Cloudera DataFlow for Data Hub 7.2.14

SQL Stream Builder

Date published: 2022-02-24
Date modified: 2022-02-24

https://docs.cloudera.com/

Legal Notice

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

Managing teams in Streaming SQL Console

You can manage your team, team members and invite new team members under the Teams menu on the Streaming SQL Console.

About this task

By default, a new user is assigned to the ssb_default team which is a default team for the administrator within SQL Stream Builder (SSB). Every user who can access the Streaming SQL Console on the same cluster, is automatically added and listed as Team Members in the ssb_default team. Only the administrator has the privilege to change the access level for a team member, and inactivate-activate a team member from the ssb_default team. Team members can create their own team. In this case, only the Team Owner can delete their team. A team can be deleted by the Team Owner of that certain team. A team cannot be deleted if it is a primary team of a user or it is the default team (ssb_default).

Every team member in a team (ssb_default team or user created team) can access the created Tables, User Defined Functions, Materialized Views and API keys within a team. A team member can also view the jobs submitted in a team they are a member of. This authorization model is also expanded to the Flink Dashboard. When you access the Flink Dashboard from the main menu of Streaming SQL Console, you can only see the jobs of the team that you select as an Active Team.

A user can be a part of multiple teams, and can switch between them. On the Streaming SQL Console, the currently selected team is shown under the Active Team header. A team member can invite other members to join their team. The invitation can be accepted or ignored. To view the invitation within a team click the View Team Invitations button.

Procedure

1. Navigate to the Streaming SQL Console.

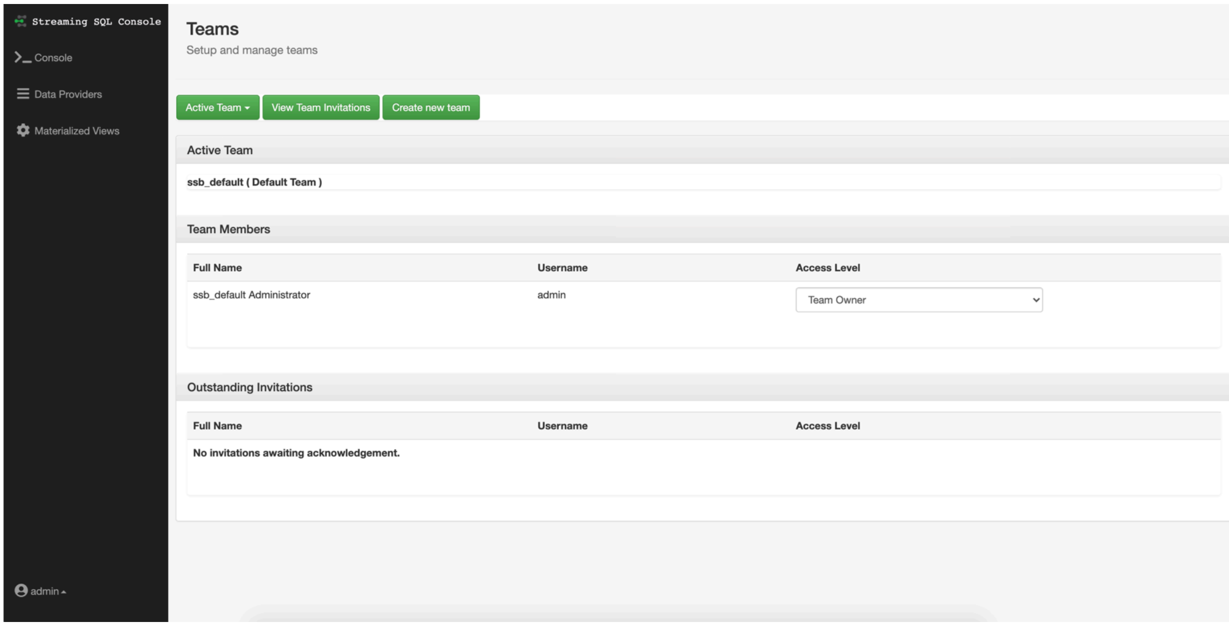
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

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

2. Select your username.

3. Click Teams.

You are redirected to the **Teams** page.



Teams

Setup and manage teams

Active Team

ssb_default (Default Team)

Team Members

Full Name	Username	Access Level
ssb_default Administrator	admin	Team Owner

Outstanding Invitations

Full Name	Username	Access Level
No invitations awaiting acknowledgement.		

Using the Streaming SQL Console

The Streaming SQL Console is the user interface for the SQL Stream Builder. You can manage your queries, tables, functions and monitor the history of the SQL jobs using the SQL Stream Console.

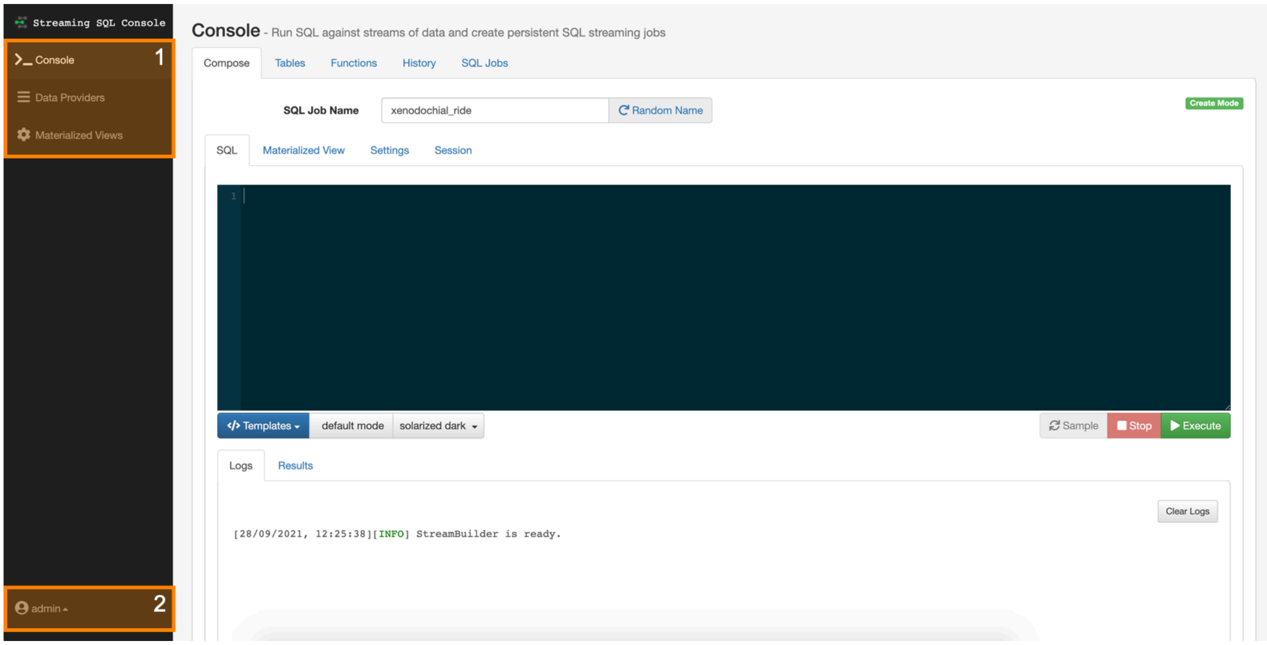
1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.

2. Select the Streaming Analytics cluster.

3. Click Streaming SQL Console from the services.

The Streaming SQL Console opens in a new window.

The following illustration details the main menu and the tabs of the user interface.



SQL Job Name: xenodochial_ride

```
[28/09/2021, 12:25:38][INFO] StreamBuilder is ready.
```

1. Main Menu

The **Main Menu** consist of the following pages:

- **Console** - The homepage of the Console where you can find the SQL window, the main tabs and audit tabs.
- **Data Providers** - The main page of Data Providers where you can add and manage the data providers and catalogs.
- **Materialized Views** - The main page of Materialized Views where you can review and manage the created Materialized View endpoints and API keys.

2. Profile Menu

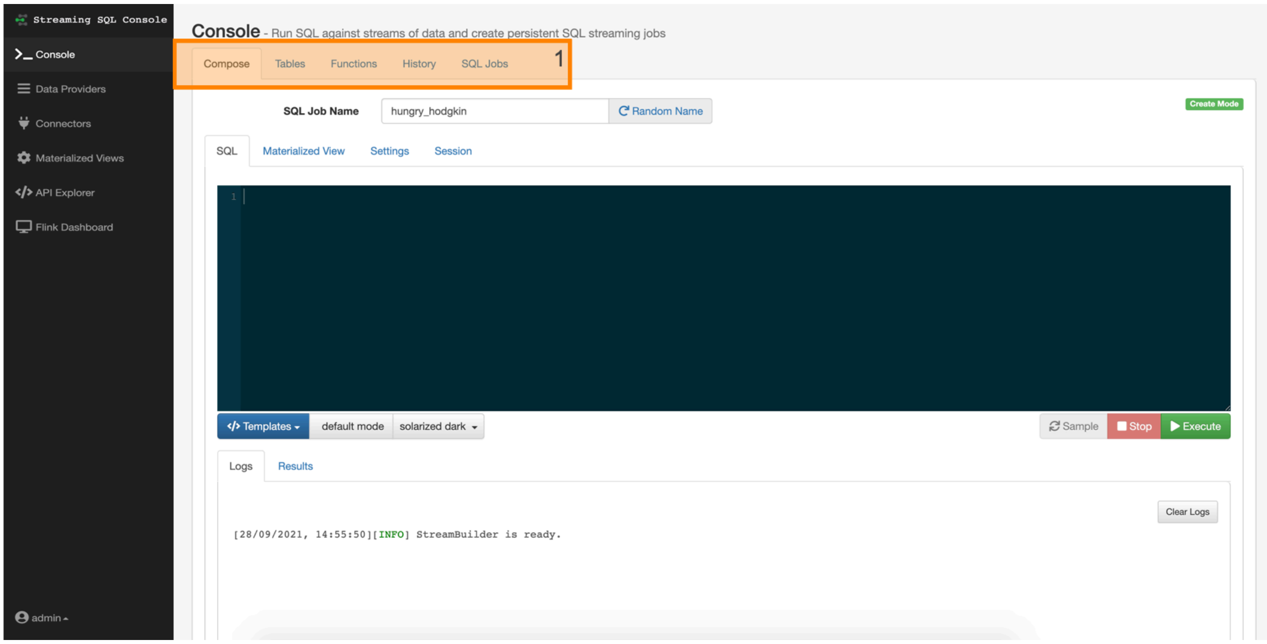
The **Profile Menu** consist of the following pages:

- **Profile** - The main page of the user profile where you can create new password.
- **Teams** - The main page of the Teams where you can create and manage your teams, invite other members to your teams.

Logout - You can use the Logout button from the **Profile Menu** to log out of the Streaming SQL Console.

Console Page

The Console page enables you to create your SQL jobs, and shows every SQL job related tabs, subtabs and audit tabs. By default the Compose tab is displayed with the SQL subtab, so you can easily create and execute your SQL queries.



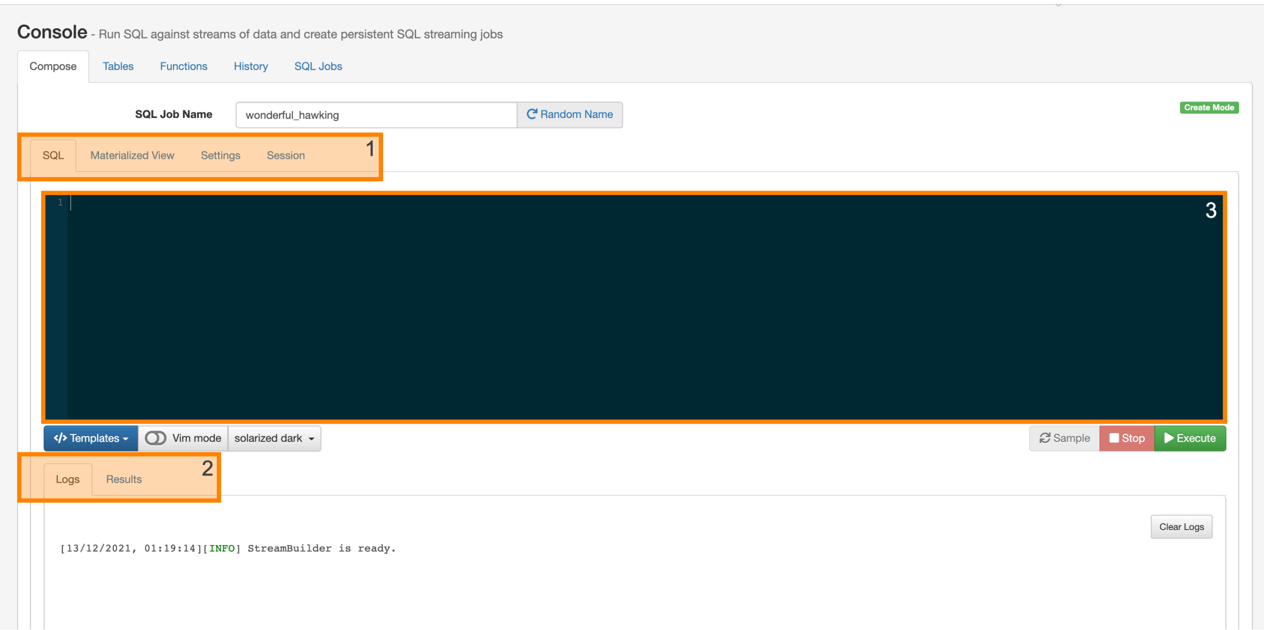
```
[28/09/2021, 14:55:50][INFO] StreamBuilder is ready.
```

1. Main Tabs

- **Compose** - By default, the Compose tab is selected on the Console.
- **Tables** - You can view, manage and add tables on this tab.
- **Functions** - You can add the User Defined Functions to your SQL query.
- **History** - You can review the history of submitted SQL queries.
- **SQL Jobs** - You can review the history of submitted SQL jobs.

Compose Tab

The Console tab shows the alternate windows to create SQL queries and Materialized Views. You can also set the SQL job settings and review the properties of the running session. With the audit tabs, you can review log information of your SQL jobs, and also check the sampled data of your query results.



```
[13/12/2021, 01:19:14][INFO] StreamBuilder is ready.
```

1. Subtabs

- **SQL** - By default, the SQL alternate window is displayed on the Compose tab where you can execute, stop and sample your SQL queries.

- **Materialized View** - The Materialized View alternate window displays settings to create Materialized Views for a SQL query.
- **Settings** - The Settings alternate window displays the SQL job relates settings that you can configure before starting a job.
- **Session** - The Session alternate window displays every property that is configurable for the running session.

2. Audit Tabs

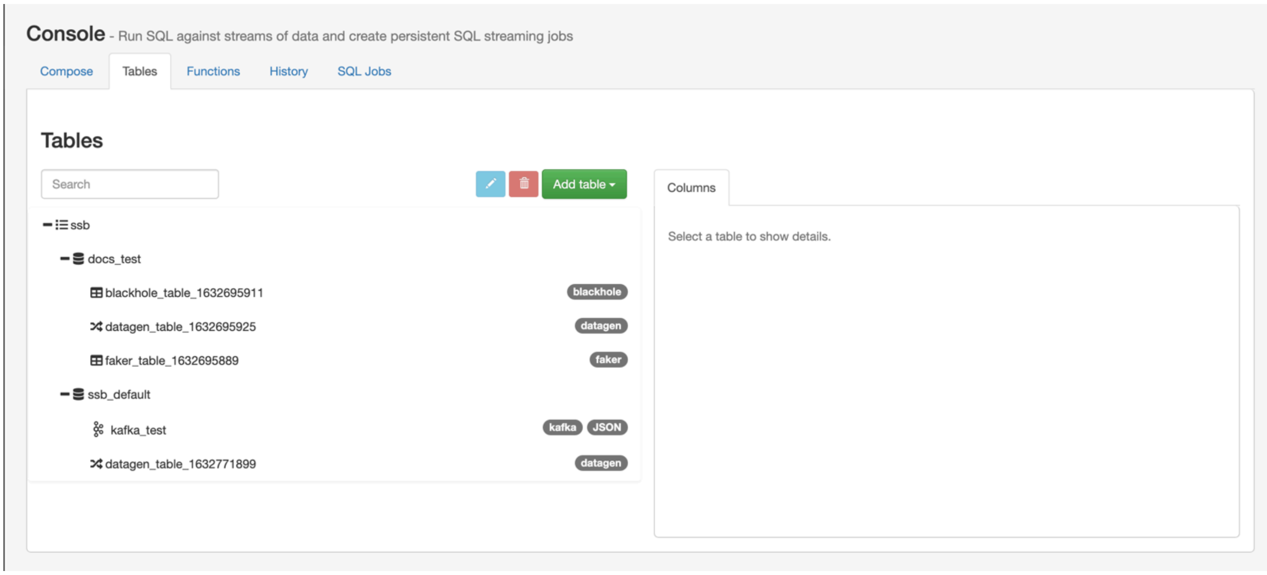
- **Logs** - The **Logs** audit tab displays the status of the SQL Console.
- **Results** - The **Results** audit tab displays the results of the executed SQL queries.

3. SQL Window

You can use the SQL window to create and add your SQL queries.

Tables Tab

The Tables tab shows the list of created tables. You can switch between Column or DDL view of the tables.



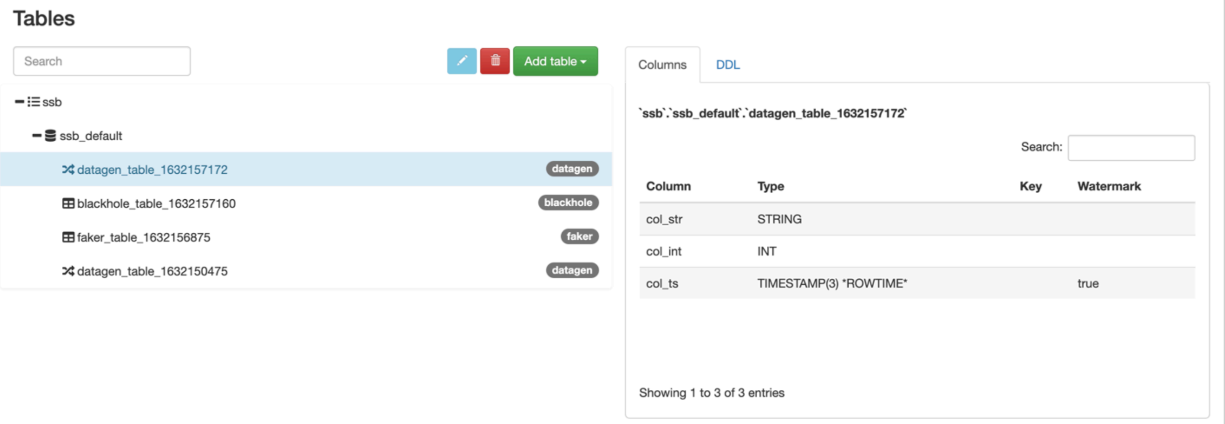
Console - Run SQL against streams of data and create persistent SQL streaming jobs

Tables

- ssb
  - docs_test
    - blackhole_table_1632695911 — blackhole
    - datagen_table_1632695925 — datagen
    - faker_table_1632695889 — faker
  - ssb_default
    - kafka_test — kafka JSON
    - datagen_table_1632771899 — datagen

Select a table to show details.

For Column view



Tables

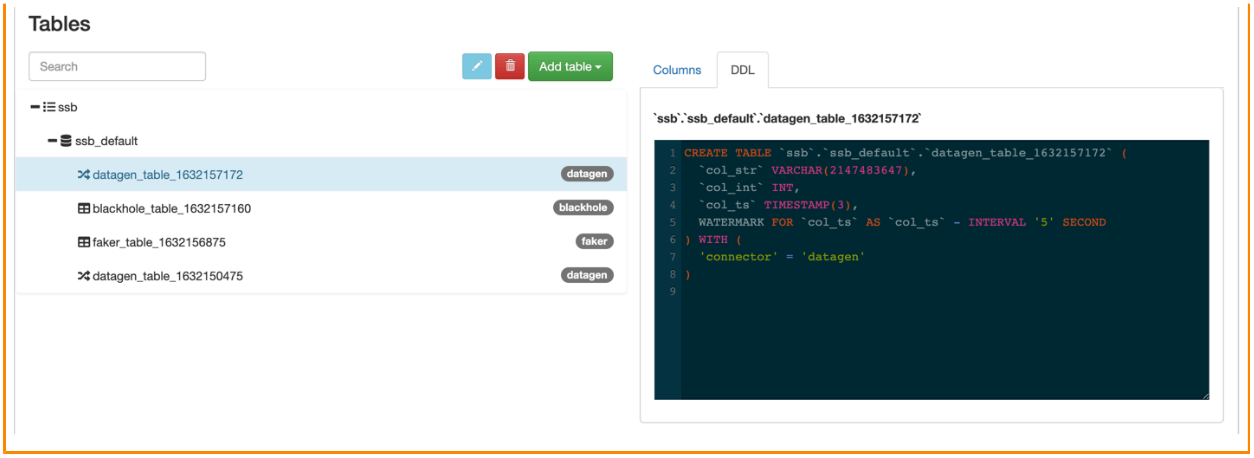
- ssb
  - ssb_default
    - datagen_table_1632157172 — datagen
    - blackhole_table_1632157160 — blackhole
    - faker_table_1632156875 — faker
    - datagen_table_1632150475 — datagen

`ssb`.`ssb_default`.`datagen_table_1632157172`

Column	Type	Key	Watermark
col_str	STRING		
col_int	INT		
col_ts	TIMESTAMP(3) *ROWTIME*		true

Showing 1 to 3 of 3 entries

For DDL view



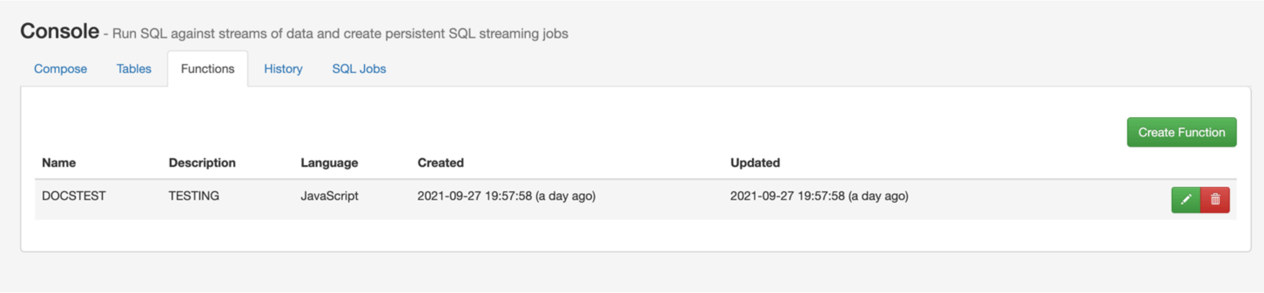
- ssb
  - ssb_default
    - datagen_table_1632157172 — datagen
    - blackhole_table_1632157160 — blackhole
    - faker_table_1632156875 — faker
    - datagen_table_1632150475 — datagen

`ssb`.`ssb_default`.`datagen_table_1632157172`

```
CREATE TABLE `ssb`.`ssb_default`.`datagen_table_1632157172` (
  `col_str` VARCHAR(2147483647),
  `col_int` INT,
  `col_ts` TIMESTAMP(3),
  WATERMARK FOR `col_ts` AS `col_ts` - INTERVAL '5' SECOND
) WITH (
  'connector' = 'datagen'
)
```

Functions Tab

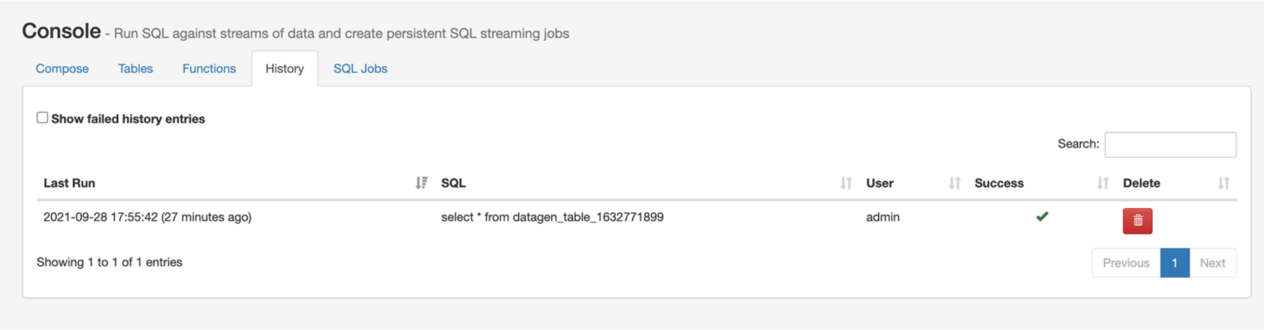
The Functions tab shows the list of created User Defined Functions (UDF) that you can use in your SQL queries.


Console - Run SQL against streams of data and create persistent SQL streaming jobs
Compose
Tables
Functions
History
SQL Jobs
Create Function

Name	Description	Language	Created	Updated
DOCSTEST	TESTING	JavaScript	2021-09-27 19:57:58 (a day ago)	2021-09-27 19:57:58 (a day ago)

History Tab

The History tab shows the list of executed SQL queries. You have the option to only show the failed history entries. When clicking on an entry from the History, the SQL query is automatically imported to the SQL window.


Console - Run SQL against streams of data and create persistent SQL streaming jobs
Compose
Tables
Functions
History
SQL Jobs
Search:
Previous
1
Next

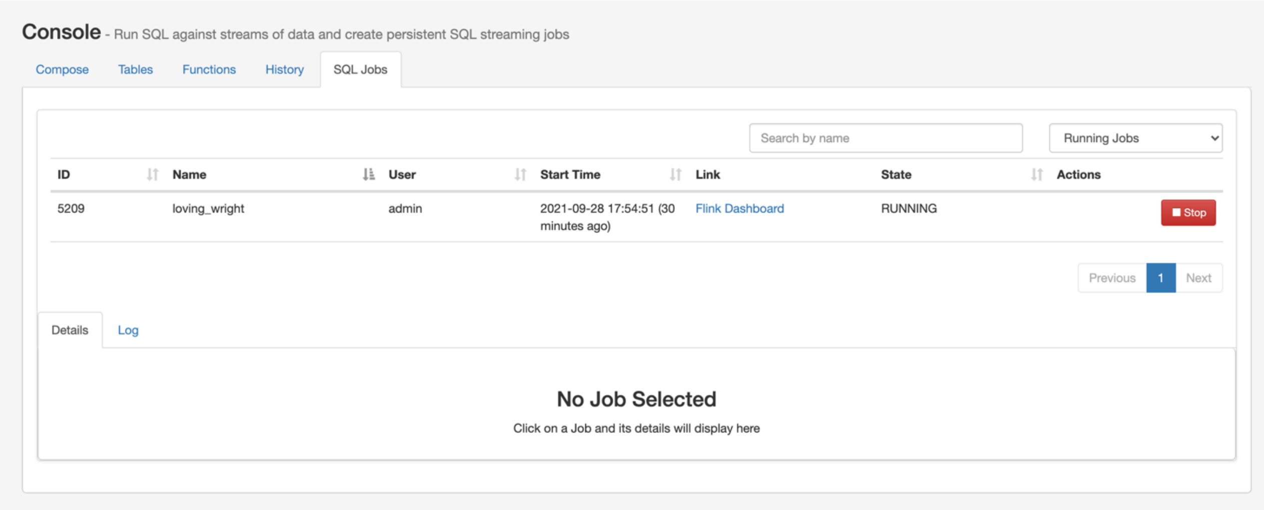
□ Show failed history entries

Last Run	SQL	User	Success	Delete
2021-09-28 17:55:42 (27 minutes ago)	select * from datagen_table_1632771899	admin	✓	

Showing 1 to 1 of 1 entries

SQL Jobs Tab

The SQL Jobs tab include the list of running and stopped SQL jobs. You can also manage and stop the SQL jobs listed on the SQL Jobs tab. Using the Details and Log windows, you can review more information about a selected job.



Console - Run SQL against streams of data and create persistent SQL streaming jobs

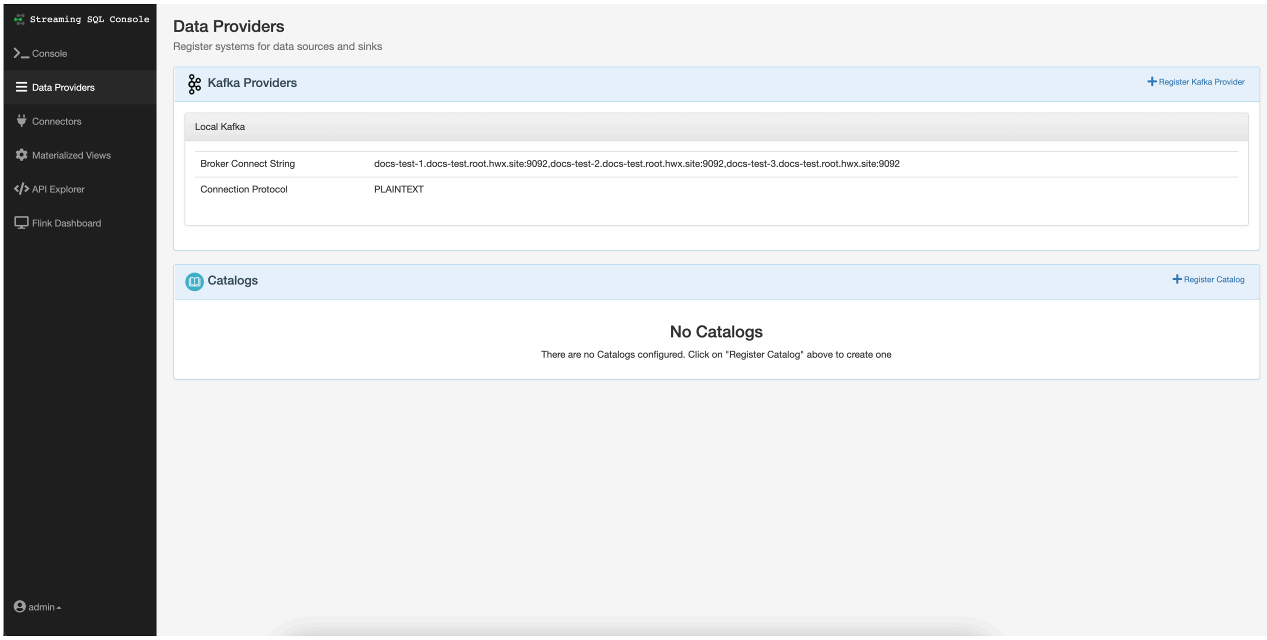
ID	Name	User	Start Time	Link	State	Actions
5209	loving_wright	admin	2021-09-28 17:54:51 (30 minutes ago)	Flink Dashboard	RUNNING	Stop

No Job Selected

Click on a Job and its details will display here

Data Providers Page

The Data Providers page enables you to register, edit and delete the data providers. The providers are displayed in two categories: Kafka providers and Catalogs.



Data Providers

Register systems for data sources and sinks

Kafka Providers

Local Kafka

Broker Connect String	docs-test-1.docs-test.root.hwx.site:9092,docs-test-2.docs-test.root.hwx.site:9092,docs-test-3.docs-test.root.hwx.site:9092
Connection Protocol	PLAINTEXT

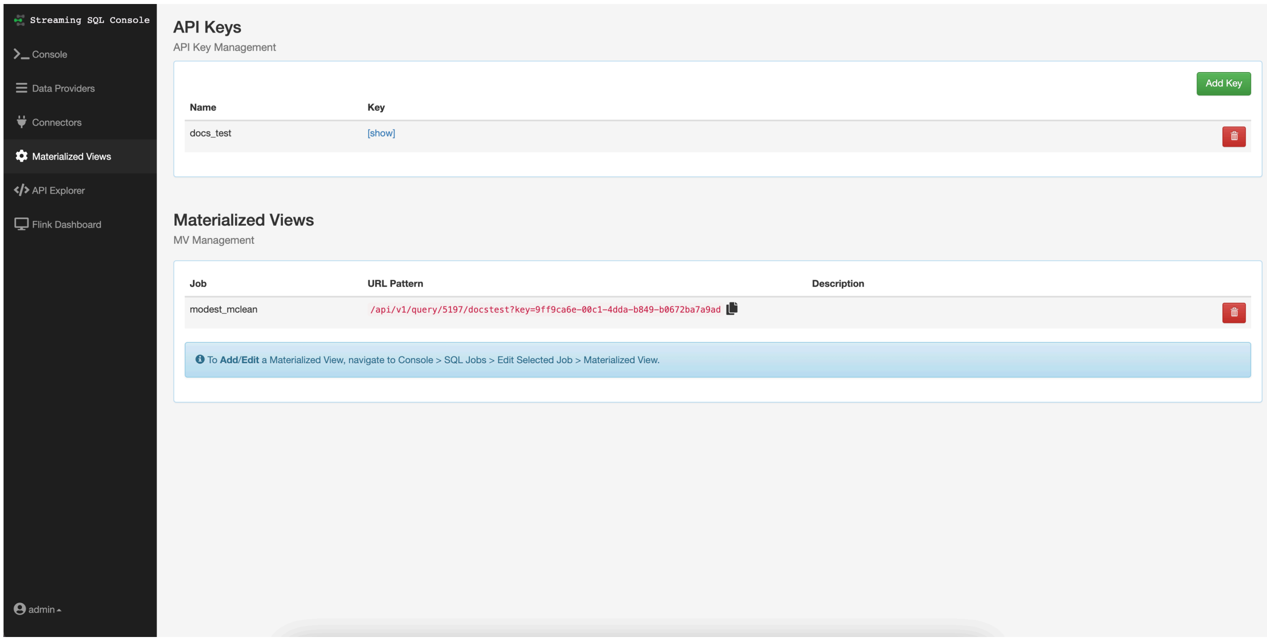
Catalogs

No Catalogs

There are no Catalogs configured. Click on "Register Catalog" above to create one

Materialized Views Page

The Materialized Views page enables you to add API keys, manage API keys and manage Materialized Views.



API Keys

API Key Management

Name	Key
docs_test	[show]

Materialized Views

MV Management

Job	URL Pattern	Description
modest_mclean	/api/v1/query/5197/docstest?key=9ff9ca6e-00c1-4dda-b849-b0672ba7a9ad	

To **Add/Edit** a Materialized View, navigate to Console > SQL Jobs > Edit Selected Job > Materialized View.

Registering Data Providers in SSB

Data Providers are a set of data endpoints to be used as sources, sinks and catalogs. Data Providers allow you to connect to an already installed component on your cluster, then use that provider for adding tables in SQL Stream Builder (SSB).

You can access the **Data Providers** page through the Streaming SQL Console:

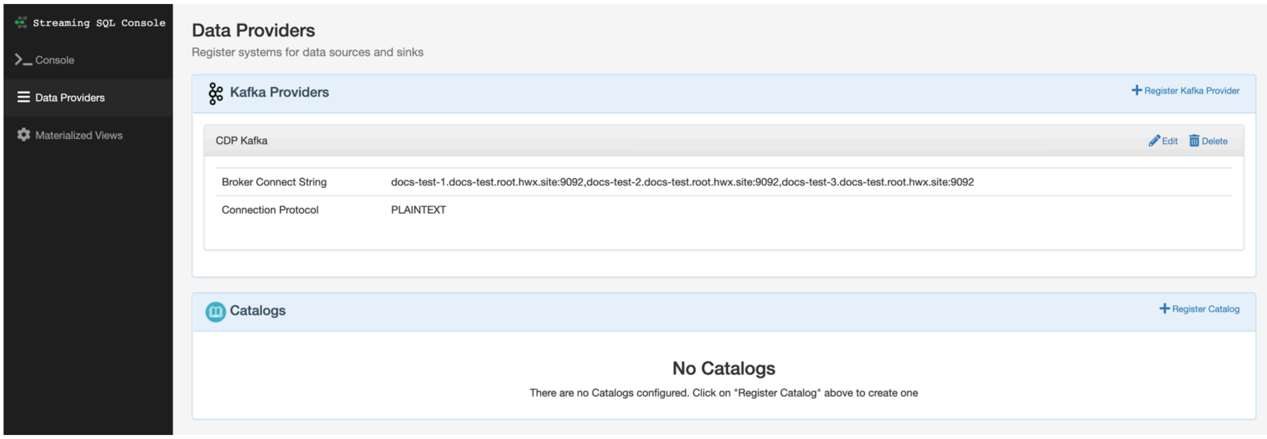
1. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
2. Select the Streaming Analytics cluster.
3. Click Streaming SQL Console from the services.

   The Streaming SQL Console opens in a new window.
4. Click Data Providers on the main menu.

   You are redirected to the Data Providers page.

You can register Kafka as a data provider, or Kudu, Hive and Schema Registry as a catalog. When registering the components, SSB can access the already existing topics from Kafka, tables from Kudu and Hive, and the schema in Schema Registry. This also means that when you update a data provider, for example add new topics, tables and schemas, SSB automatically detects the changes.

You can also manage your data providers after registering them. You can Edit the providers if there is any change in the connection. You can also Delete them when you no longer need the specific provider.



Data Providers

Register systems for data sources and sinks

Kafka Providers

CDP Kafka

Broker Connect String	docs-test-1.docs-test.root.hwx.site:9092,docs-test-2.docs-test.root.hwx.site:9092,docs-test-3.docs-test.root.hwx.site:9092
Connection Protocol	PLAINTEXT

Catalogs

No Catalogs

There are no Catalogs configured. Click on "Register Catalog" above to create one

Related Information

Adding Kafka as Data Provider

Adding catalogs as Data Provider

Using auto discovery of services

Using the auto discovery of services in SQL Stream Builder (SSB), you can easily connect to catalogs and data providers that are running on a different Data Hub cluster in the same CDP Public Cloud environment.

Important: The Auto discovery of services feature is provided as a technical preview at this time. The tool is still under development and not recommended for a production environment.

You can use the service discovery to automatically register Kafka and Kudu with SSB within the same environment. This means when you have a Streams Messaging or Real-Time Data Mart cluster beside your Streaming Analytics cluster in your environment, you can use the auto discovery to automatically import the tables from both services. Using the auto discovery has the same result as manually adding the catalogs and providers: you can read from and write to Kafka and Kudu. However, when you have numerous Kafka and Kudu services in your environment, you do not have to add them one by one, instead you can use the service discovery to import them all at once.

For more information about manually adding data providers and catalogs, see the Registering Data providers in SSB section.

Note: In the Streaming Analytics 7.2.14 cluster definition, Apache Hive and Schema Registry are not supported for the auto discovery feature of SSB.

The auto discovery feature does not check the changes you make in your environment. This means that you can register the newly added services from your environment to SSB using the same process. However, when you delete the cluster from the environment, the already registered catalog and data providers are not removed from SSB, you need to manually delete them from the Streaming SQL Console.

Setting up the auto discovery

As an administrator, you need to make sure to set up a machine user that is used to register the services from the clusters to SSB. The machine user information must be provided to the user.

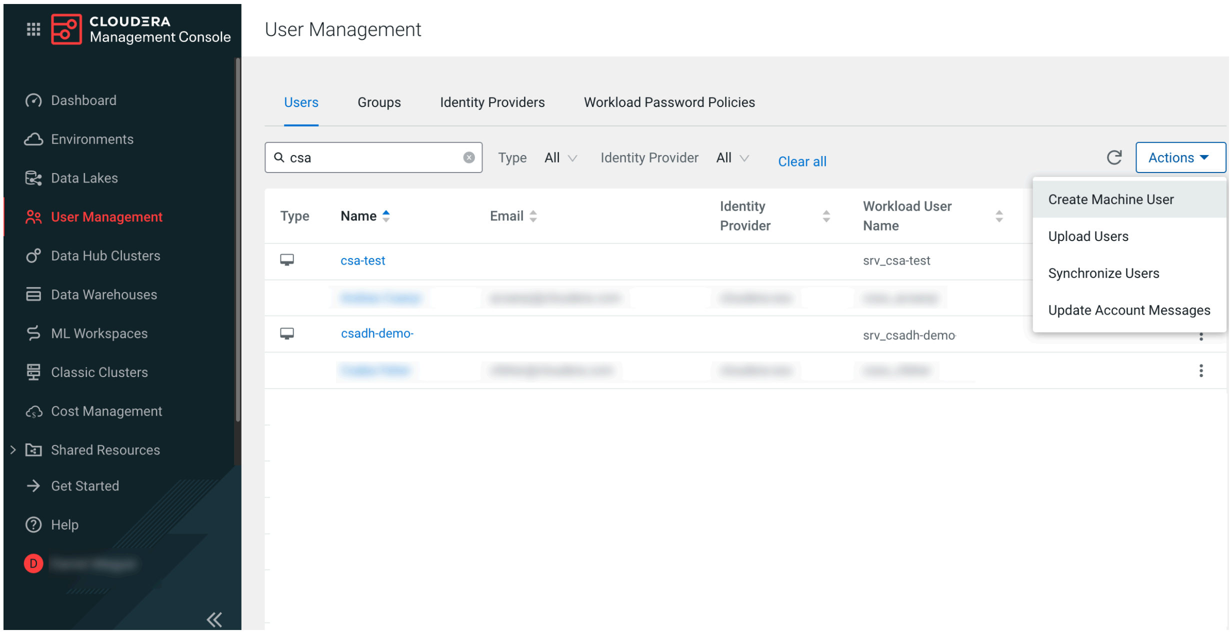
About this task

For the auto discovery, you need to create a machine user, and assign the correct role so the machine user has access to the environment. The machine user also must be synchronized to the environment.

Important: Make sure that the network policies are correctly set up between the CDP Public Cloud environment and the cloud provider. For more information, see the cloud provider specific information under *Related information*.

Procedure

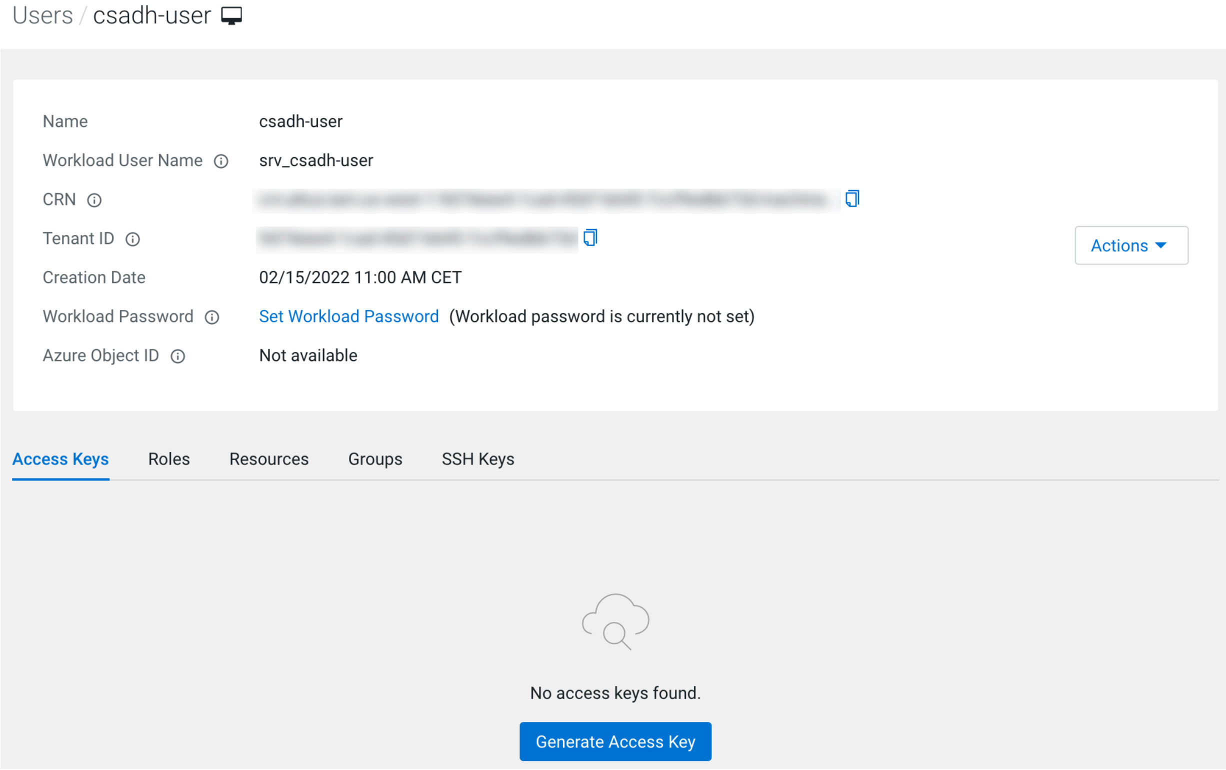
1. Navigate to Management Console > User Management page.
2. Click Actions > Create Machine User .



Type	Name	Email	Identity Provider	Workload User Name
🖥	csa-test			srv_csa-test
	[redacted]	[redacted]	[redacted]	[redacted]
🖥	csadh-demo-			srv_csadh-demo
	[redacted]	[redacted]	[redacted]	[redacted]

3. Provide a name to the Machine User.
4. Click Create.

   You are redirected to the **User Profile** page.


Users / csadh-user 🖥

Name csadh-user

Workload User Name ⓘ srv_csadh-user

CRN ⓘ [redacted]

Tenant ID ⓘ [redacted]

Creation Date 02/15/2022 11:00 AM CET

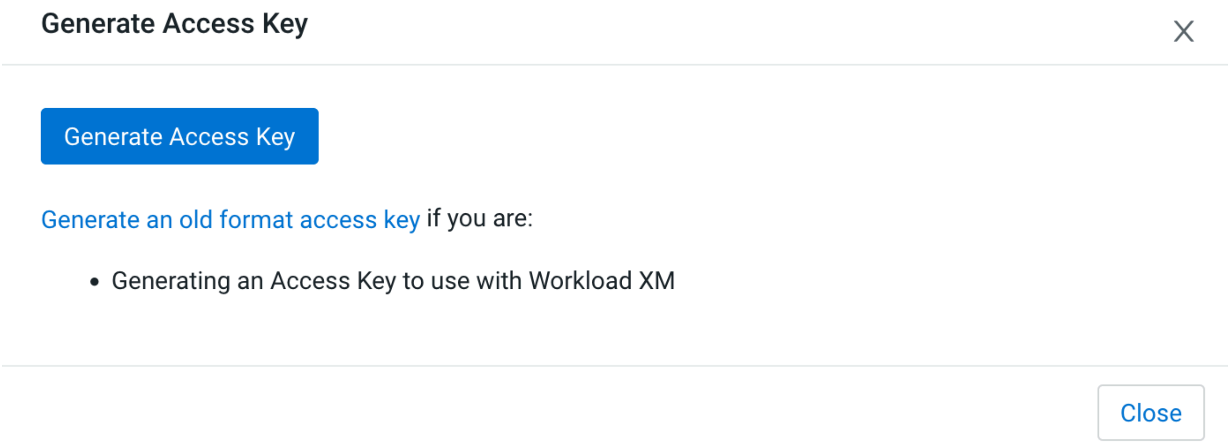
Workload Password ⓘ Set Workload Password (Workload password is currently not set)

Azure Object ID ⓘ Not available

No access keys found.

5. Click Set Workload Password.
6. Provide a Workload Password for the Machine User.
7. Click Set Workload Password.

8. Select Generate Access Key under **Access Keys** tab.



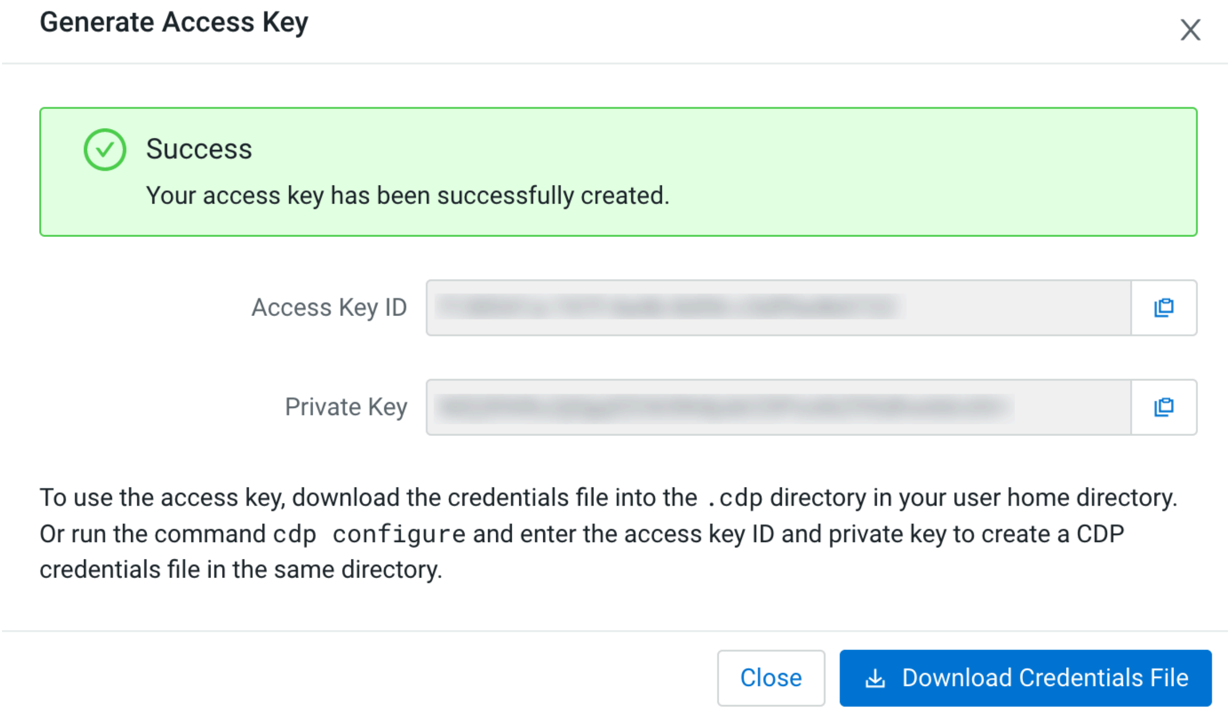
Generate Access Key

Generate Access Key

Generate an old format access key if you are:

- Generating an Access Key to use with Workload XM

9. Click Generate Access Key.
   The **Generate Access Key** window pops up to indicate that the access key generation was successful.



Generate Access Key

Success

Your access key has been successfully created.

Access Key ID [redacted]

Private Key [redacted]

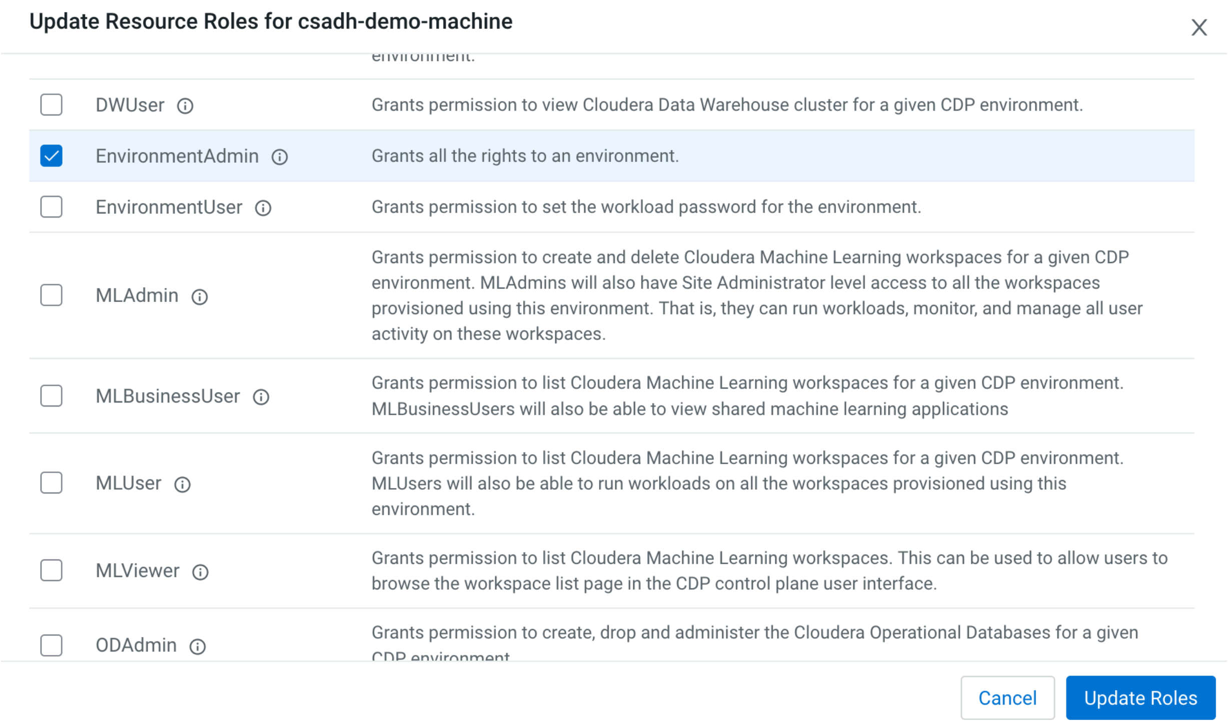
To use the access key, download the credentials file into the `.cdp` directory in your user home directory. Or run the command `cdp configure` and enter the access key ID and private key to create a CDP credentials file in the same directory.

10. Take note of the Access Key ID and Private Key.
11. Click Download Credentials File.

Note: Make sure to download the credentials file as you will not be able to retrieve the private key after closing the window.

12. Click Close.
13. Navigate to Management Console > Environments and select your environment.
14. Click Actions > Manage Access .
15. Search for the Machine User you have created.

16. Select *EnvironmentAdmin* role from the list of Resource Roles.



Update Resource Roles for csadh-demo-machine

environment.

□	DWUser ⓘ	Grants permission to view Cloudera Data Warehouse cluster for a given CDP environment.
☑	EnvironmentAdmin ⓘ	Grants all the rights to an environment.
□	EnvironmentUser ⓘ	Grants permission to set the workload password for the environment.
□	MLAdmin ⓘ	Grants permission to create and delete Cloudera Machine Learning workspaces for a given CDP environment. MLAdmins will also have Site Administrator level access to all the workspaces provisioned using this environment. That is, they can run workloads, monitor, and manage all user activity on these workspaces.
□	MLBusinessUser ⓘ	Grants permission to list Cloudera Machine Learning workspaces for a given CDP environment. MLBusinessUsers will also be able to view shared machine learning applications
□	MLUser ⓘ	Grants permission to list Cloudera Machine Learning workspaces for a given CDP environment. MLUsers will also be able to run workloads on all the workspaces provisioned using this environment.
□	MLViewer ⓘ	Grants permission to list Cloudera Machine Learning workspaces. This can be used to allow users to browse the workspace list page in the CDP control plane user interface.
□	ODAdmin ⓘ	Grants permission to create, drop and administer the Cloudera Operational Databases for a given CDP environment

17. Click Update Roles.

    The Resource Role for the selected user or group will be updated.

18. Navigate to Management Console > Environments , and select the environment where you want to create a cluster.
19. Click Actions > Synchronize Users to FreeIPA .
20. Click Synchronize Users.

What to do next

After setting up the auto discovery feature, you need to access the Streaming SQL Console and import the data providers or catalogs from the clusters in your environment.

Related Information

AWS Requirements: VPC and subnets

Azure Requirements: VNet and subnets

GCP Requirements: VPC network and subnet

Using the auto discovery on Streaming SQL Console

Using the Streaming SQL Console, you can import the Kafka and Kudu services that already exist in your CDP Public Cloud environment. For the auto discovery, you need to provide the environment and machine user information that was provided by your administrator.

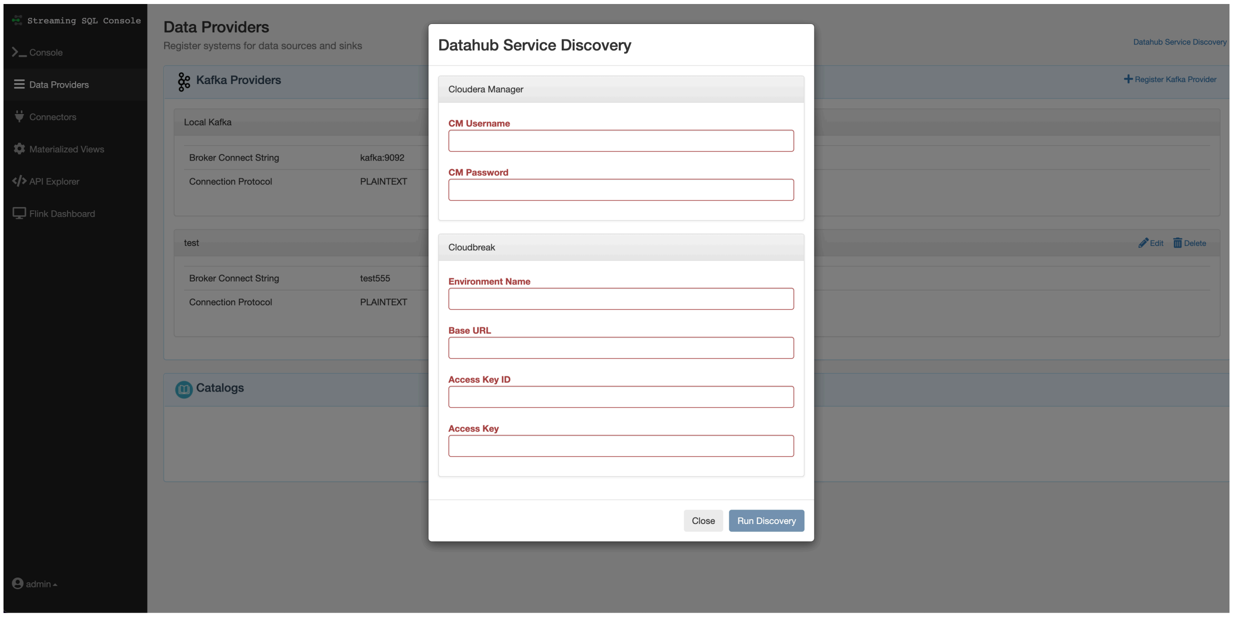
Procedure

1. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

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

2. Click Data Providers.
3. Click Datahub Service Discovery.

   The **Datahub Service Discovery** window appears.



Local Kafka

Broker Connect String	kafka:9092
Connection Protocol	PLAINTEXT

test

Broker Connect String	test555
Connection Protocol	PLAINTEXT

Datahub Service Discovery

Cloudera Manager

CM Username

CM Password

Cloudbreak

Environment Name

Base URL

Access Key ID

Access Key

4. Provide the needed information to the corresponding field.

   Important: The environment and machine user information that is needed for the service discovery is provided by the administrator.

Table 1:

Field	Description	Example
CM Username	Workload username of the Machine User.	*csa-test*
CM Password	Workload password of the Machine User.	*Password123!*
Environment Name	The name of the CDP Public Cloud environment you are using.	*csa-test-env*
Base URL	The URL snippet from the Management Console access URL.	*https://console.us-west-1.cdp.cloudera.com*
Access Key ID	Access Key ID of the Machine User.	*00a0a0aa-00aa-00aa-00a0-0000a0a00a0*
Access Key	Private Key of the Machine User.	*Z66pxEXAMPLEKEY +Y3oIU070GEHS3kqi1EXAMPLEKEY=*

5. Click Run discovery.

   The existing Kafka and Kudu services in your environment should appear under Data Providers and Catalogs

Managing registered Data Providers

You can edit or delete the registered Data Providers if you need to change their configurations or if you no longer need them.

Editing registered Data Providers

1. Click Data Providers from the main menu.
2. Search for the Kafka provider or catalog you want to modify.

3. Click Edit.

   The Edit Provider or Catalog window appears.
4. Change the settings as required.

   Note: You must validate the modified catalog before saving the changes.
5. Click Save Changes.

Deleting registered Data Providers

1. Click Data Providers from the main menu.
2. Search for the Kafka provider or catalog you want to modify.
3. Click Delete.
4. Click Confirm to delete the provider or catalog.

Connectors in SSB

Connector support in SSB

SQL Stream Builder (SSB) supports different connector types and data formats for Flink SQL tables to ease development and access to all kinds of data sources.

The following table summarizes the supported connectors and how they can be used in SSB:

Connector	Type	Description
Kafka	source/sink	Supported as exactly-once-sink
Hive	source/sink	Can be used as catalog
Kudu	source/sink	Can be used as catalog
Schema Registry	source/sink	Can be used as catalog
JDBC	source/sink	Can be used with Flink SQL. PostgreSQL, MySQL and Hive are supported.
Filesystems	source/sink	Filesystems such as HDFS, S3 and so on. Can be used with Flink SQL
Debezium CDC	source	Can be used with Flink SQL. PostgreSQL, MySQL, Oracle DB and Db2 are supported.
Webhook	sink	Can be used as HTTP POST/PUT with templates and headers
PostgreSQL	sink	Materialized View connection for reading views. Can be used with anything that reads PostgreSQL wire protocol
REST	sink	Materialized View connection for reading views. Can be used with anything that reads REST (such as notebooks, applications, and so on)
BlackHole	sink	Can be used with Flink SQL.

Kafka connectors

When using the Kafka connector, you can choose between using an internal or external Kafka service. Based on the connector type you choose, there are mandatory fields where you must provide the correct information.

You can choose from the following Kafka connectors when creating a table in Streaming SQL Console:

Template: local-kafka

Using the Kafka service that is installed on your cluster.

Type: source/sink

The following fields are mandatory to use the connector:

- scan.startup.mode: Startup mode for the Kafka consumer. group-offsets is the default value. You can choose from earliest-offset, latest-offset, timestamp and specific-offsets as startup mode.
- topic: The topic from which data is read as a soruce, or the topic to which data is written to. No default value is specified. You can also add a topic list in case of sources. In this case, you need to separate the topics by semicolon. You can only specify the topic-pattern or topic for the sources.
- format: The format used to deserialize and serialize the value part of Kafka messages. No default value is specified. You can use either the format or the value.format option.

Template: kafka

Using an external Kafka service as a connector. To connect to the external Kafka service, you need to speficy the Kafka brokers that are used in your deployment.

Type: source/sink

The following fields are mandatory to use the connector:

- properties.bootstrap.servers: Specifying a list of Kafka brokers that are separated by comma. No default value is specified.
- topic: The topic from which data is read as a soruce, or the topic to which data is written to. No default value is specified. You can also add a topic list in case of sources. In this case, you need to separate the topics by semicolon. You can only specify the topic-pattern or topic for the sources.
- format: The format used to deserialize and serialize the value part of Kafka messages. No default value is specified. You can use either the format or the value.format option.

Template: upsert-kafka

Using the upsert Kafka service as a connector. For more information about the upsert Kafka connector, see the Apache Flink documentation.

Type: source/sink

- properties.bootstrap.servers: Specifying a list of Kafka brokers that are separated by comma. No default value is specified.
- topic: The topic from which data is read as a soruce, or the topic to which data is written to. No default value is specified. You can also add a topic list in case of sources. In this case, you need to separate the topics by semicolon. You can only specify the topic-pattern or topic for the sources.
- key.format: The format used to deserialize and serialize the key part of Kafka messages. No default value is specified. Compared to the regular Kafka connector, the key fields are specified by the PRIMARY KEY syntax.
- value.format: The format used to deserialize and serialize the value part of Kafka messages. No default value is specified. You can use either the format or the value.format option.

Using the Kafka connectors

You can access and import the templates of the Kafka connectors from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

      The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select one of the Kafka templates you want to use.

   The template is imported to the SQL window.
5. Provide information to the mandatory fields of the template.
6. Click Execute.

CDC connectors

You can use the Debezium Change Data Capture (CDC) connector to stream changes in real-time from MySQL, PostgreSQL, Oracle, Db2 and feed data to Kafka, JDBC, the Webhook sink or Materialized Views using SQL Stream Builder (SSB).

Concept of Change Data Capture

Change Data Capture (CDC) is a process to capture changes in a source system, and update the data within a downstream system or application with the changes.

The Debezium implementation offers CDC with database connectors from which real-time events are updated using Kafka and Kafka Connect. Debezium captures every row-level change in each database table of an event stream. Applications read these streams to see the change events in the same order as they occurred. The change events are routed to a Kafka topic from which Kafka Connect feeds the records to other systems and databases.

For more information about Debezium, see the official Debezium site.

CDC in Cloudera Streaming Analytics (CSA) does not require Kafka or Kafka Connect as Debezium is implemented as a library within the Flink runtime. This means that the captured changes are propagated downstream to any connector that Flink supports. CSA allows queries to be issued at change data capture time, which means filtering, grouping, joining, and so on, can be performed on the change stream as it comes from the source database.

For more information about the Flink implementation of Debezium, see the official Apache Flink documentation.

From the supported set of Debezium connectors, MySQL, PostgreSQL, Oracle and Db2 are supported in Cloudera Streaming Analytics.

Note: The Db2 connector is only supported for Linux operating system.

Note: You need to configure the databases, users and permissions for the supported connectors before you are able to use them in SSB. For more information about setting up the databases, see the MySQL, PostgreSQL, Oracle, Db2 documentation.

Using the CDC connectors

You can access and import the templates of the CDC connectors from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
    a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
    b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
    c. Select Streaming SQL Console from the list of services.

    The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select one of the CDC templates you want to use.

    The template is imported to the SQL window.
5. Provide information to the mandatory fields of the template.
6. Click Execute.

JDBC connector

When using the JDBC connector, you can choose between using a PostgreSQL, MySQL or Hive databases. Based on the connector type you choose, there are mandatory fields where you must provide the correct information.

Template: jdbc

Using either PostgreSQL, MySQL or Hive as databases. When you use the JDBC connector, you must specify the JDBC database which are going to be used for the connection.

Type: source/sink

The following fields are mandatory to use the connector:

- url: The URL of the JDBC database. No default value is specified.
- table-name: The name of the JDBC table in the database that you need to connecto to. No default value is specified.

Using the JDBC connector

You can access and import the templates of the JDBC connector from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
    a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
    b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
    c. Select Streaming SQL Console from the list of services.

    The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select the JDBC template.

    The template is imported to the SQL window.
5. Provide information to the mandatory fields of the template.
6. Click Execute.

Filesystem connector

When using the Filesystem connector, you can choose between HDFS, S3 and so on storage systems. Based on the connector type you choose, there are mandatory fields where you must provide the correct information.

Template: filesystem

Using either HDFS, S3 or any type of storage system. When you use the Filesystem connector, you must specify the path to the file system which are going to be used for the connection.

Type: source/sink

The following fields are mandatory to use the connector:

- path: Path to the root directory of the table data. No default value is specified.
- format: The format used to for the file system. No default value is specified.

Using the Filesystem connector

You can access and import the templates of the Filesystem connector from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select the Filesystem template.

   The template is imported to the SQL window.
5. Provide information to the mandatory fields of the template.
6. Click Execute.

Datagen connector

The Datagen connector can be used a source to generate random data. The Datagen connector works out of the box, no mandatory field is required to use the connector. The Datagen connector is useful when you want to experiment and try out SQL Stream Builder or to test your SQL queries using random data.

Template: datagen

You do not need to provide any type of information when using the Datagen connector and template.

Type: source

Using the Datagen connector

You can access and import the templates of the Datagen connector from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select the Datagen template.

   The template is imported to the SQL window.
5. Click Execute.

Faker connector

The Faker connector can be used a source to generate random data. To use the Faker connector, you need to specify the usage The Datagen connector is useful when you want to experiment and try out SQL Stream Builder or to test your SQL queries using random data.

Template: faker

You do not need to provide any type of information when using the Datagen connector and template.

Type: source

The following fields are mandatory to use the connector:

- fields.#.expression: The Java Faker expression to generate the values for a specific field. For more information about the list and use of the Faker expressions, see the flink-faker documentation.

Using the Faker connector

You can access and import the templates of the Datagen connector from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

   The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select the Faker template.
5. Provide information to the mandatory fields of the template.
6. Click Execute.

Blackhole connector

The Blackhole connector can be used a sink where you can write any type of data into. The Blackhole connector works out of the box, no mandatory field is required to use the connector. The Blackhole connector is useful when you want to experiment and try out SQL Stream Builder or to test your SQL queries using random data.

Template: blackhole

You do not need to provide any type of information when using the Blackhole connector and template.

Type: sink

Using the Blackhole connector

You can access and import the templates of the Blackhole connector from Streaming SQL Console:

1. Navigate to the Streaming SQL Console.
   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c. Select Streaming SQL Console from the list of services.

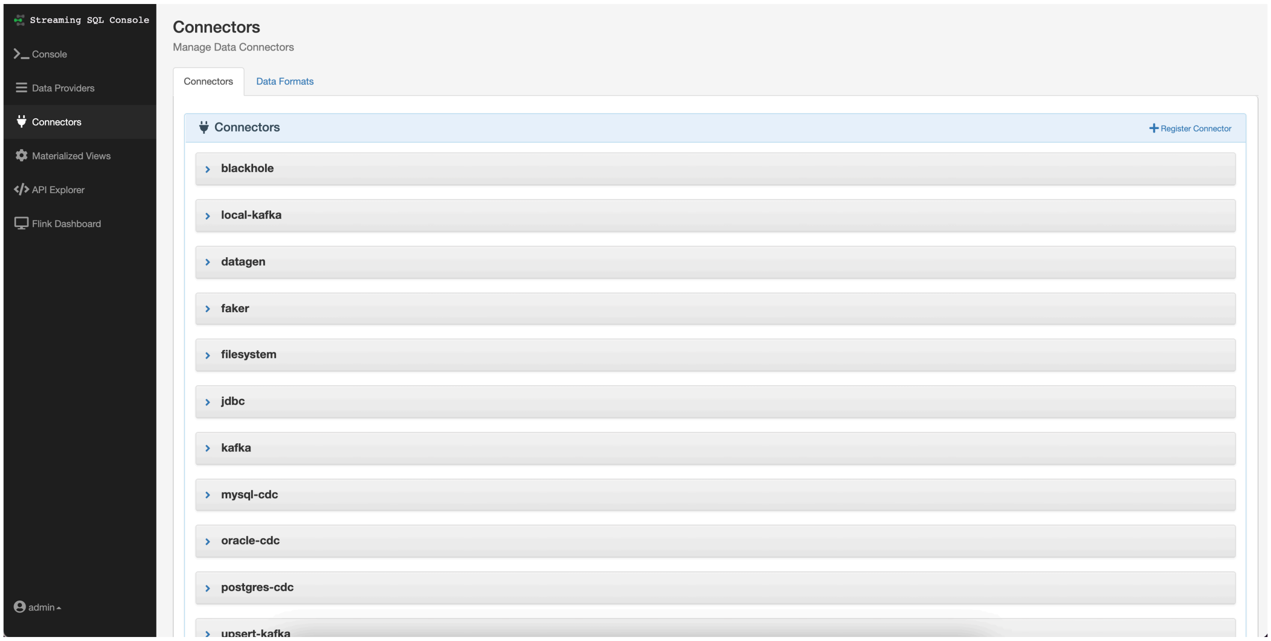
   The **Streaming SQL Console** opens in a new window.
2. Select Console from the main menu.
3. Click Templates under the SQL window.
4. Select the Blackhole template.

   The template is imported to the SQL window.
5. Click Execute.

Managing connectors and data formats

You can add new connectors and data formats in SQL Stream Builder (SSB) using the Streaming SQL Console. The newly added connectors and data formats will appear in Templates under the SQL window on the Compose page.

The newly added connectors and data formats are automatically added to the list of Templates in SSB. The templates are built based on the properties and data formats defined for the connectors.



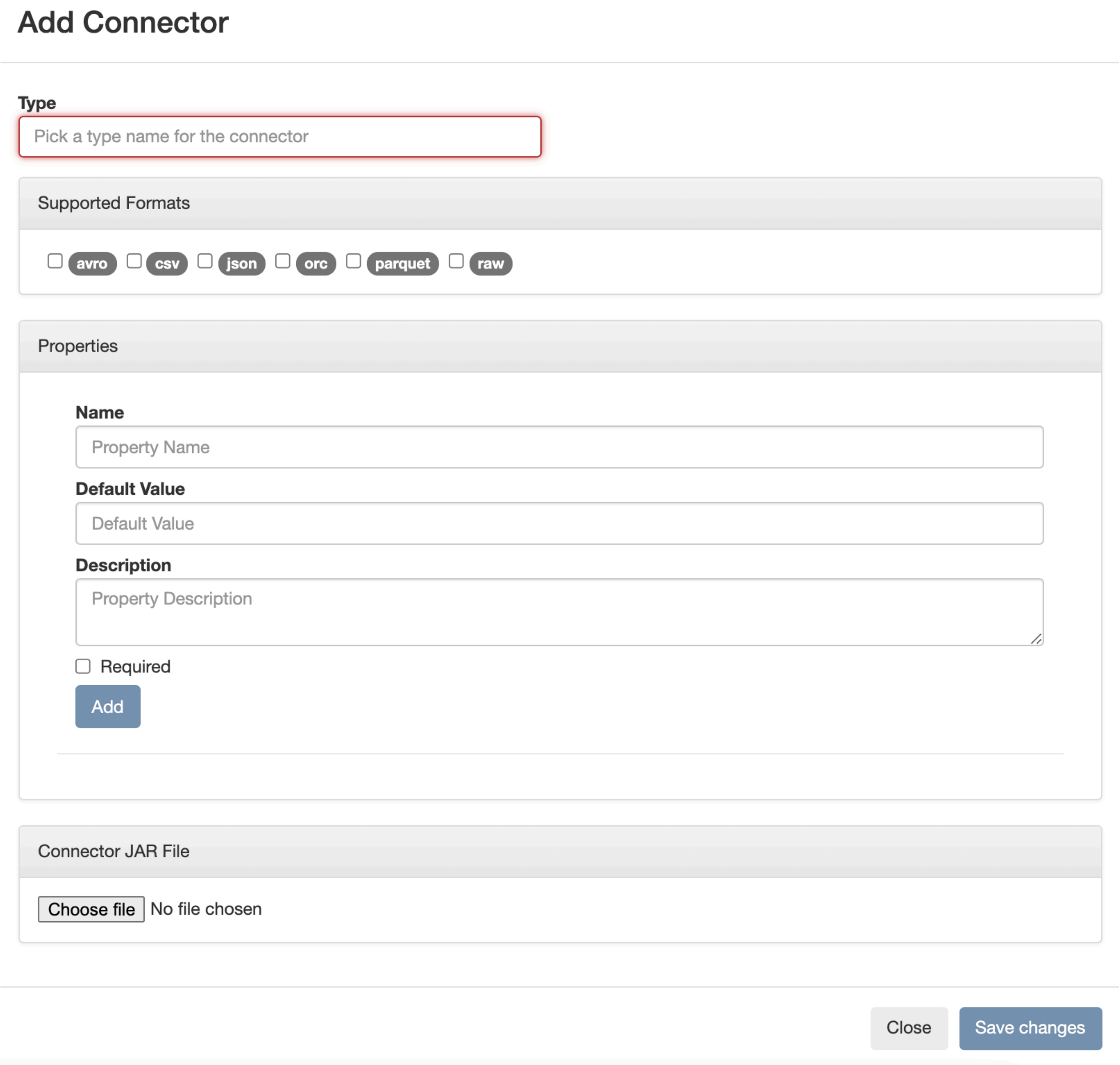
Adding new connectors

When adding new connectors to SQL Stream Builder, you need to specify the property list, data types and must upload the connector JAR file to the Console.

Procedure

1. Select Connectors on the main menu.

2. Click Register Connector.



3. Provide a name for the connector as Type.
4. Select a data format to be supported for the connector from the Supported Formats.
5. Provide properties that are mandatory or optional for the connector.
   a) Add a name to the property.
   b) Add a default value to the property.
   c) Add a description to the property.
   d) Check the Required checkbox to make a property mandatory.
   e) Click Add.

   You can specify as many properties as needed for the connector.

6. Upload the connector JAR file.

   Note: When uploading a JAR file larger than 1 MB, you need to set the server.tomcat.max-swallow-size in Cloudera Manager using the following steps:

   a. Open your cluster in Cloudera Manager.
   b. Select SQL Stream Builder from the list of services.
   c. Select Configuration.
   d. Search for Streaming SQL Engine Advanced Configuration Snippet (Safety Valve) for ssb-conf/application.properties in the search bar.
   e. Add server.tomcat.max-swallow-size=2000MB to the **Safety Valve**.
   f. Click Save.
   g. Restart the SQL Stream Builder service.
7. Click Save changes.

   The newly added connector is listed on the Connectors page.

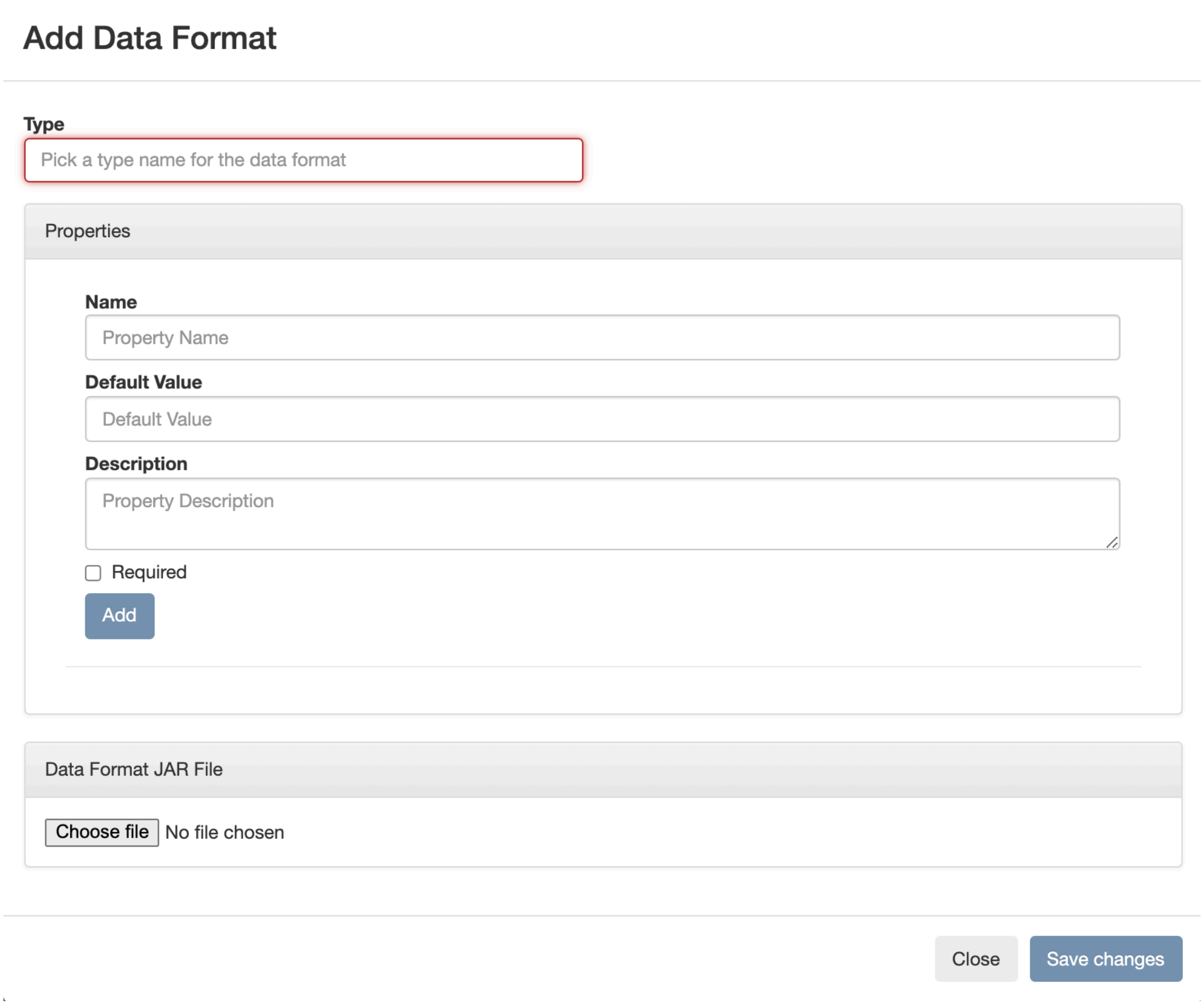
Adding data formats

When adding new data formats to SQL Stream Builder, you need to specify the property list and must upload the data format JAR file to the Console.

Procedure

1. Select Connectors on the main menu.
2. Click Data Formats.

3. Click Register Data Format.



Add Data Format

Type

Pick a type name for the data format

Properties

Name

Property Name

Default Value

Default Value

Description

Property Description

□ Required

4. Provide a name for the data format as Type.
5. Provide properties that are mandatory or optional for the data format.
   a) Add a name to the property.
   b) Add a default value to the property.
   c) Add a description to the property.
   d) Check the Required checkbox to make a property mandatory.
   e) Click Add.

   You can specify as many properties as needed for the data format.
6. Upload the data format JAR file.
7. Click Save changes.

   The newly added data format is listed on the Data Formats page.

Concept of tables in SSB

The core abstraction for Streaming SQL is a Table which represents both inputs and outputs of the queries. SQL Stream Builder (SSB) tables are an extension of the tables used in Flink SQL to allow a bit more flexibility to the users. When creating tables in SSB, you have the option to either add them manually, import them automatically or create them using Flink SQL depending on the connector you want to use.

A Table is a logical definition of the data source that includes the location and connection parameters, a schema, and any required, context specific configuration parameters. Tables can be used for both reading and writing data in most cases. You can create and manage tables either manually or they can be automatically loaded from one of the catalogs as specified using the Data Providers section.

In SELECT queries the FROM clause defines the table sources which can be multiple tables at the same time in case of JOIN or more complex queries.

When you execute a query, the results go to the table you specify after the INSERT INTO statement in the SQL window. This allows you to create aggregations, filters, joins, and so on, and then route the results to another table. The schema for the results is the schema that you have created when you ran the query.

For example:

```
INSERT INTO air_traffic -- the name of the table sink
SELECT
lat,lon
FROM
airplanes -- the name of the table source
WHERE
icao <> 0;
```

Tables types in SSB

Kafka Tables

Apache Kafka Tables represent data contained in a single Kafka topic in JSON, AVRO or CSV format. It can be defined using the Streaming SQL Console wizard or you can create Kafka tables from the pre-defined templates.

Tables from Catalogs

SSB supports Kudu, Hive and Schema Registry as catalog providers. After registering them using the Streaming SQL Console, the tables are automatically imported to SSB, and can be used in the SQL window for computations.

Note: You cannot edit the properties of the already existing tables that are automatically imported from the catalogs. To distinguish between editable and not editable tables, in other words, user defined and catalog tables, the Edit and Delete table options are not available on the Tables page.

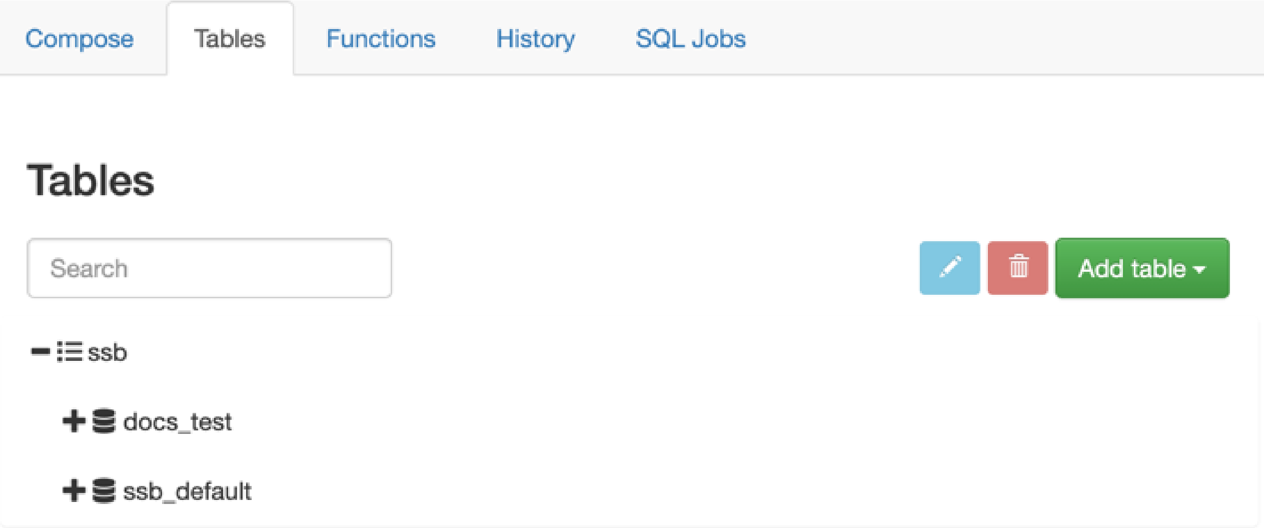
Flink Tables

Flink SQL tables represent tables created by the standard CREATE TABLE syntax. This supports full flexibility in defining new or derived tables and views. You can either provide the syntax by directly adding it to the SQL window or use one of the predefined DDL templates.

Webhook Tables

Webhooks can only be used as tables to write results to. When you use the Webhook Tables the result of your SQL query is sent to a specified webhook.

After creating your tables for the SQL jobs, you can review and manage them on the Tables tab in Streaming SQL Console. The created tables are organized based on the teams a user is assigned to.



Related Information

Creating Kafka tables

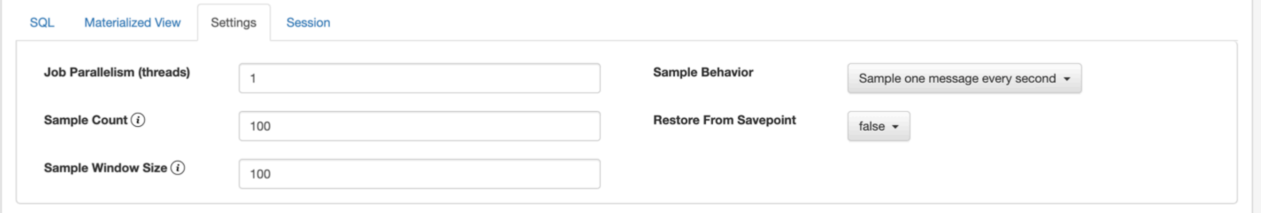
Creating tables with Flink DDL

Job Lifecycle

Configuring SQL job settings

If you need to further customize your SQL Stream job, you can add more advanced features to configure the job restarting method and time, threads for parallelism, sample behavior, exatly once processing and restoring from savepoint.

Before running a SQL query, you can configure advanced features by clicking on the Settings tabs at the SQL window.



Job Parallelism (threads)	1	Sample Behavior	Sample one message every second
Sample Count	100	Restore From Savepoint	false
Sample Window Size	100		

Job parallelism (threads)

The number of threads to start to process the job. Each thread consumes a slot on the cluster. When the Job Parallelism is set to 1, the job consumes the least resources. If the data provired supports parallel reads, increasing the parallelism can raise the maximum throughput. For example, when using Kafka as a data provider, setting the parallelism to the equal number as the partitions of the topic can be a starting point for performance tuning.

Sample Count

The number of sample entries shown under the Results tab. To have an unlimited number of sample entries, add 0 to the Sample Count value.

Sample Window Size

The number of sample entries to keep in under the Results tab. To have an unlimited number of sample entries, add 0 to the Sample Window Size value.

Sample Behavior

You have the following options to choose the behavior of the sampled data under the Results tab:

- Sample all messages
- Sample one message every second
- Sample one message every five seconds

Restore From Savepoint

You can enable or disable restoring a SQL job from a Flink savepoint after stopping it. The savepoint is saved under hdfs:///user/flink/savepoints by default.

Adjusting logging configuration in Advanced Settings

You can customize the logging configurations for the SQL Stream Builder (SSB) job on the Streaming SQL Console in per-job mode or session mode. Adjusting the log configuration enables you to control the log levels of all the underlying libraries: Flink, Hadoop, Kafka, Zookeeper, other common libraries, and connectors to get more or less information in your job’s log.

About this task

The customization of the log configuration works differently based on the job deployment mode:

Session mode

The execution.target is set to *yarn-session* mode, this is the default execution mode.

The log configuration is set at the start time if the Flink YARN session is applied to every job execution. For example, the current log configuration is applied if and only if the Flink YARN session is not set on the Session tab of the Compose page.

> **Note:** The Reset Session button only resets the SSB Session, not the underlying Flink YARN session. To do that, you have to kill the YARN application that is indicated under Flink Yarn Session on the Session tab.

Per-job mode

The execution.target is set to *yarn-per-job* mode.

When you change the default execution mode to per-job, the currently applied log configuration is going to be used for the job. To configure the execution mode, you need to start the SQL query with the following line:

```
SET 'execution.target'='yarn-per-job';
```

Procedure

1. Select Console on the main menu.
2. Select Settings from the Compose page.
3. Click Advanced Settings.

   The **Custom Log Configuration** window appears.
4. Click into the **configuration** window or click Edit.
5. Modify the settings based on your requirements.
6. Click **Apply**.
7. Add and execute a SQL statement.

8. Click **SQL Jobs**.
9. Search for the job you have executed previously.
10. Click **Flink Dashboard**.

    The **Flink Dashboard** opens in a new window.
11. Click Task Managers > Logs .

    The log information appears in the log window based on your custom configurations.

Stopping, restarting and editing SQL jobs

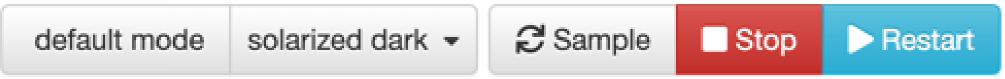
As a SQL Stream job processes streaming data, you need to stop the job to finish the process. You can restart a SQL Stream job after stopping it. In case you need to update or change the configurations that you have set for a SQL Stream job, you can restart it. You can navigate through the job life cycle using the Streaming SQL Console.

Stopping a SQL Stream job

You can stop a running job either on the **Compose** or the **SQL Jobs** page.
Stopping job from Compose page

1. Select Console from the main menu.

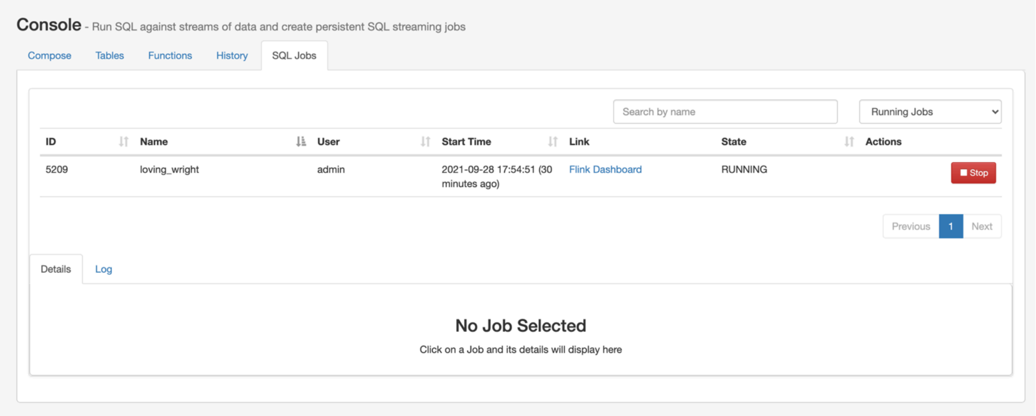
   By default, you are on the Compose page when selecting Console from the main menu.
2. Click Stop under the SQL window.



Stopping job from SQL Jobs page

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.



ID	Name	User	Start Time	Link	State	Actions
5209	loving_wright	admin	2021-09-28 17:54:51 (30 minutes ago)	Flink Dashboard	RUNNING	Stop

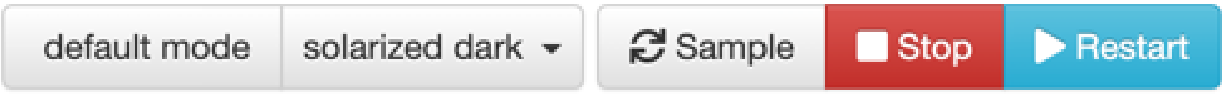
No Job Selected

Click on a Job and its details will display here

3. Click on the job you want to stop.
   a. You can further filter down the results, by directly searching for the job name in the Search field.
4. Click Stop under Actions.

Restarting a running SQL Stream job

You can restart a running SQL job using the Restart button under the SQL window.

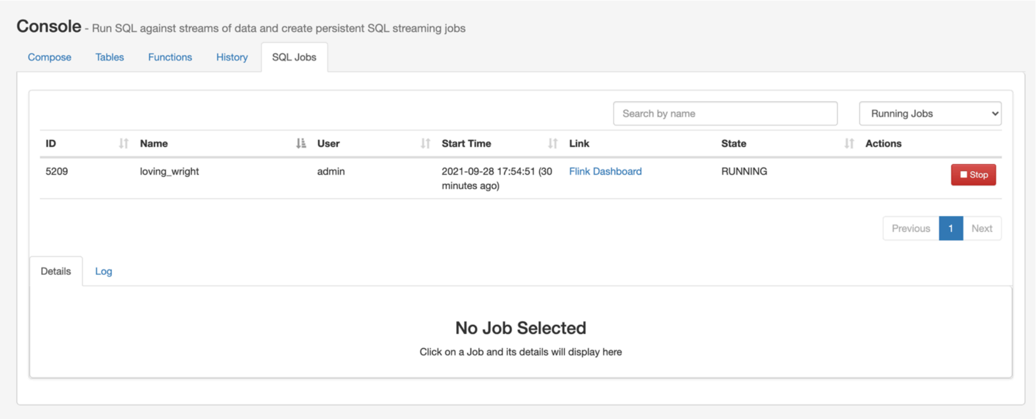


In case the job is running in the background, you can load it with its properties from the SQL Jobs tab.

Restarting job from SQL Jobs page

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.

   Console - Run SQL against streams of data and create persistent SQL streaming jobs

   Running Jobs

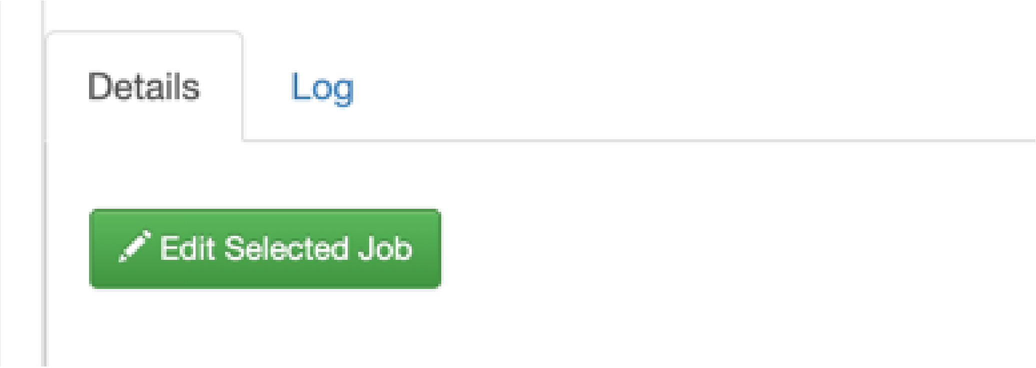
ID	Name	User	Start Time	Link	State	Actions
5209	loving_wright	admin	2021-09-28 17:54:51 (30 minutes ago)	Flink Dashboard	RUNNING	Stop

   No Job Selected

   Click on a Job and its details will display here

3. Click on the job you want to restart.

   a. You can further filter down the results, by directly searching for the job name in the Search field.
4. Click Edit Selected Job under the **Details** tab.

   The SQL job and its configuration is loaded in the SQL window on the **Compose** page.
5. Click Restart.

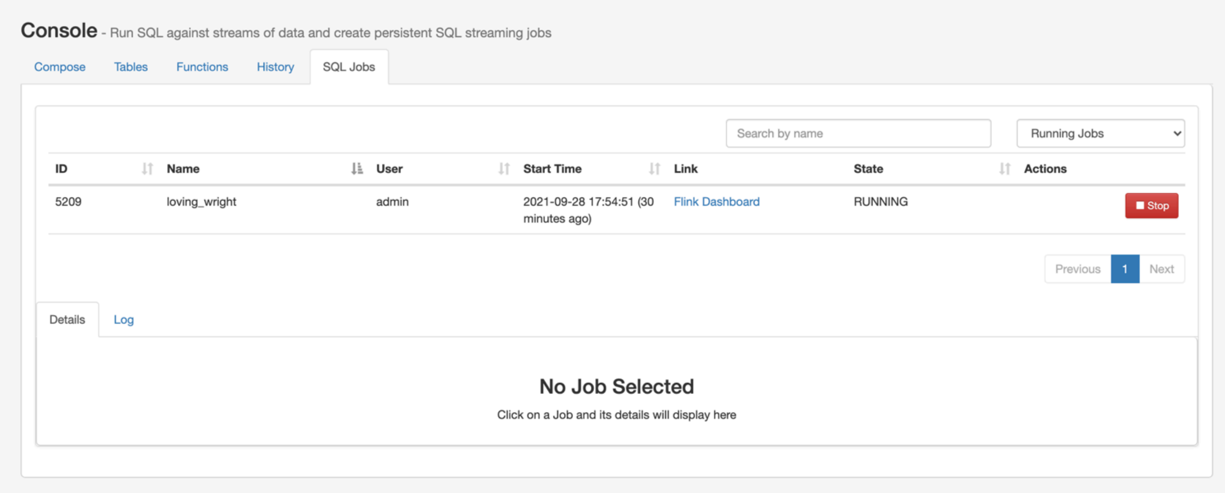
Restarting a stopped SQL Stream job

You can restart a SQL job that was previously stopped by locating it in the SQL Jobs page.

1. Click Console on the main menu.

2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.



Console - Run SQL against streams of data and create persistent SQL streaming jobs

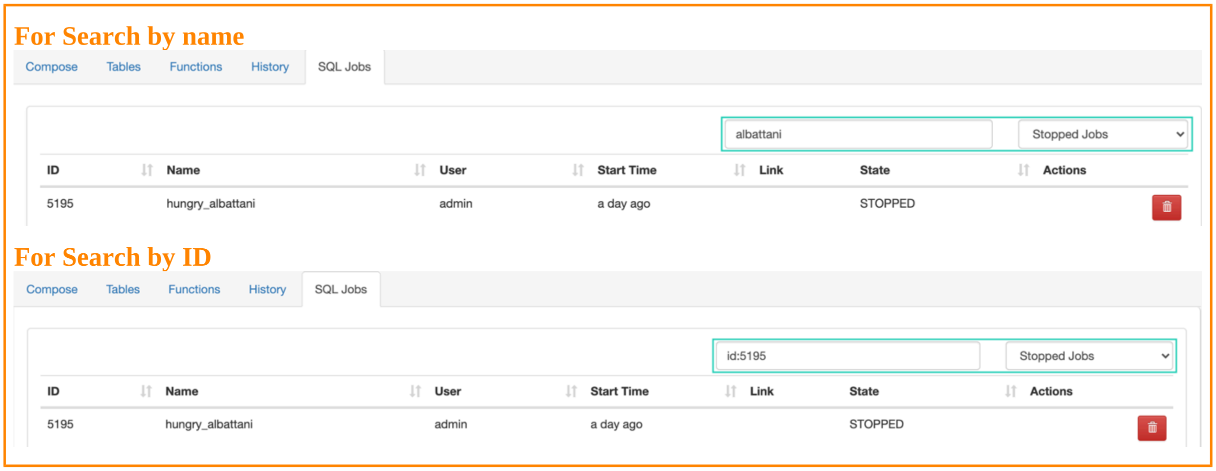
ID	Name	User	Start Time	Link	State	Actions
5209	loving_wright	admin	2021-09-28 17:54:51 (30 minutes ago)	Flink Dashboard	RUNNING	Stop

No Job Selected

Click on a Job and its details will display here

3. Select Stopped Jobs from the drop-down menu.
4. Click on the job you want to restart.

   a. You can further filter down the job list by searching for the job name, or locate a specific job ID by prefixing your search with id.



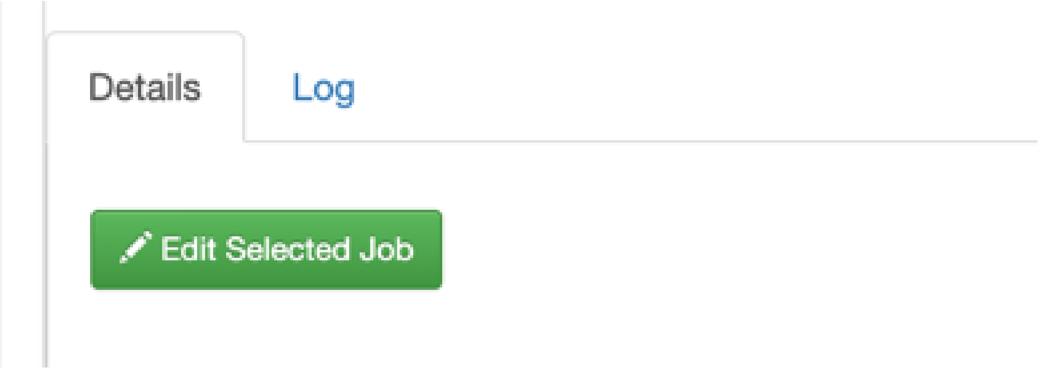
For Search by name

ID	Name	User	Start Time	Link	State	Actions
5195	hungry_albattani	admin	a day ago		STOPPED	🗑

For Search by ID

ID	Name	User	Start Time	Link	State	Actions
5195	hungry_albattani	admin	a day ago		STOPPED	🗑

5. Click Edit Selected Job under the **Details** tab.



   The SQL job and its configuration is loaded in the SQL window on the **Compose** page.

6. Click on Execute.

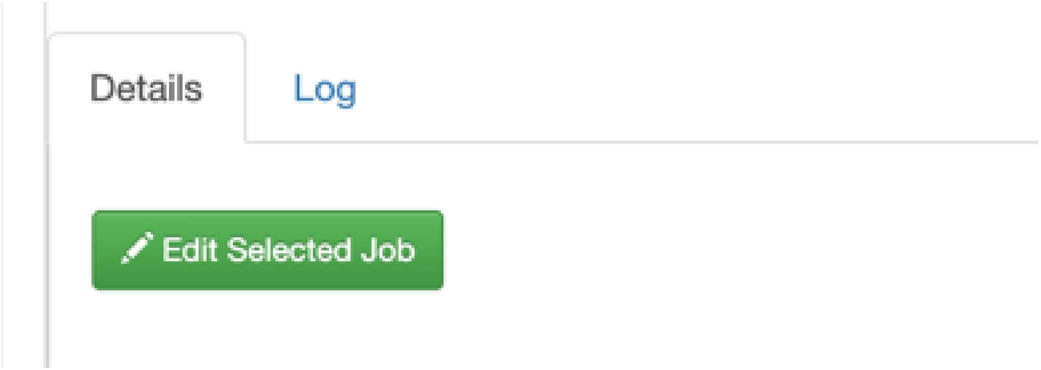
Note: If you are using Materialized Views, the same Materialized View parameters will be selected. Make sure that the configured parameters are still appropriate for the job if the source schemas have been modified.

Editing a SQL Stream job

Before editing, you must stop the running SQL job. After modifying the properties of the job, you need to execute them again to apply the changes.

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.
3. Select Stopped Jobs from the drop-down menu.
4. Click on the job you want to restart.
   a. You can further filter down the results, by directly searching for the job name in the Search field.
5. Click Edit Selected Job under the **Details** tab.



   The SQL job and its configuration is loaded in the SQL window on the **Compose** page.
6. Edit any configuration of the selected SQL job.

   You need to click on Advanced Settings to display more configuration of the SQL job.
7. Click on Execute.

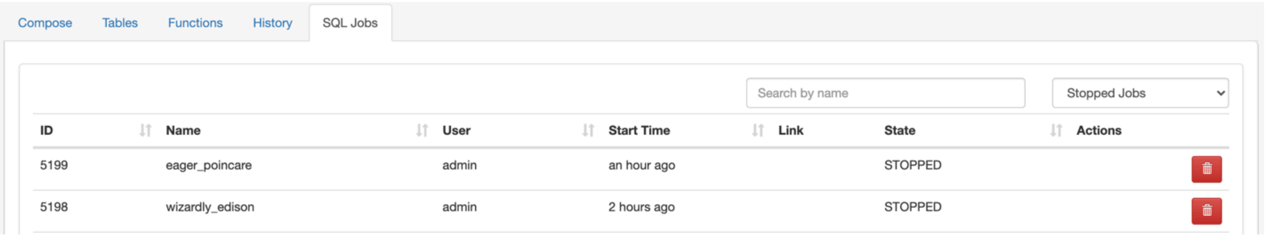
Deleting a SQL Stream job

You can delete a stopped SQL job from SQL Stream Builder on the SQL Jobs tab. By deleting the SQL job, you remove them from the list of stopped jobs.

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.
3. Select Stopped Jobs from the drop-down menu.

   The list of stopped jobs is diplayed on the **SQL Jobs** page.
4. Click on Delete under Actions.



ID	Name	User	Start Time	Link	State	Actions
5199	eager_poincare	admin	an hour ago		STOPPED	🗑
5198	wizardly_edison	admin	2 hours ago		STOPPED	🗑

Sampling data for a running job

You can sample data from a running job. This is useful if you want to inspect the data to make sure the job is running correctly and producing the results you expect.

About this task

Sampling the results to your browser allows you to inspect the queried data and iterate on your query. You can sample 100 rows in the Results tab by clicking on the Sample button in the Console. In case you do not add any sink to the SQL job, the results automatically appear in the Results tab.

Procedure

1. Select Console on the main menu.
2. Go to the SQL Jobs tab.
3. Select the job you want to edit.
4. Go to the Details tab at the bottom.
5. Click Edit Selected Job.

   The SQL window in Edit Mode appears.
6. Click Sample.

Results

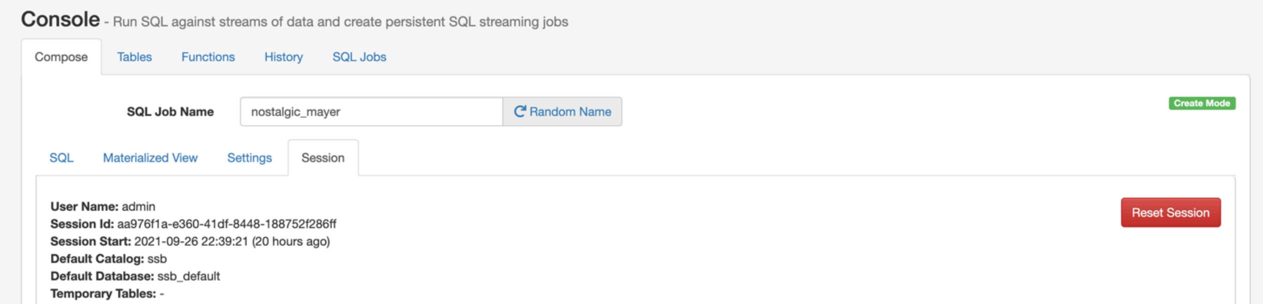
Sample results are displayed in the results window. If there is no data meeting the SQL query, sampling stops after a few attempts.

Managing session for SQL jobs

By default, the SQL Stream jobs are running in a session cluster. This means that multiple Flink jobs run in the same YARN session sharing the cluster, allocated resources, the Job Manager and Task Managers. The session starts when you open the Streaming SQL Console. You can reset the session, and set the properties of the session using the Streaming SQL Console.

Resetting a session

When you reset your session, every configuration, temporary table and view, default database and catalog will be lost.



SQL Job Name: nostalgic_mayer

User Name: admin
Session Id: aa976f1a-e360-41df-8448-188752f286ff
Session Start: 2021-09-26 22:39:21 (20 hours ago)
Default Catalog: ssb
Default Database: ssb_default
Temporary Tables: -

1. Navigate to the **Streaming SQL Console**.

   a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.

   b. Select the Streaming Analytics cluster from the list of Data Hub clusters.

   c. Select Streaming SQL Console from the list of services.

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

2. Select Session on the Console page.
3. Click Reset.

Configuring properties for a session

You can configure the session properties using the SET statement in the SQL window.

1. Navigate to the **Streaming SQL Console**.
    a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
    b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
    c. Select Streaming SQL Console from the list of services.

    The **Streaming SQL Console** opens in a new window.
2. Use the SET statement in the **SQL window** to configure a session property.

    Example:

    ```
    SET state.backend=rocksdb
    ```
3. Click Execute.

Executing SQL jobs in production mode

As by default the SQL jobs are running in a session cluster, there is a risk in case of a cluster failure that every job is affected within that cluster. However, you can set a per-job production mode in SQL Stream Builder to create a dedicated environment for your production jobs.

Production mode means that separately from the running session in SSB, you deploy a SQL job (Flink job) in per-job mode with a dedicated YARN cluster that is configured specifically to that particular production job.

To run your SQL jobs in production mode, you need to add a --PROD prefix to the SQL window at the beginning of your SQL query, and execute the SQL query after setting the execution target and properties in the same window:

```
--PROD
set 'execution.target' = 'yarn-per-job';
set 'logging.configuration.file' = '/tmp/log4j.properties';
select * from datagen_table_1631781644;
```

In the above example, the production mode is indicated as --PROD, and the execution target is set to per-job to create a new YARN application for the job. Setting the execution target to per-job allows you to have an individual cluster for the specific job. The additional properties that you configure using the SET statement overwrites the properties that are configured for the running session. However, when you set properties for the production mode, the settings of the session cluster are not affected.

1. Navigate to the **Streaming SQL Console**.
    a. Navigate to Management Console > Environments , and select the environment where you have created your cluster.
    b. Select the Streaming Analytics cluster from the list of Data Hub clusters.
    c. Select Streaming SQL Console from the list of services.

    The **Streaming SQL Console** opens in a new window.
2. Specify a job name in the SQL Job Name field, or click Random Name.
3. Add --PROD to the **SQL window**.
4. Set the execution mode to per-job.

    ```
    set 'execution.target' = 'yarn-per-job';
    ```

5. Add additional configuration to the production job.

   For the list of configurable parameters, see *Session cluster properties* section.
6. Add a SQL statement you want to execute

   Example:

   ```
   --PROD
   set 'execution.target' = 'yarn-per-job';
   set 'state.backend' = 'rocksdb';
   select * from faker_table_1631781644;
   ```
7. Click Execute.

Migrating SQL jobs

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

You can migrate you SQL jobs either using the API Explorer or the SSB 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 SSB REST API endpoints. However, when using the API Explorer, you need to manual copy and paste the job details between the endpoints. The migration tool in CLI automatically takes care of accessing this information.

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 API Explorer

1. Access the API Explorer.
   a. Click API Explorer on the main menu of Streaming SQL Console.
   b. Go to Cloudera Manager > Cluster > SQL Stream Builder , and click SQLStreamBuilder API Explorer from Web UI.
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-job-migration -api <cluster_hostname> -o job-export -j <jobname> -f
<target_filename>
```

3. Use the following command to import the job:

```
ssb-job-migration -api <cluster_hostname> -o job-import -j <jobname> -f
<source_filename>
```

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

Using SQL Stream Builder REST API

You can use the REST API to monitor, manage and configure the SQL Stream jobs with GET, POST and DELETE HTTP methods. You can use the SQL Stream Builder (SSB) REST API in command line or use a REST API Tool.

The following HTTP methods are available for SSB:

- GET to query information about the specified endpoint
- POST to create resources for the specified endpoint
- DELETE to remove objects from the specified endpoint

The REST API Reference document contains the available endpoints for SQL Stream Builder.

You can also reach the SSB REST API reference document from the following places:

- From Cloudera Manager:
  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. Select Cloudera Manager from the services.

     You are redirected to the **Cloudera Manager** user interface.
  4. Select SQL Stream Builder from the list of services.
  5. Click Web UI.
  6. Select SQLStreamBuilder API Explorer.
- From Streaming SQL Console:
  1. Access Streaming SQL Console.
  2. Click API Explorer from the main menu.

You can use the SSB REST API with Command Line Interface (CLI), but you can also use a REST API Tool, for example Postman.

The Streaming SQL Engine API details the following operations for SQL Stream Builder and Flink:

- SQL Operations: You can execute and analyze the SQL queries
- Session Operations: You can manage and reset the SSB session
- Sampling Operations: You can configure the sampling behavior and retrieve sampling results.
- Job Operations: You can create and stop jobs, and also retrieve job information
- Flink Session Cluster Operations: You can manage and reset the Flink session
- Flink Job Management: You can run flink applications
- Artifact Management: You can add and delete jar files and configuration files, and also retrieve information about them

Using REST API with CLI

When using the SSB REST API with CLI Tool, you need to create the POST or GET commands with curl, and also include the Manager host FQDN of the Data Hub cluster, the name of the Data Hub cluster, and the required operation for the endpoint. If necessary, you can include the Streaming SQL Engine port. The submitted commands return the information, or complete the process you have requested and display the status in the CLI.

The following examples show a secured GET method to list the SQL jobs:

```
curl -ik -u '<workload_username>:<workload_password>' \
    'https://<dh_cluster_manager_host_fqdn>/<dh_cluster_name>/cdp-proxy-api/
ssb-sse-api/api/v1/ssb/jobs'
```

Using REST API Tool

When using REST API Tools, make sure that the base URL in the HTTP request is similar to the following example:

```
https://<dh_cluster_manager_host_fqdn>/<dh_cluster_name>/cdp-proxy-api/ssb-s
se-api/
```

The base path has to be the same for all endpoints. In terms of authentication, make sure you set “Basic Auth”, and use your workload username and password as credentials.

Creating User Defined Functions

With SQL Stream Builder, you can create user functions to write powerful functions in JavaScript, that you can use to enhance the functionality of SQL.

About this task

User functions can be simple translation functions like Celsius to Fahrenheit, more complex business logic, or even looking up data from external sources. User functions are written in JavaScript. When you write them, you create a library of useful functions.

Procedure

1. Navigate to the Streaming SQL Console.
    a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
    b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
    c) Select Streaming SQL Console from the list of services.

    The **Streaming SQL Console** opens in a new window.
2. Select Console from the left-hand menu.
3. Select Functions > Add .
4. Name the function HELLO_WORLD, give it a short description, select JavaScript as the language.
5. Select STRING as output type and add STRING to the input type by selecting it from the list and clicking the plus button.
6. Paste this code into the JavaScript editor:

```
// check to see if the card is VISA
function HELLO_WORLD(card){
    var cardType = "Other";
    if (card.charAt(0) == 4){
      cardType = "Visa";
    }
    return cardType;
}
HELLO_WORLD($p0); // this line must exist
```

7. Click Save.
8. Once created, you can use a User Function in your SQL statement:

```
-- simple usage
SELECT HELLO_WORLD(card) AS IS_VISA
FROM ev_sample_fraud;

-- in the predicate
SELECT amount, card
FROM ev_sample_fraud
WHERE HELLO_WORLD(card) = "Visa";
```

Note: Valid inputs can be a field in the source virtual table or any other valid input. Functions must be in upper case.

Note: User Functions have access to the Java 8 API, this increases the overall usefulness and power. For example:

```
function GETPLANE(icao) {
  try {
    var c = new java.net.URL('http://yyyyyy.io' + icao).openConnec
tion();
    c.requestMethod='GET';
    var reader = new java.io.BufferedReader(new java.io.InputStreamRe
ader(c.inputStream));
    return reader.readLine();
  } catch(err) {
    return "Unknown: " + err;
  }
}
GETPLANE($p0);
```

Developing JavaScript functions

When developing JavaScript functions that are more complicated than just simple logic, it is recommended to use the jjs command-line utility to create and iterate while writing functions.

About this task

After the function performs the required task, migrate it to the console. Additionally these files/functions can be saved in a source code control system like git/Github.

Procedure

1. Create a file for your function.

   Note: It is recommended to name the file with the same name as that of the function.

2. Create some sample input when calling the function.
3. Call jjs on the command line to test the function.

```
$>cat TO_EPOCH.js
function TO_EPOCH(strDate) {
   var strFmt = "yyyy-MM-dd HH:ss:mm";
   var c = new java.text.SimpleDateFormat(strFmt).parse(strDate).getTime()
/1000;
   return c.toString();
}

print(TO_EPOCH("2019-02-02 22:23:13"));

then
$>jjs TO_EPOCH.js
1549167203
```

What to do next

After you have successfully developed the JavaScript code, copy and paste only the function to your code window when creating the JavaScript function in SQL Stream Builder.

Adding Java to the Functions language option

You can create User Defined Functions (UDF) using Java after manually adding the UDF function JAR file that contains the UDF class to the Flink connectors. After creating the function in the SQL window, you can select Java as a language for the UDFs.

Procedure

1. Create a Java UDF function JAR file based on the following example:

   ```
   package udf;
   import org.apache.flink.table.functions.ScalarFunction;

   public class UdfTest extends ScalarFunction {
      public String eval(String input){
          return "Hello World " + input;
      }
   }
   ```

   Note: The function needs to be named as ‘eval’ in the JAR file.

2. Copy the JAR file to the flink connectors folder:

   ```
   scp <jar_location>/<jar_filename> <workload_username>@<datahub_hostname>:/
   usr/share/flink-connectors
   ```

3. Navigate to the Streaming SQL Console.
   a) Navigate to Management Console > Environments , and select the environment where you have created your cluster.
   b) Select the Streaming Analytics cluster from the list of Data Hub clusters.
   c) Select Streaming SQL Console from the list of services.

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

4. Run the following query to create the function:

   ```
   CREATE FUNCTION myFunction AS 'udf.UdfTest’ LANGUAGE java;
   ```

5. Test the function by running a SELECT query.

   ```
   SELECT myFunction('test');
   ```

Monitoring SQL Stream jobs

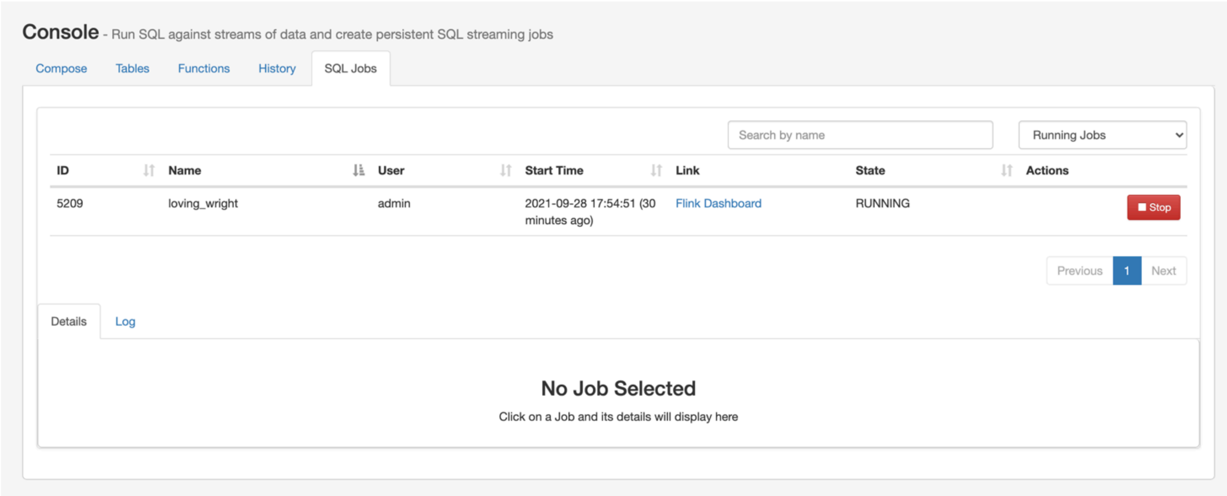
You can use the Streaming SQL Console to review the status, properties and log of your SQL Stream jobs executed in SQL Stream Builder. Using the Flink Dashboard, you can also monitor the Flink job that is submitted when you execute a SQL query.

Using the Streaming SQL Console

When using the **SQL Jobs** page in Streaming SQL Console to monitor your SQL jobs, you can review the ID, the Name, the Start time, the State of the submitted jobs, and the User who submitted the SQL jobs. When monitoring running jobs, you are also able to open the Flink Dashboard, and stop the job using the Stop button under Actions. The Flink Dashboard link and Stop button are not available for Stopped Jobs.

1. Click Console on the main menu.
2. Select SQL Jobs tab.

   By default, Running Jobs are displayed on the **SQL Jobs** page.



Console - Run SQL against streams of data and create persistent SQL streaming jobs

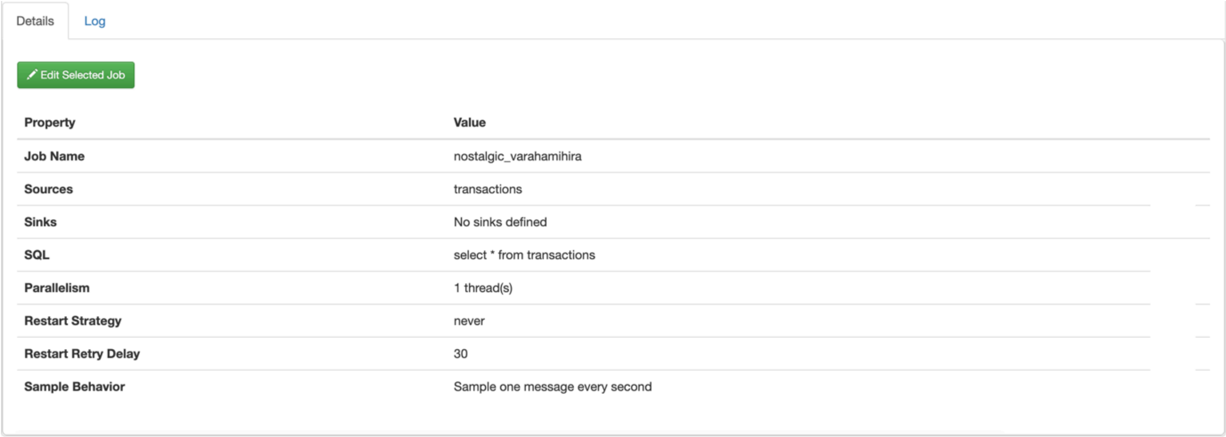
ID	Name	User	Start Time	Link	State	Actions
5209	loving_wright	admin	2021-09-28 17:54:51 (30 minutes ago)	Flink Dashboard	RUNNING	Stop

No Job Selected

Click on a Job and its details will display here

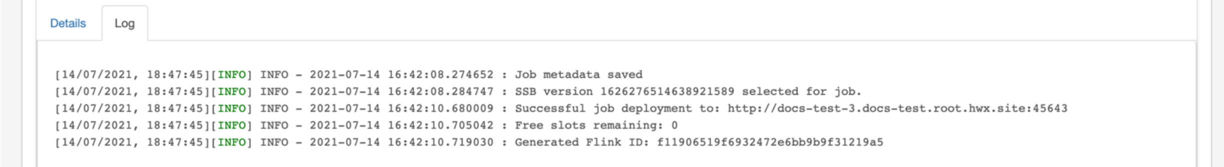
3. Select Running Jobs or Stopped Jobs from the drop-down menu.
4. Click on the job you want to monitor.

   a. You can further filter down the results, by directly searching for the job name in the Search field.
5. Select Details tab on the bottom panel to display additional details and configurations about the SQL job.



Property	Value
Job Name	nostalgic_varahamihira
Sources	transactions
Sinks	No sinks defined
SQL	select * from transactions
Parallelism	1 thread(s)
Restart Strategy	never
Restart Retry Delay	30
Sample Behavior	Sample one message every second

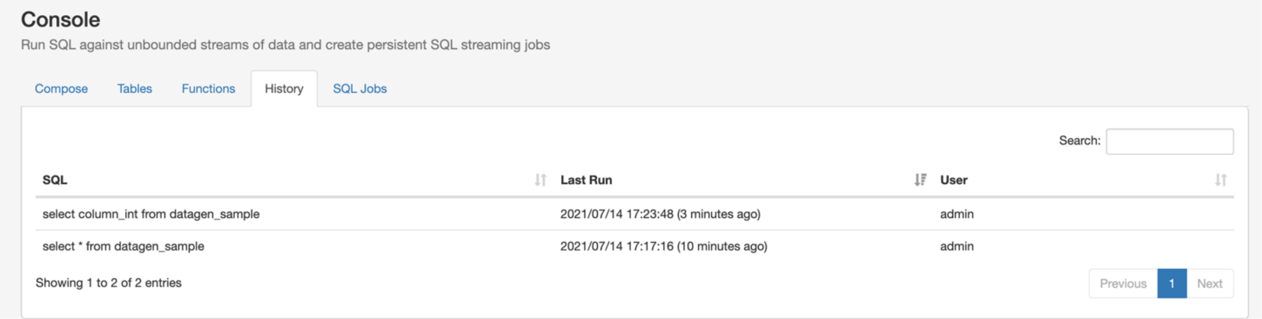
6. Select Log tab on the bottom panel to review the log information of the job submission.



```
[14/07/2021, 18:47:45][INFO] INFO - 2021-07-14 16:42:08.274652 : Job metadata saved
[14/07/2021, 18:47:45][INFO] INFO - 2021-07-14 16:42:08.284747 : SSB version 1626276514638921589 selected for job.
[14/07/2021, 18:47:45][INFO] INFO - 2021-07-14 16:42:10.680009 : Successful job deployment to: http://docs-test-3.docs-test.root.hwx.site:45643
[14/07/2021, 18:47:45][INFO] INFO - 2021-07-14 16:42:10.705042 : Free slots remaining: 0
[14/07/2021, 18:47:45][INFO] INFO - 2021-07-14 16:42:10.719030 : Generated Flink ID: f11906519f6932472e6bb9b9f31219a5
```

History of SQL queries

You can review and reuse the SQL queries that were previously executed on the **History** page. When you click on one of the SQL queries, it is automatically imported to the SQL window for execution. You can filter the SQL queries by the time they were last run or by the user who run them. You can also search for a type of SQL query using the Search field.

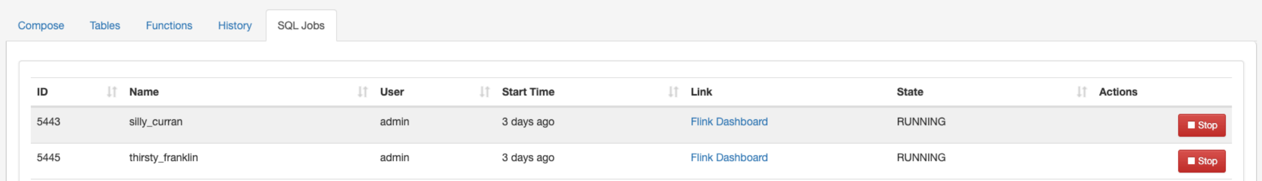

Console
Run SQL against unbounded streams of data and create persistent SQL streaming jobs
Compose
Tables
Functions
History
SQL Jobs
Search:

SQL	Last Run	User
select column_int from datagen_sample	2021/07/14 17:23:48 (3 minutes ago)	admin
select * from datagen_sample	2021/07/14 17:17:16 (10 minutes ago)	admin

Showing 1 to 2 of 2 entries

Using the Flink Dashboard

You can also monitor your running SQL jobs using the Flink Dashboard. You can easily reach the Flink Dashboard on the SQL jobs tab below the Console main menu. After clicking on the Flink Dashboard, you are redirected to the Flink Dashboard interface.



ID	Name	User	Start Time	Link	State	Actions
5443	silly_curran	admin	3 days ago	Flink Dashboard	RUNNING	Stop
5445	thirsty_franklin	admin	3 days ago	Flink Dashboard	RUNNING	Stop

SQL Syntax Guide

The SQL Syntax Guide provides a collection of SQL syntaxes that is supported in SQL Stream Builder.

SQL Syntax

- Filtering WHERE
- Projections FOO || 'baz'
- Tumble, Hopping window expressions TUMBLE(), HOP()
- Grouping GROUP BY
- Filter after aggregation HAVING
- Join JOIN ON..
- Comparison operators
- Logical operators
- Arithmetic operators and functions
- Character string operators and functions

- Binary string operators and functions
- Date/Time functions
- Type conversion
- Aggregate functions
- Grouped window functions
- Grouped Auxiliary functions

Data types

- Datatypes

Metadata commands

- show tables - list virtual tables.
- desc <vtable> - describe the specified virtual table, showing columns and types.
- show jobs - list current running SQL jobs.
- show history - show SQL query history (only successfully parsed/executed).
- help - show help.