chown $USER_NAME:$USER_NAME /user/$USER_NAME`
Kerberos disabled	`HADOOP_USER_NAME=hdfs hdfs dfs -mkdir /user/$USER_NAME`
	`HADOOP_USER_NAME=hdfs hdfs dfs -chown $USER_NAME:$USER_NAME /user/$USER_NAME`

In case of an enterprise environment, you can use Hue to set up the Home directory by enabling automatic synchronization for users. For more information, see the Cloudera Runtime documentation.

Setting the Java executable for the Flink client

To avoid error when using Flink, you must set the Java_home environment manually through the command line for the Flink clients. The configuration in Cloudera Manager only applies to services, and not to clients.

About this task

Cloudera Manager offers a configuration for the JAVA_HOME environment variable under Hosts > All Hosts > Configuration . However, this only applies to services (for example YARN NodeManager or Flink HistoryServer) and does not propagate to clients such as the JVM created locally by the Flink executable. JVM uses the Bigtop utility under /usr/bin/bigtop-detect-javahome to automatically detect the JAVA_HOME.

Procedure

Set a fixed value for JAVA_HOME:

```
> cat /etc/default/bigtop-utils
export JAVA_HOME=/usr/java/default
```



Note:

Cloudera recommends providing the same value set in Cloudera Manager. It is also recommended to set it uniformly on all the nodes to avoid confusion. This is a known issue in the Cloudera Community.

Configuring Databases for Cloudera SQL Stream Builder

Before adding Cloudera SQL Stream Builder as a service to your cluster, you need to manually configure the databases to use. The databases are used to store the metadata information of the Streaming SQL Engine and the Materialized Views.

Cloudera SQL Stream Builder supports the following databases:

Database	Streaming SQL Engine	Materialized View Engine
MySQL/MariaDB	Supported	Not supported
PostgreSQL	Supported	Supported
Oracle DB	Supported	Not supported

For more information about the supported versions of the databases, see the Cloudera Streaming Analytics-specific System Requirements.

Setting up MySQL/MariaDB database for Cloudera SQL Stream Builder

After installing MySQL/MariaDB server, you must rename the JDBC connector, and create a database with credentials for to be able to install the service on your cluster. You also must install the MySQL Python connector to integrate with the Streaming SQL Engine.

Before you begin

You need to install and configure MySQL or MariaDB based on which one you plan to use, before setting up the databases for Cloudera SQL Stream Builder. To install and configure MySQL or MariaDB, you must complete the basic steps mentioned in the Cloudera Base on premises documentation.

- Install and configure MySQL for Cloudera Software
- Install and configure MariaDB for Cloudera Software

Installing the MySQL JDBC connector

1. Download the MySQL JDBC Driver from the MySQL website.
2. Extract the JDBC driver JAR file from the downloaded file with the following command:

```
tar zxvf mysql-connector-java-8.0.27.tar.gz
```

3. Rename the JDBC jar file to mysql-connector-java.jar.

```
mv mysql-connector-java-8.0.27-bin.jar mysql-connector-java.jar
```

4. Copy the MySQL JDBC jar file to your host.

   This host must be the same host where you plan to assign the Streaming SQL Engine service role. The service roles are assigned as a next step when installing Cloudera SQL Stream Builder as a service in Cloudera Manager.

```
scp <LOCATION>/mysql-connector-java.jar root@<YOUR_HOSTNAME>:
```

   You will be prompted to provide your password.
5. Access the host on your cluster.

   This host must be the same host where you have added the JDBC jar file.

```
ssh root@<your_hostname>
```

   You will be prompted to provide your password.
6. Copy the MySQL JDBC jar file to /usr/share/java folder using the following command.

```
sudo mkdir -p /usr/share/java
sudo cp <LOCATION>/mysql-connector-java.jar /usr/share/java
```

7. Check if the MySQL connector is in the folder with ls command.

Creating MySQL/MariaDB database for SSB

1. Access a host on your cluster.

This host must be the same host where you plan to assign the Streaming SQL Engine service role. The service roles are assigned as a next step when installing Cloudera SQL Stream Builder as a service in Cloudera Manager.

```
ssh root@<your_hostname>
```

You will be prompted to provide your password.

2. Log in as the root user to MySQL:

```
mysql -u root -p
Enter password:
```

3. Create databases for the Streaming SQL Engine:

```
CREATE DATABASE ssb_admin DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf
8_general_ci;
```

4. Grant all privileges for the database:

```
GRANT ALL ON ssb_admin.* TO 'ssb_admin'@'%' IDENTIFIED BY '<password>';
```

5. Confirm that you have created the Streaming SQL Engine database:

```
SHOW DATABASES;
```

Setting up PostgreSQL database for Cloudera SQL Stream Builder

After installing PostgreSQL server, you must rename the JDBC connector, and create a database with credentials for Cloudera SQL Stream Builder before installing the service on your cluster. You also must install the PostgreSQL Python connector to integrate with the Streaming SQL Engine.

Before you begin

You need to install and configure PostgreSQL, before setting up the databases for Cloudera SQL Stream Builder. To install and configure PostgreSQL, you must complete the basic steps mentioned in the Cloudera Base on premises documentation.

- Install and configure PostgreSQL for Cloudera Software

Installing the PostgreSQL JDBC connector

1. Download the PostgreSQL JDBC Driver from the PostgreSQL website.
2. Rename the JDBC jar file to postgresql-connector-java.jar.

```
mv postgresql-jdbc.jar postgresql-connector-java.jar
```

3. Copy the PostgreSQL JDBC jar file to your host.

This host must be the same host where you plan to assign the Streaming SQL Engine service role. The service roles are assigned as a next step when installing Cloudera SQL Stream Builder as a service in Cloudera Manager.

```
scp <LOCATION>/postgresql-connector-java.jar root@<YOUR_HOSTNAME>:
```

You will be prompted to provide your password.

4. Access the host on your cluster.

This host must be the same host where you have added the JDBC jar file.

```
ssh root@<your_hostname>
```

You will be prompted to provide your password.

5. Copy the PostgreSQL JDBC jar file to /usr/share/java folder using the following command:

```
sudo mkdir -p /usr/share/java
sudo cp <LOCATION>/postgresql-connector-java.jar /usr/share/java
```

6. Check if the PostgreSQL connector is in the folder with ls command.

Creating PostgreSQL database for Cloudera SQL Stream Builder

1. Access a host on your cluster.

   This host must be the same host where you plan to assign the Materialized Views Engine and Streaming SQL Engine service role. The service roles are assigned as a next step when installing Cloudera SQL Stream Builder as a service in Cloudera Manager.

```
ssh root@<your_hostname>
```

   You will be prompted to provide your password.
2. Connect to PostgreSQL:

```
sudo -u postgres psql
```

3. Create a database for the Cloudera SQL Stream Builder metadata:

```
CREATE ROLE ssb_admin LOGIN PASSWORD '<password>';
CREATE DATABASE ssb_admin OWNER ssb_admin ENCODING 'UTF8';
```

4. Set the search_path for the Cloudera SQL Stream Builder metadata role to public.

   SSB schema migrations reference the public schema explicitly. If the role default search path is not public, installation or upgrade can fail.

```
ALTER ROLE ssb_admin SET search_path TO public;
```

5. Create a database for the Materialized View Engine:

```
CREATE ROLE ssb_mve LOGIN PASSWORD '<password>';
CREATE DATABASE ssb_mve OWNER ssb_mve ENCODING 'UTF8';
```

6. Confirm that you have created the Streaming SQL Engine and Materialized View Engine database using the \l command.

Setting up Oracle database for Cloudera SQL Stream Builder

After installing Oracle database server, you must rename the JDBC connector, and create a database with credentials for Cloudera SQL Stream Builder before installing the service on your cluster. You also must install the Oracle Python connector to integrate with the Streaming SQL Engine.

Before you begin

You need to install and configure Oracle database, before setting up the databases for Cloudera SQL Stream Builder. To install and configure Oracle DB, you must complete the basic steps mentioned in the Cloudera Base on premises documentation.

- Install and configure Oracle Database for Cloudera Software

Installing the Oracle JDBC connector

1. Download the Oracle JDBC Driver from the Oracle website.
2. Rename the JDBC jar file to oracle-connector-java.jar.

```
mv ojdbc8-19.3.0.0.jar oracle-connector-java.jar
```

3. Copy the Oracle JDBC jar file to your host.

This host must be the same host where you plan to assign the Streaming SQL Engine service role. The service roles are assigned as a next step when installing Cloudera SQL Stream Builder as a service in Cloudera Manager.

```
scp <LOCATION>/oracle-connector-java.jar root@<YOUR_HOSTNAME>:
```

You will be prompted to provide your password.

4. Access the host on your cluster.

   This host must be the same host where you have added the JDBC jar file.

   ```
   ssh root@<your_hostname>
   ```

   You will be prompted to provide your password.
5. Copy the Oracle JDBC jar file to /usr/share/java folder using the following command:

   ```
   sudo mkdir -p /usr/share/java
   sudo cp <LOCATION>/oracle-connector-java.jar /usr/share/java
   sudo chmod 644 /usr/share/java/oracle-connector-java.jar
   ```
6. Check if the Oracle connector is in the folder with ls command.

Creating Oracle database for Cloudera SQL Stream Builder

1. Access a host on your cluster.

   This host must be the same host where you plan to assign the Streaming SQL Engine service role. The service roles are assigned as a next step when installing Cloudera SQL Stream Builder as a service in Cloudera Manager.

   ```
   ssh root@<your_hostname>
   ```

   You will be prompted to provide your password.
2. Log in to the Oracle client:

   ```
   sqlplus system@localhost
   Enter password: ******
   ```
3. Create a user and schema for Streaming SQL Engine:

   ```
   create user ssb_admin identified by <password> default tablespace ssb_ad
   min;
   grant CREATE SESSION to ssb_admin;
   grant CREATE TABLE to ssb_admin;
   grant CREATE SEQUENCE to ssb_admin;
   grant EXECUTE on sys.dbms_lob to ssb_admin;
   ```
4. Grant a quota on the ssb_admin tablespace where the tables will be created:

   ```
   ALTER USER ssb_admin quota 100m on ssb_admin;
   ```

   You can also create unlimited space with the following command:

   ```
   ALTER USER ssb_admin quota unlimited on ssb_admin;
   ```
5. Confirm that you have created the Streaming SQL Engine and Materialized View Engine database using the \l command.

Adding Cloudera SQL Stream Builder as a Service

You need to use the Add Service wizard in Cloudera Manager to have the Cloudera SQL Stream Builder service on your cluster.

Before you begin

- Make sure that the Cloudera SQL Stream Builder CSD file is in the /opt/cloudera/csd folder.
- Make sure that the Cloudera Streaming Analytics parcel is added to Cloudera Manager.
- Make sure that you have installed and configured the Cloudera SQL Stream Builder databases correctly, and installed the required Java drivers as well.
- Check that the following components are installed on your cluster:

Mandatory components	Optional components
Flink	Schema Registry
Kafka	Kudu
	Hive
	KNOX

Procedure

1. Open Cloudera Manager.
2. If KNOX is installed in the cluster, enable the async support flag.

   Note: If your cluster is secure but doesn't contain the KNOX service, you must install it first. For more information, see Apache Knox Gateway Overview.

   a) In **Cloudera Manager**, navigate to KNOX serviceConfiguration.
   b) Add the following property to the conf/gateway-site.xml_role_safety_valve configuration:

   ```
   <property>
       <name>gateway.servlet.async.supported</name>
       <value>true</value>
   </property>
   ```

   c) Restart the KNOX service.
3. On the Home screen, select the drop-down menu to the right of your cluster.
4. Select Add Service.
5. From the list, select Cloudera SQL Stream Builder as the type of service, then click Continue.
   The Add Service wizard launches.
6. Assign the SQL Stream Engine and Materialized View Engine service roles to hosts, then click Continue.

   You need to assign the service roles based on where you have created the databases, and where you have assigned the Flink and Kafka roles.
7. Connect the Cloudera SQL Stream Builder service to a database.

   Important: You must install and configure MySQL/MariaDB, PostgreSQL or Oracle database before adding Cloudera SQL Stream Builder as a service. In case you did not set up any database for your cluster, see the *Configuring databases for Cloudera SQL Stream Builder* documentation.

   a) Select MySQL, PostgreSQL, or Oracle as the type of database.

      Important: If you plan to use Materialized Views in Cloudera SQL Stream Builder, you must install a PostgreSQL database.

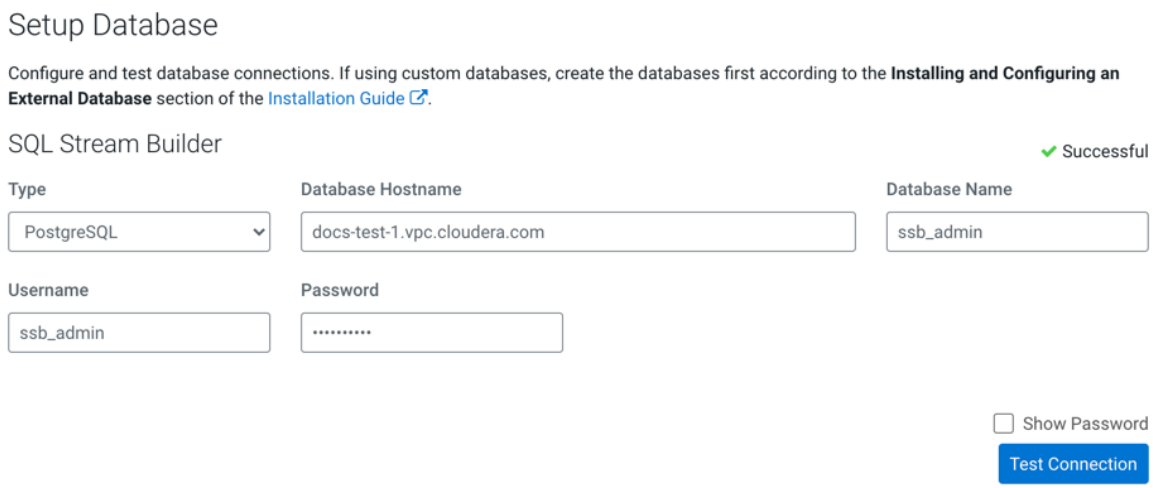
   b) Choose the host on which you want to add the database.

You must add the databases to the same host where you have assigned the service roles.

c) Provide a name to the database.

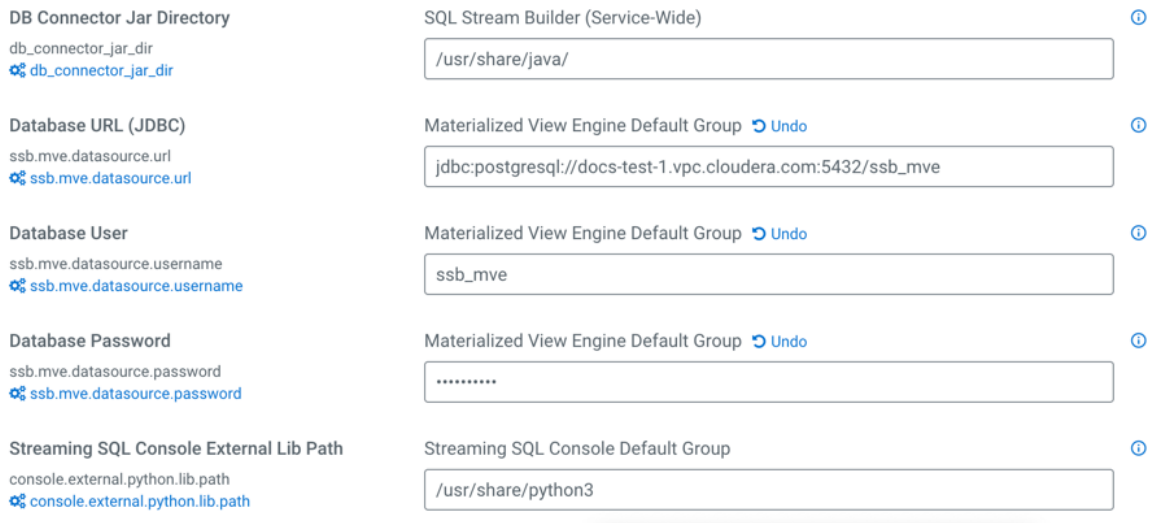
d) Provide the user and password of the created database.

e) Click Test connection.



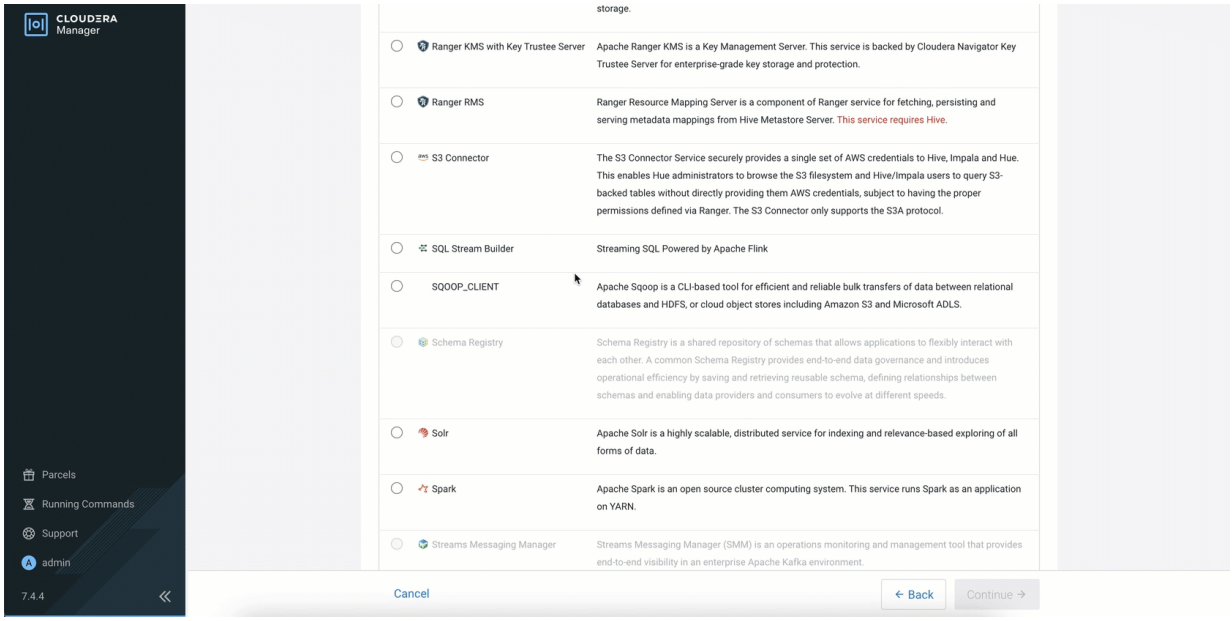
8. Review the changes needed for your service.

   In case you are using MySQL or Oracle for the Streaming SQL Engine, and PostgreSQL for Materialized Views, you will be prompted to provide information about the database for Materialized View Engine. Provide the database hostname with the default port, your user and password in the Materialized View Engine fields.



Note: You can configure the security properties for Cloudera SQL Stream Builder in this step, or after adding the service in the Configuration page. For more information about configuring security, see the *Manage security* documentation for Cloudera SQL Stream Builder.

9. Click Continue and wait until the first run of the Cloudera SQL Stream Builder service is completed.
10. Click Continue and then Finish.



Related Information

Installing CSD and parcel
Add Flink as a Service
Install and configure MySQL for Cloudera Software
Install and configure MariaDB for Cloudera Software
Install and configure PostgreSQL for Cloudera Software
Install and configure Oracle database for Cloudera Software
Configuring MySQL/MariaDB for Cloudera SQL Stream Builder
Configuring PostgreSQL for Cloudera SQL Stream Builder
Configuring Oracle database for Cloudera SQL Stream Builder

Enabling High Availability for Cloudera SQL Stream Builder

High Availability for Cloudera SQL Stream Builder can be enabled using load-balancing through KNOX. To be able to use load-balancing for Cloudera SQL Stream Builder after the service is installed, you need to have a KNOX service installed in the SSB cluster.

Using KNOX for load-balancing Cloudera SQL Stream Builder

From Cloudera Streaming Analytics 1.14.0, Cloudera SQL Stream Builder uses KNOX as a load-balancer for multiple instances.

Important: KNOX is the only load-balancing solution available for Cloudera SQL Stream Builder. If your cluster does not support KNOX, load-balancing will not be available.

KNOX handles load-balancing automatically when multiple instances are added. For more information on KNOX, see *Apache Knox Gateway Overview*.

Note: Accessing Cloudera SQL Stream Builder instances through Cloudera Manager is not load-balanced. Load-balancing is only available when accessing Cloudera SQL Stream Builder through the KNOX gateway.

Upgrade

You can upgrade Flink and Cloudera SQL Stream Builder from an older version to the latest by changing the Cloudera Streaming Analytics CSD and parcel in Cloudera Manager, and restarting the services. After the upgrade is completed, you can migrate your Flink and SQL jobs to the new clusters.

Before upgrading your cluster

Before upgrading your cluster, you must stop your Flink and SQL Stream jobs with or without a savepoint to be able to restart them on your upgraded cluster. You can also export your SQL projects to a repository to create a backup of the different versions.

Stopping Flink applications

You must stop your Flink applications before updating the artifacts. You can use the stop command to create a savepoint of your application. You can also use the cancel command to stop your Flink application without creating a savepoint.

About this task

You can use savepoints to resume the application state after the upgrade.

Procedure

1. Connect to your host where you run your Flink jobs using ssh.

```
ssh root@[***FLINK HOSTNAME***]
Password:[***PASSWORD***]
```

2. Determine the related Flink job IDs.

```
flink list
```

3. Stop your Flink applications.

   You can choose from the following options based on the application state and savepoint path:

   - Stop your applications with a savepoint to store the application state.

     ```
     flink stop [***FLINK JOB ID***]
     ```

     The command returns an HDFS path, which is the automatically created savepoint that stores the application state.
   - Stop your Flink jobs with a savepoint with providing the savepoint path for HDFS.

     ```
     flink stop --savepointPath hdfs:///tmp/savepoints [***FLINK JOB ID***]
     ```

     Use this method if you want to specify the savepoint path.
   - Cancel your applications without creating a savepoint.

     ```
     flink cancel [***FLINK JOB ID***]
     ```

     Use this method if you do not need to restore the application state after the upgrade.

4. Take note of the savepoint path in the output of the command as you must provide the path when resuming Flink stateful applications.

Stopping SQL stream jobs

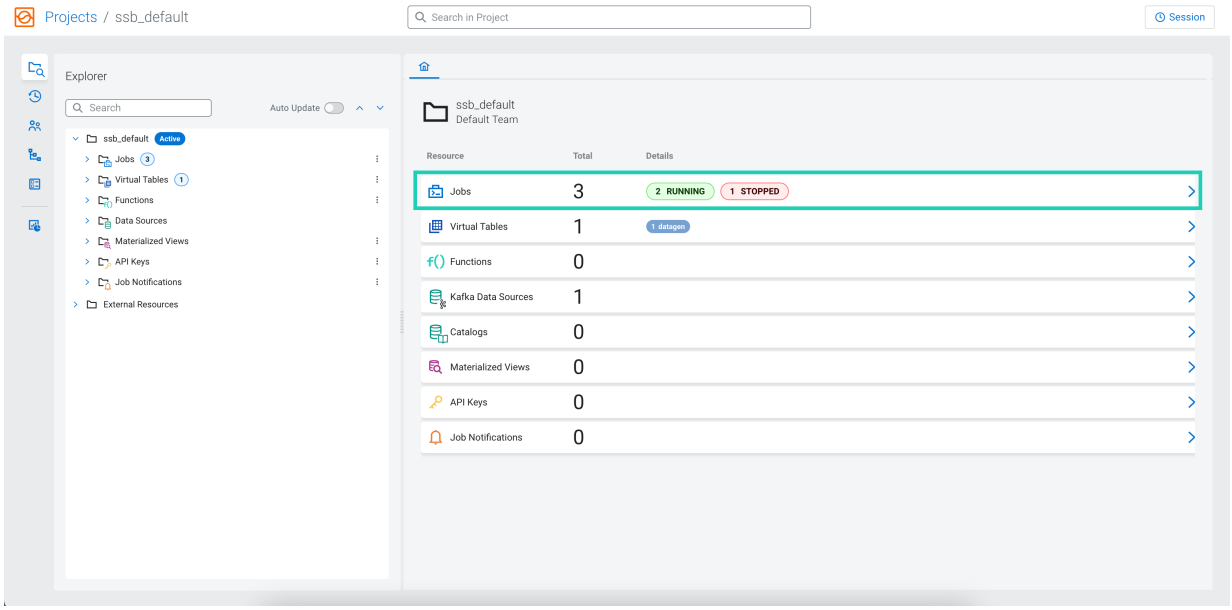
Before upgrading your cluster, you need to stop your running jobs on the Jobs page of Streaming SQL Console, and stop all Flink and Cloudera SQL Stream Builder related YARN sessions. Otherwise Flink will not be able to submit applications to YARN after upgrade.

Procedure

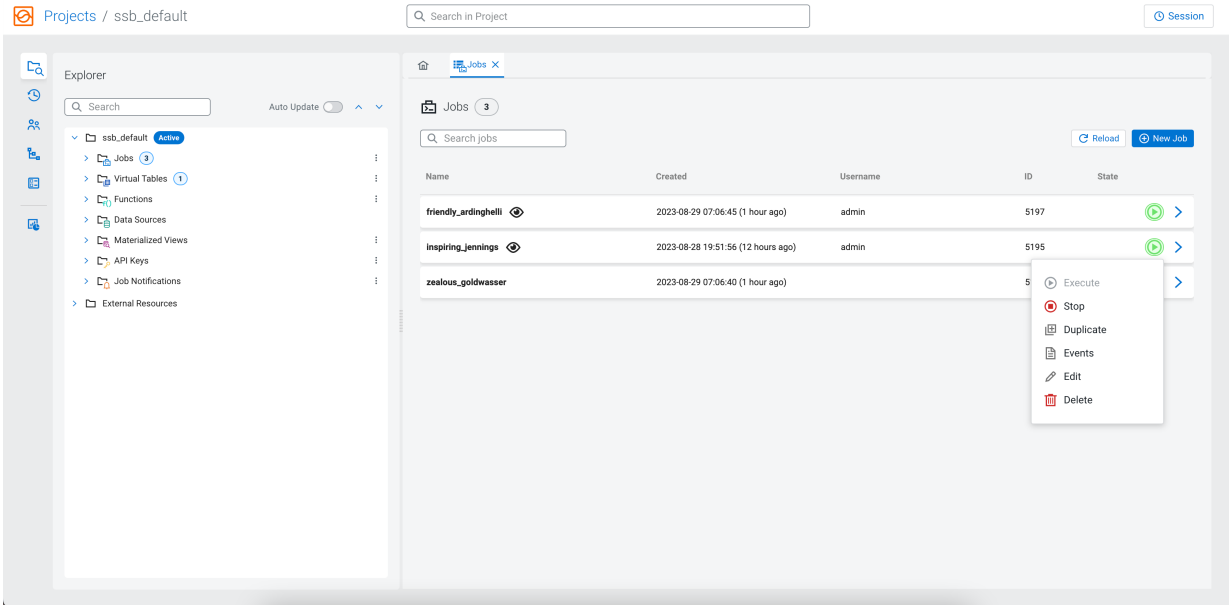
1. Navigate to the Streaming SQL Console.
   a) Go to your cluster in Cloudera Manager.
   b) Select SQL Stream Builder from the list of services.
   c) Click SQLStreamBuilder Console .

   The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Select Jobs on the homepage of your project.



4. Click on  at the jobs you want to stop.



You can further filter down the results, by directly searching for the job name in the Search field.

5. Click Stop.

   After you stop the SQL job on Streaming SQL Console, go back to Cloudera Manager to stop the YARN session.
6. Navigate to the YARN Resource Manager to stop the Cloudera SQL Stream Builder YARN job.
   a) Select YARN from the list of Services.
   b) Select YARN.
   c) Select Applications.

      The running Cloudera SQL Stream Builder applications are displayed.
7. Select the application you need to stop.
8. Click Settings.
9. Click Kill application.

Exporting SQL projects

You can use source control to create a backup from your Cloudera SQL Stream Builder projects. After the upgrade is successful, you can import the projects on Streaming SQL Console.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Go to your cluster in Cloudera Manager.
   b) Select SQL Stream Builder from the list of services.
   c) Click SQLStreamBuilder Console .

      The **Streaming SQL Console** opens in a new window.
2. Open a project from the **Projects** page of Streaming SQL Console.
   a) Select an already existing project from the list by clicking the Open button or Switch button.
   b) Create a new project by clicking the New Project button.
   c) Import a project by clicking the Import button.

   You are redirected to the **Explorer** view of the project.
3. Click Source Control.

   You can configure the source control settings with the following steps, if the source control is not configured yet:
   a) Add a remote Github repository URL to the **Clone URL** field.

You can use the following example URLs with or without authentication:

- To clone without authentication: https://github.com/cloudera/ssb-examples.git
- To clone and push with HTTP basic authentication (username/password): https://github.com/cloudera/ssb-examples.git

  In case of using basic authentication, the personal access token needs to be provided as password. For more information about creating personal access tokens, see the Github documentation.
- To clone and push with SSH authentication (SSH private key): git@github.com:cloudera/ssb-examples.git

  In case of using SSH authentication, you need to generate the private key using the following command:

  ```
  ssh-keygen -t ecdsa -b 256 -m pem -C "NAME@EXAMPLE.COM"
  ```

  As not every SSH private key is supported, you can use the following command to convert your key to the acceptable format:

  ```
  ssh-keygen -m pem -p -f [***PATH_TO_PRIVATE_KEY***]
  ```
- To clone and push a local git repository on the filesystem of SSB host: file:///usr/src/ssb-examples/

b) Provide the branch name or tag of the Github repository to the **Branch** field. This branch is checked out when pulling or pushing to the Github repository.

4. Provide a commit message.
5. Click Push.
6. Review that the commit appears in the repository you pushed the change.

Upgrading Cloudera Streaming Analytics artifacts and services

You need to upgrade the Cloudera Streaming Analytics artifacts to the newer version by manually deleting the old CSD file from the repository, and adding the latest CSD file to the same location.

Before you begin

- Upgrade Cloudera Base on premises to the 7.3.2 version. For more information see the Upgrading Cloudera Base on premises to a higher version documentation.
- Review the Support Matrix page to ensure that you use the correct Operating System (OS), Java Development Kit (JDK) and database version that is compatible with the new Cloudera Streaming Analytics version.

Procedure

1. Ensure that the Flink and Cloudera SQL Stream Builder services are stopped on your cluster.
   a) Go to your cluster in Cloudera Manager.
   b) Select Cloudera SQL Stream Builder from the list of services.
   c) Click Action Stop .
   d) Go back to the homepage of Cloudera Manager.
   e) Select Flink from the list of services.
   f) Click Action Stop .
2. Stop and remove any Load Balancer instances from the cluster.
   a) Go to your cluster in **Cloudera Manager**.
   b) Select Cloudera SQL Stream Builder from the list of services.
   c) Click Instances.
   d) If there's an instance with Load Balancer role, select it and click Actions for SelectedDelete.

   Cloudera Streaming Analytics 1.14.0 and higher uses the KNOX service for load balancing, without needing a load balancer instance. For more information see Enabling High Availability for Cloudera SQL Stream Builder.

Note: If your cluster doesn't have the KNOX service installed, you need to install and configure it before load balancing can be used for Cloudera SQL Stream Builder. For more information on KNOX, refer to Apache Knox Gateway overview.

3. Download the new CSD files from the Download location.

   In case you are upgrading to the latest Cloudera Streaming Analytics version that includes Cloudera SQL Stream Builder, you also must download the Cloudera SQL Stream Builder CSD, and add it to the /opt/cloudera/csd folder with the new Flink CSD. For more information, see the Installing parcel and CSD documentation.

4. Delete the old CSD version from /opt/cloudera/csd.

   a) Access the host where the CSD is stored.

   ```
   ssh root@[***HOSTNAME***]
   cd /opt/cloudera/csd
   rm FLINK-1.20.1-csa1.18.0.0-77091851.jar
   rm SQL_STREAM_BUILDER-1.20.1-csa1.18.0.0-77091851.jar
   ```

5. Copy the new artifacts to the /opt/cloudera/csd folder.

   Warning: The upgrade process fails if you only add the new parcel file in Cloudera Manager. You must add the Flink and Cloudera SQL Stream Builder CSD to the /opt/cloudera/csd folder as well.

   ```
   scp [***DOWNLOAD LOCATION***]/FLINK-1.20.5-csa1.18.0.0-82229597.jar roo
   t@[***HOSTNAME***]/opt/cloudera/csd:
   scp [***DOWNLOAD LOCATION***]/SQL_STREAM_BUILDER-1.20.5-
   csa1.18.0.0-82229597.jar root@[***HOSTNAME***]/opt/cloudera/csd:
   ```

6. Ensure that you maintain the appropriate file ownership and access the attributes if needed.

   ```
   chown cloudera-scm:cloudera-scm /opt/cloudera/csd/FLINK-1.20.5-
   csa1.18.0.0-82229597.jar
   chown cloudera-scm:cloudera-scm /opt/cloudera/c
   sd/SQL_STREAM_BUILDER-1.20.5-csa1.18.0.0-82229597.jar
   ```

7. Restart the Cloudera Manager Server with the following command:

   - RHEL 7 Compatible, SLES 12, Ubuntu:

     ```
     systemctl restart cloudera-scm-server
     ```

   - RHEL 6 Compatible:

     ```
     service cloudera-scm-server restart
     ```

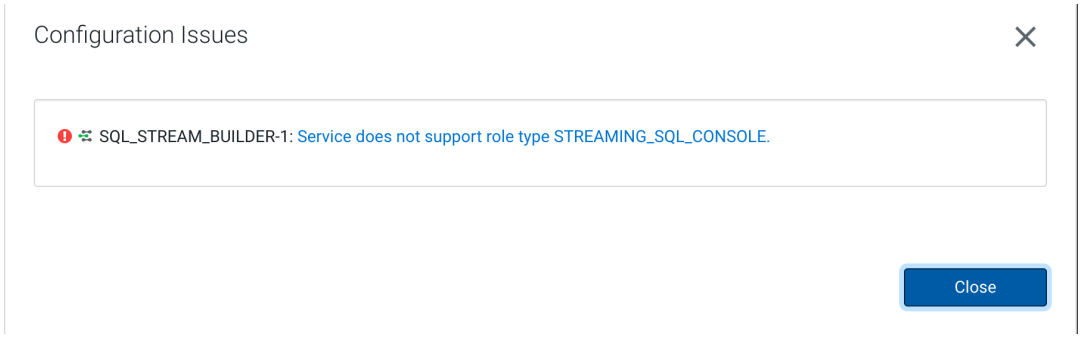
8. Restart the Cloudera Management Services.

   When upgrading from versions lower than Cloudera Streaming Analytics 1.7.0:

   Due to the backend changes of Cloudera SQL Stream Builder, after adding the new CSD, the Cloudera SQL Stream Builder service will not work which is indicated in the list of services in the following way:

   The following error message is displayed when reviewing the configuration issue:



This means that you need to manually remove the Streaming SQL Console role, and restart the Streaming SQL Engine role after activating the new parcel.

9. Select Parcels on the Home Hosts tab in the Cloudera Manager main navigation bar.
10. Click on Parcel Repositories & Network Settings tab.
11. Add the new Remote Parcel Repository URL for CSA.

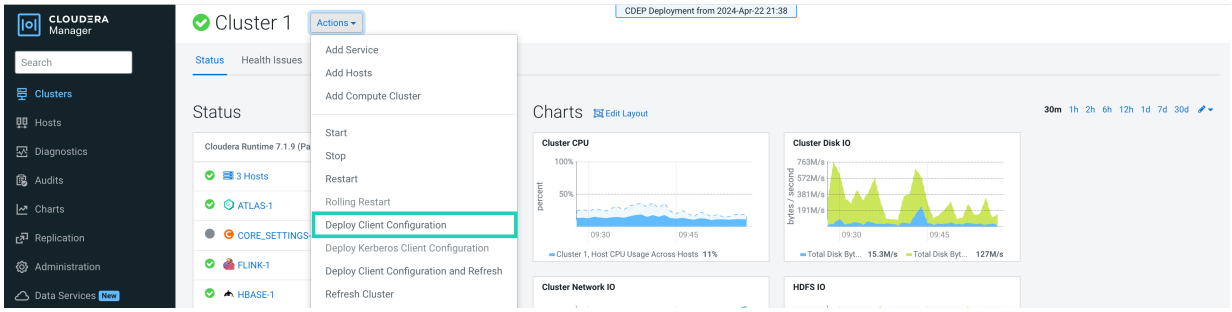
```
https://[***USER:PASSWORD***]@archive.cloudera.com/p/csa/1.18.0.0/parcels/
```

Note: Make sure that the Remote Parcel Repository URL uses HTTPS link. To install a different version of the parcel, you can change the URL as needed.

12. Click Save & Verify Configuration to commit the change.
13. Click Close.
    You are redirected to the Parcels page.
14. Deactivate the old version of the parcel on the **Parcels** page.
15. Download, distribute, and activate the new version of the parcel.
16. Select the Activate Only option when prompted in the wizard.

    For more information about the general process of Service Management, see the Cloudera Manager documentation.
17. Navigate back to **Clusters**.
18. Select Actions Deploy Client Config .

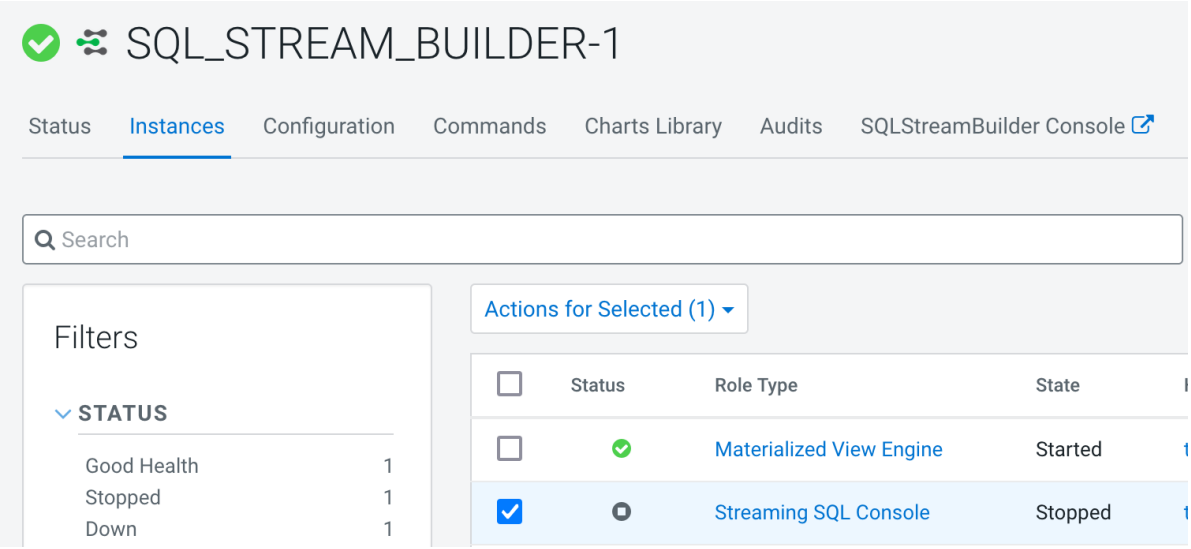
    Deploying client configuration is needed as the Flink service becomes stale after activation.



19. **When upgrading from versions lower than Cloudera Streaming Analytics 1.7.0:**

    Delete the Streaming SQL Console role from your cluster.

    a) Stop the Cloudera SQL Stream Builder service using Actions Stop next to the Cloudera SQL Stream Builder service name.
    b) Select the Cloudera SQL Stream Builder service.
    c) Select the Streaming SQL Console role.
    d) Click Actions for Selected Delete .



20. Update the Admin Database for the Cloudera SQL Stream Builder service.

a) Click Actions Update Admin Database next to the Cloudera SQL Stream Builder service name.

21. Start the Flink service.

a) Click Actions Start next to the Flink service name.

22. Start the SQL Stream Builder service.

a) Click Actions Start next to the Cloudera SQL Stream Builder service name.

23. Refresh your keytab in Streaming SQL Console.

a) Select Cloudera SQL Stream Builder from the list of services.

b) Click SQLStreamBuilder Console.

The **Streaming SQL Console** opens in a new window.

c) Select your username on the left main menu of Streaming SQL Console.

d) Click Manage keytab.

e) Provide your password again.

f) Click Refresh Keytab.

After upgrading your cluster

After upgrading your cluster, you need to update the Flink job dependencies, and restart your Flink and SQL stream jobs on the new clusters.

Repairing the MariaDB Admin Database

If you are updating MariaDB to version 11.x from any lower version, you must repair the SQL Stream Builder admin database schema before starting the service.

About this task

When you update MariaDB to version 11.x from any lower version, the Flyway database migration scripts diverge from the MySQL scripts.

Because some SQL scripts have changed to maintain compatibility with MariaDB 11.x, you must run the Repair Admin Database script in Cloudera Manager before starting the new Cloudera SQL Stream Builder instance. This repairs the schema history checksums.

Procedure

In Cloudera Manager, run the Repair Admin Database action for Cloudera SQL Stream Builder before you start the Cloudera SQL Stream Builder service after the database upgrade.

What to do next

Additionally, for MariaDB 11.4.2 and higher versions, you must ensure that the mariadb-connector-java.jar file is added to the Cloudera SQL Stream Builder classpath.

Updating Flink job dependencies

When you migrate your Flink jobs to a cluster that has a new supported version of Flink, the applications need to use a new version of the artifacts provided by the Flink deployment in your cluster. To avoid incompatibilities between the packaged artifacts of your application and the artifacts provided by the Flink cluster, ensure that the POM file of the application is updated to match the Flink version of the new Cloudera Base on premises cluster.

Procedure

1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select the Streaming Analytics cluster from the list of Data Hub clusters.
3. Access the latest Flink version of your cluster.
   a) Go to your cluster in Cloudera Manager.
   b) Select Hosts Parcels from the main menu of Cloudera Manager.
   c) Search for **Flink** under **Parcel Name**, and review the Flink version.

      Name: FLINK

      Version: 1.13.2-csadh1.5.0.0-cdh7.2.12.0-35-17794544
4. Update the <flink.version> property of your POM file using the Flink and Cloudera Streaming Analytics version of your Cloudera Data Hub cluster.

   You need to copy and paste only the prefix before the 'cdh' version number: 1.13.2-csadh1.5.0.0.

```
<properties>
        ...
        <flink.version>1.13.2-csadh1.5.0.0</flink.version>
        ...
    </properties>
    ...
    <dependencies>
        ...
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-java</artifactId>
            <version>${flink.version}</version>
        </dependency>
        ...
    <dependencies>
```

5. Rebuild your JAR file.

```
mvn clean package
```

Configuring the Python environment for SQL Stream Builder

Starting in Cloudera Streaming Analytics 1.16.0, the Python User Defined Functions (UDF) functionality is enabled by default. Cloudera Manager and the related Cloudera Streaming Analytics scripts automatically configure the Python environment and install the necessary Python modules during the upgrade process.

About this task

However, manual configuration is required in the following scenario:

- **Custom Python Environments:** If you have a custom-configured Python environment, Cloudera Manager does not overwrite your modified configuration parameters during the upgrade. You must manually update the Python paths to point to the new environment.

Manual configuration steps

If your environment meets the criteria above, perform the following steps to ensure the Python runtime is correctly configured for Cloudera SQL Stream Builder.

Procedure

1. Reset Python parameters in SSB

To ensure Cloudera SQL Stream Builder uses the correct Python version (Python 3.11), you must reset the executable paths in Cloudera Manager.

1. Log in to the Cloudera Manager Admin Console.
2. Navigate to ClustersSQL Stream Builder.
3. Click the Configuration tab.
4. Search for the following parameters and reset them to the default value of /usr/bin/python3.11:
   - Python Client Executable (ssb.python.client.executable)
   - Python Executable (ssb.python.executable)
5. Click Save Changes.
6. In Cloudera Manager, navigate to the Flink service.
7. Click the Instances tab.
8. Click Add Role Instances.
9. Select the Flink Gateway role.
10. Assign the Flink Gateway role to the Worker hostgroup (or all hosts that will run PyFlink jobs).
11. Click Continue and follow the prompts to add the roles.

2. Deploy Client Configuration

After updating the roles, you must deploy the client configuration to trigger the installation of the PyFlink Python module.

12. Navigate back to the Flink service in Cloudera Manager.
13. Click the Actions button and select Deploy Client Configuration.
14. Confirm the action and wait for the command to complete.

    This ensures the pyflink module is installed across all relevant hosts.
15. Restart the SQL Stream Builder service if prompted or to apply the changes.

Resuming Flink applications

After updating the Cloudera Streaming Analytics artifacts, you can resume your application by the Flink run command. In case you have stopped your application by creating a savepoint, you must add the savepoint location to the command.

Resuming Flink applications without state

1. Connect to your host where you run your Flink jobs using ssh.

```
ssh root@[***FLINK HOSTNAME***]
Password:[***PASSWORD***]
```

2. Submit your Flink application that was stopped before the upgrade.

```
flink run [***RUN ARGUMENTS***] \
[***JAR FILE***] [***APP ARGUMENTS***]
```

Resuming stateful Flink applications

1. Connect to your host where you run your Flink jobs using ssh.

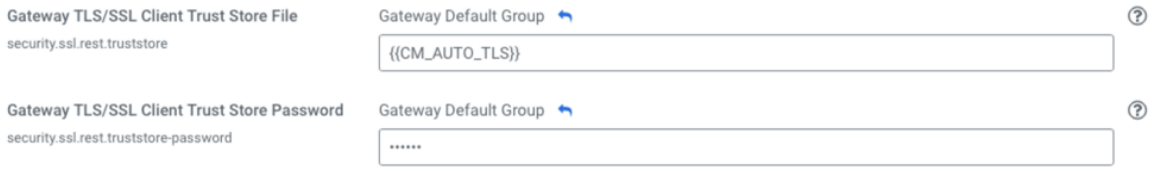
```
ssh root@[***FLINK HOSTNAME***]
Password:[***PASSWORD***]
```

2. Submit your Flink application that was stopped before the upgrade.

```
flink run [***RUN ARGUMENTS***] \
-fromSavepoint hdfs:///tmp/savepoints/[***SAVEPOINT NAME***] \
[***JAR FILE***] [***APP ARGUMENTS***]
```

Important:

Additional TLS related properties were added to the Gateway role. If you previously enabled TLS on your cluster, you can define Gateway TLS/SSL Client Trust Store File and Gateway TLS/SSL Client Trust Store Password configurations. You can set {{CM_AUTO_TLS}} as the property values when using Auto TLS in Cloudera Manager.



You can verify your settings by running the flink list -yD security.ssl.rest.enabled=true command.

Related Information

Running a Flink job

Importing a project

You can use a Git repository to import a Cloudera SQL Stream Builder project that was previously created and exported using source control.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Go to your cluster in Cloudera Manager.
   b) Select SQL Stream Builder from the list of services.
   c) Click SQLStreamBuilder Console .

   The **Streaming SQL Console** opens in a new window.
2. Click Import Project on the **Projects** page of Streaming SQL Console.

   The **Import Project** window appears.
3. Optional: Provide a prefix that overrides the Materialized View table names.

   Providing a prefix to the Materialized View tables names that overrides the default one is an advanced option. The tables are prefix by default with the ID of the projects to avoid collision between projects. This configuration allows setting a custom prefix to the Materialized View tables created in the project.
4. Provide the **Source Settings** of the project.
   a) Add a remote Github repository URL to the **Clone URL** field.

You can use the following example URLs with or without authentication:

- To clone without authentication: https://github.com/cloudera/ssb-examples.git
- To clone and push with HTTP basic authentication (username/password): https://github.com/cloudera/ssb-examples.git

  In case of using basic authentication, the personal access token needs to be provided as password. For more information about creating personal access tokens, see the Github documentation.
- To clone and push with SSH authentication (SSH private key): git@github.com:cloudera/ssb-examples.git

  In case of using SSH authentication, you need to generate the private key using the following command:

  ```
  ssh-keygen -t ecdsa -b 256 -m pem -C "NAME@EXAMPLE.COM"
  ```

  As not every SSH private key is supported, you can use the following command to convert your key to the acceptable format:

  ```
  ssh-keygen -m pem -p -f [***PATH_TO_PRIVATE_KEY***]
  ```
- To clone and push a local git repository on the filesystem of SSB host: file:///usr/src/ssb-examples/

b) Provide the branch name or tag of the Github repository to the **Branch** field. This branch is checked out when pulling or pushing to the Github repository.

5. Provide the name of the project to the **Project** field.
6. Enable or disable Allow deletions on import.

   Enabling the allow deletion upon import allows you to hard-reset your project to its versioned state when there are untracked resources in it. For example, the job is deleted from the project when importing the project from Github if the job does not exist in the repository. In other words, the upstream version overrides the version you have in Streaming SQL Console. If disabled, the project keeps the resources that were not pushed to the Github repository.

   The deletion affects the jobs, data sources, functions, Materialized Views and their API keys.
7. Enable or disable Authentication for the project.
8. Click Import.

What to do next

After you have created the project, you have the following options to manage the project on the **Projects** page:

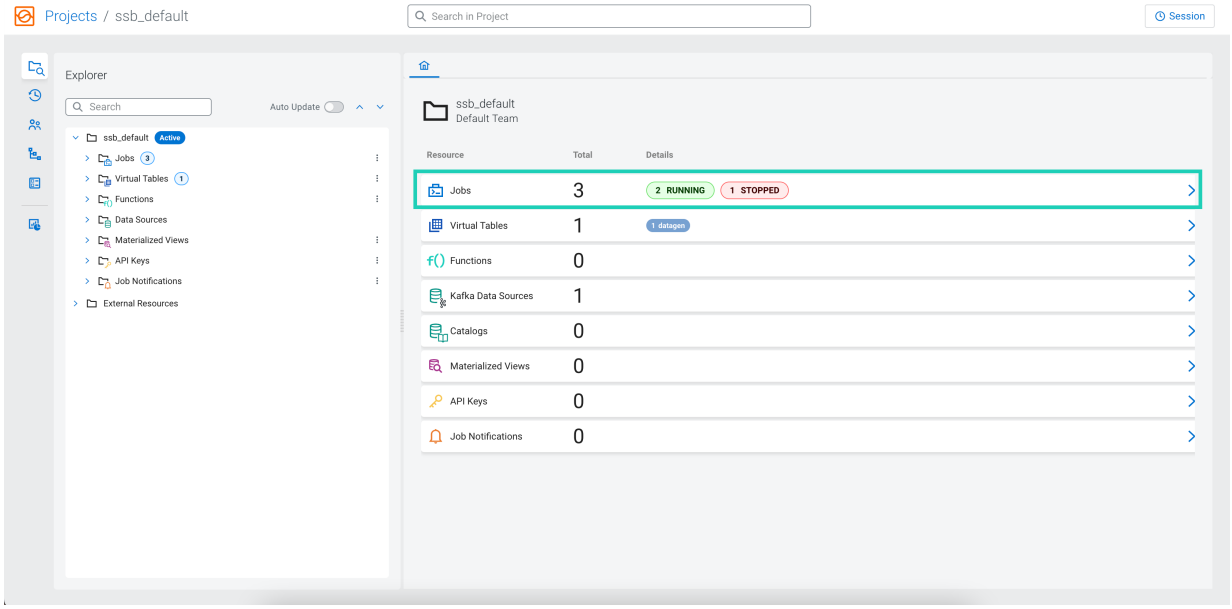
1. Click Open to use the created project and start creating SQL jobs.
2. Click Switch to open another project than the active one.
3. Click Delete to remove the project from Cloudera SQL Stream Builder.

Resuming SQL jobs

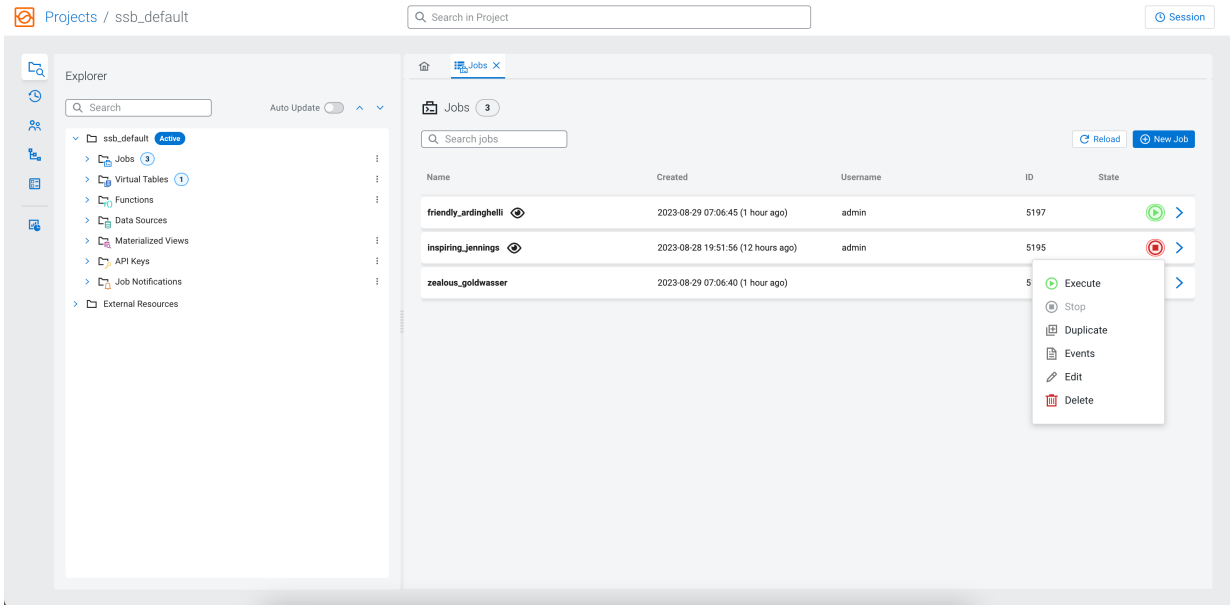
After upgrading the Cloudera SQL Stream Builder service in Cloudera Manager, resume your SQL jobs by restarting them using the SQL jobs page of Streaming SQL Console.

Procedure

1. Navigate to the Streaming SQL Console.
   a) Go to your cluster in Cloudera Manager.
   b) Select Cloudera SQL Stream Builder from the list of services.
   c) Click SQLStreamBuilder Console.
2. Select Jobs on the homepage of your project.



3. Click on  at the jobs you want to resume.



You can further filter down the results, by directly searching for the job name in the Search field.

4. Click Execute.

   If it is needed, you can edit the job by loading it to the SQL Editor using Edit, and execute it from the **SQL Editor**.

Migration

You can migrate your Flink service and SQL jobs to different clusters. When migrating the Flink service, you need to resubmit your Flink jobs on the new cluster. For migrating your SQL jobs you can use a migration tool in command line or the REST API.

Migrating Flink service to a different host

The Flink Dashboard role cannot be directly migrated from one host to another. You need to delete the existing role and add a new one to the host where you want to migrate the Flink Dashboard role. As Flink has mandatory dependencies for YARN, ZooKeeper and HDFS, these services should already be on the target host.

Before you begin

- Make sure that you have stopped your Flink jobs with savepoints, and saved your Cloudera Manager configurations.

  For more information, see the Stopping Flink applications and the Backing up the Cloudera Manager configuration documentations.
- Make sure that YARN, Zookeeper and HDFS services are already installed on the target host.

Procedure

1. Go your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Click Instances.
4. Check the box for the Flink Dashboard role.
5. Click Action for Selected > Stop to stop the Flink Dashboard role.
6. Click Action for Selected > Restart to delete the Flink Dashboard role.

   When deleting the Flink Dashboard role, an error message will be displayed as the Flink Dashboard role is mandatory for the Flink service:

   ```
   Service FLINK-1 has 0 Flink Dashboards. Flink requires at least 1 Flink Dashboard.
   ```
7. Add a new Flink Dashboard role instance to the target host.
   a) Click Add Role Instances.

      The **Add Role Instances** wizard appears.
   b) Assign the Flink Dashboard role to a host.
   c) Review the changes.
   d) Click Finish.

      The new role is stopped when added.
8. Check the box for the Flink Dashboard role.
9. Click Action for Selected > Restart to restart the new Flink Dashboard role.
10. Migrate the Flink configurations, and submit your Flink jobs to the host where the newly added Flink Dashboard role is running.

What to do next

Start your Flink applications from savepoint and restore the Cloudera Manager configurations.

Migrating SQL jobs

You can migrate your SQL jobs using the Cloudera SQL Stream Builder job migration tool. The migration tool can be used from Streaming SQL Console, Command Line Interface (CLI) or using the Cloudera SQL Stream Builder REST API. The migration tool enables you to export your SQL jobs, and import them to a different cluster or deployment when needed.

You can migrate your SQL jobs using the Streaming SQL Console, the API Explorer or the Cloudera SQL Stream Builder migration tool from CLI. The migration process consists of exporting and importing the details of the SQL

job. No matter which migration option you choose, the underlying process is the same as both solutions are based on calling the Cloudera SQL Stream Builder REST API endpoints. However, when using the API Explorer, you need to manual copy and paste the job details between the endpoints. The Streaming SQL Console and the migration tool in CLI automatically take care of accessing this information.

The following details of the SQL jobs can be exported and imported:

- SQL statement executed in the job

  When importing a SQL job under a different team, avoid using the simple names of the tables as it can cause unexpected behavior. The tables used for the query are exported with fully qualified names.
- Tables and views used in the query. This include the DDL and the tables that are in the Cloudera SQL Stream Builder catalogs

  When importing a SQL job, the tables and views are not created if there are tables and views with the same name already in the database. The tables and views that reference each other also need to be in topological order, so that the tables are only referenced when they are created.
- Table names of the external catalogs used in the query

  The external catalogs need to be registered before importing the SQL job.
- Names and SQL statements of the Java User-Defined Functions (UDFs)

  The Java jars need to be uploaded manually before importing the SQL job.
- Names and implementations of the Python UDFs
- Details of the Materialized Views



Note: Job settings, advanced configurations, job notifications, UDF jars and Materialized View queries are not exported.

Before migrating your SQL jobs, you need to stop the job if it is in a running state. To stop the SQL job you can use the Stop button either under the SQL window on the Compose page, or next to the running job on the SQL Jobs tab. Make a note of the SQL job name as you need to provide it for the job migration.

For Streaming SQL Console

You can import and export your SQL jobs using Streaming SQL Console.

The Import Job button can be found on the **Getting Started** page or on the **Console** page using

button. After clicking on the Import Job button, you need to choose the SQL job file, and provide a chosen name for it if needed. The selected SQL job is imported to the Streaming SQL Console.

The Export as JSON button can be found on the **Console** page or the **SQL Jobs** page using

button. The JSON file is automatically saved in your downloads folder and named as the job.

For API Explorer

1. Access the API Explorer.
   a. Click API Explorer on the main menu of Streaming SQL Console.
2. Open the GET/api/v1/ssb/jobs/{jobsName} operation.
3. Paste the copied job name to jobName.
4. Provide your username.
5. Click Execute.

   The details about the job is exported, and returned in a JSON string. To import the job, you need to copy the JSON detail and add it to the POST request.

6. Copy the JSON string.
7. Open the POST/api/v1/ssb/jobs/{jobsName} operation.
8. Provide a job name.
9. Provide your username.
10. Click Execute.

    The job is imported to the Streaming SQL Console, and is listed under the **SQL jobs** tab with a stopped status.

For CLI

1. Connect to your cluster using ssh.
2. Use the following command to export the job:

```
ssb-migration-tool -u <ssb_username> -pw <ssb_password> -api <api_base_u
rl> -o job-export -j <jobname> -f <target_filename>
```

3. Use the following command to import the job:

```
ssb-migration-tool -u <ssb_username> -pw <ssb_password> -api <api_base_u
rl> -o job-import -j <jobname> -f <source_filename>
```

   Example:

```
ssb-migration-tool -u test_user -pw test_password -api http://csa-test-1
.vpc.cloudera.com:18121/api/v1 -o job-export -j gallant_keller -f export
.json
```

   The job is imported to the Streaming SQL Console, and is listed under the **SQL jobs** tab with a stopped status.