

SQL Stream Job Lifecycle

Date published: 2019-12-17

Date modified: 2022-02-28



Legal Notice

© Cloudera Inc. 2024. 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

Running SQL Stream jobs.....	4
Configuring SQL job settings.....	6
Adjusting logging configuration in Advanced Settings.....	6
Stopping, restarting and editing SQL jobs.....	7
Sampling data for a running job.....	12
Managing session for SQL jobs.....	12
Executing SQL jobs in production mode.....	13

Running SQL Stream jobs

Every time you run an SQL statement in the SQL Stream console, it becomes a job and runs on the deployment as a Flink job. You can manage the running jobs using the Jobs tab on the UI.

About this task

There are two logical phases to run a job:

1. **Parse:** The SQL is parsed and checked for validity and then compared against the virtual table schema(s) for correct typing and key/columns.
2. **Execution:** If the parse phase is successful, a job is dynamically created, and runs on an open slot on your cluster. The job is a valid Flink job.

Before you begin

- Make sure that you have registered a Data Provider if you use the Kafka service on your cluster.
- Make sure that you have added Kudu, Hive or Schema Registry as a catalog if you use them for your SQL job.

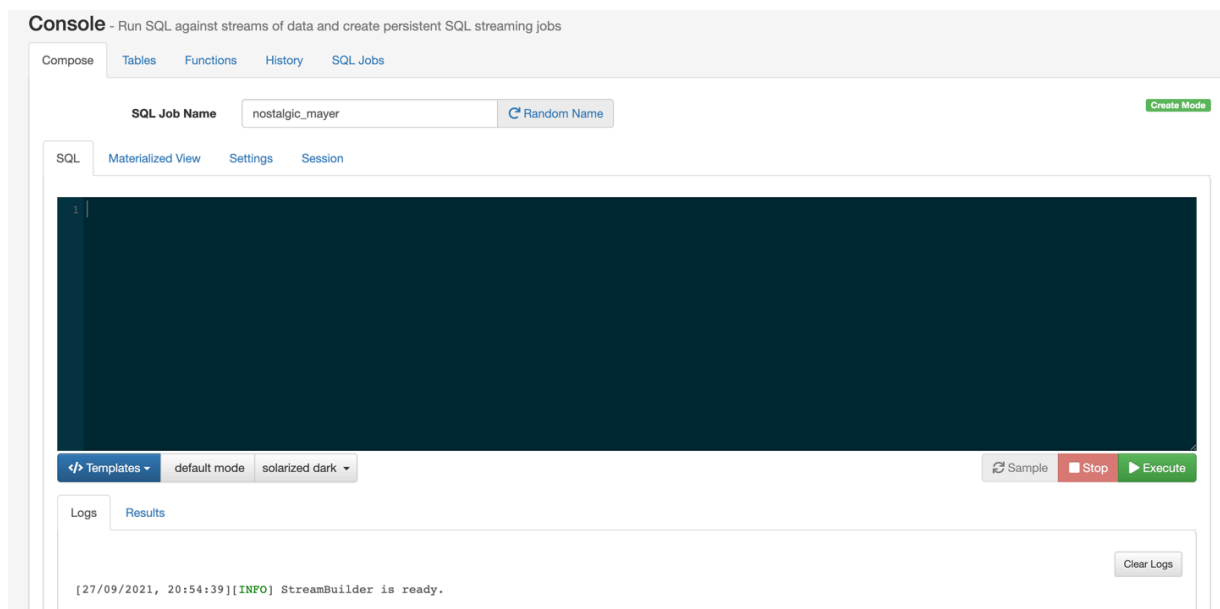
Procedure

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 **Web UI > SQLStreamBuilder Console**.The **Streaming SQL Console** opens in a new window.
2. Provide a name for the SQL job.
 - a) Optionally, you can click **Random Name** to generate a name for the SQL job.
3. Create a table in SQL window.

You have the option to create a table in the following ways:

 - Using the Console wizard under **Tables** tab to add Kafka and Webhook tables.
 - Using the **Templates** under the SQL window.
 - Add your custom **CREATE TABLE** statement to the SQL window.

4. Add a SQL query to the SQL window.

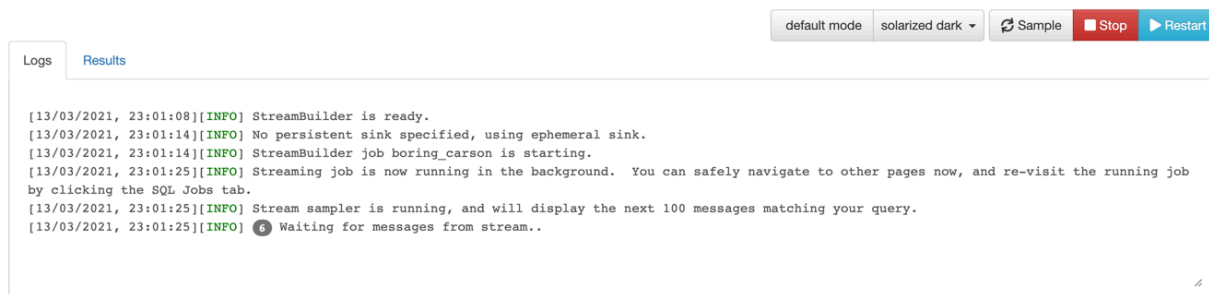


Note: You cannot start a job without adding a SQL statement in the SQL editor window. In case, there are no SQL statements provided and you click Execute, the following error message is displayed: You must provide a SQL query.

When starting a job, the number of slots consumed on the specified cluster is equal to the parallelism setting. The default is one slot. To change the parallelism setting and more job related configurations, click Settings.

5. Click Execute.

The Logs window updates the status of SSB.



6. Click Results to check the sampled data.

These results are only samples, not the entire result of the new stream being created from the output of the query. The entire result set is written out based on what you define at the INSERT INTO statement. You can also output the result to a Materialized View.



Note: While SSB is querying unbounded data streams, you can click Stop to stop the job execution.

Results

A job is generated that runs the SQL continuously on the stream of data from a table, and pushes the results to a table, to the Console under the Results tab or to a Materialized View.

Related Information

[Registering Data Providers in SSB](#)

[Creating Tables for SQL Stream jobs](#)

[Monitoring SQL Stream jobs](#)

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, exactly 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.

The screenshot shows the 'Settings' tab of the SQL job configuration window. It contains the following settings:

- Job Parallelism (threads):** A text input field with the value '1'.
- Sample Count:** A text input field with the value '100'.
- Sample Window Size:** A text input field with the value '100'.
- Sample Behavior:** A dropdown menu with the selected option 'Sample one message every second'.
- Restore From Savepoint:** A dropdown menu with the selected option 'false'.

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 provided 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.

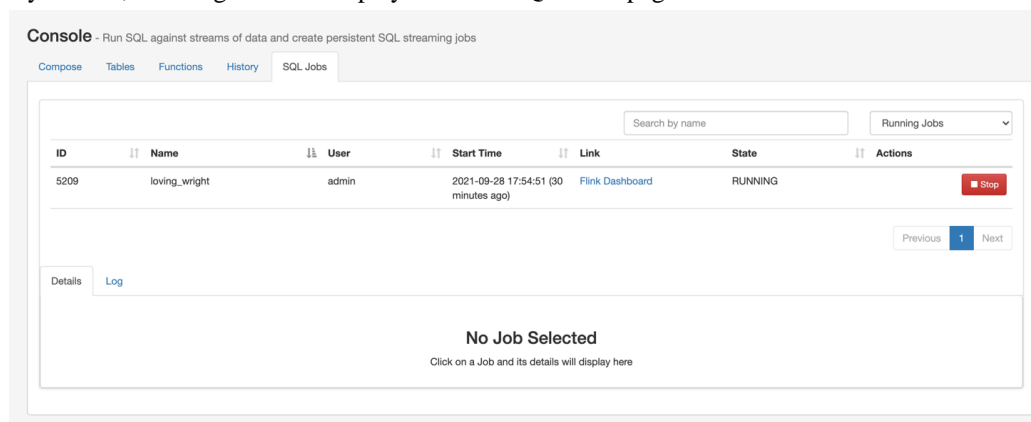
- Click Stop under the SQL window.



Stopping job from SQL Jobs page

- Click Console on the main menu.
- Select SQL Jobs tab.

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



- Click on the job you want to stop.
 - You can further filter down the results, by directly searching for the job name in the Search field.
- 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.



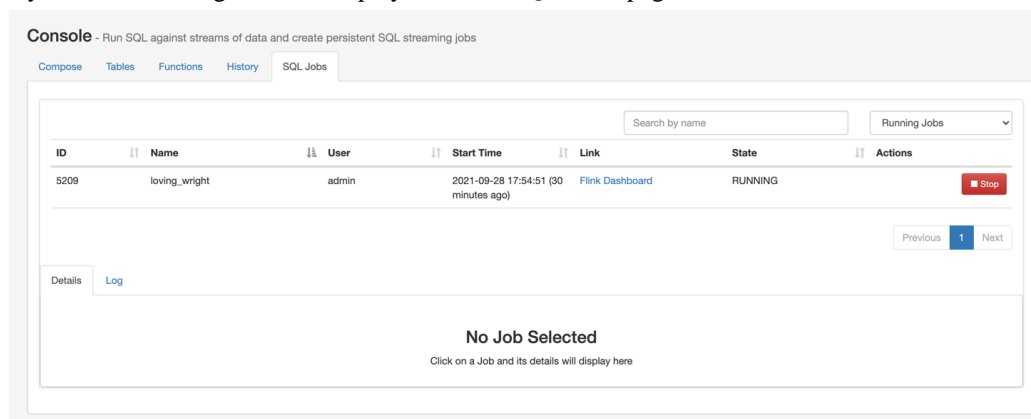
In case the job is running in the background, you can load it with its properties from the SQL Jobs tab.

Restarting job from SQL Jobs page

- Click Console on the main menu.

2. Select SQL Jobs tab.

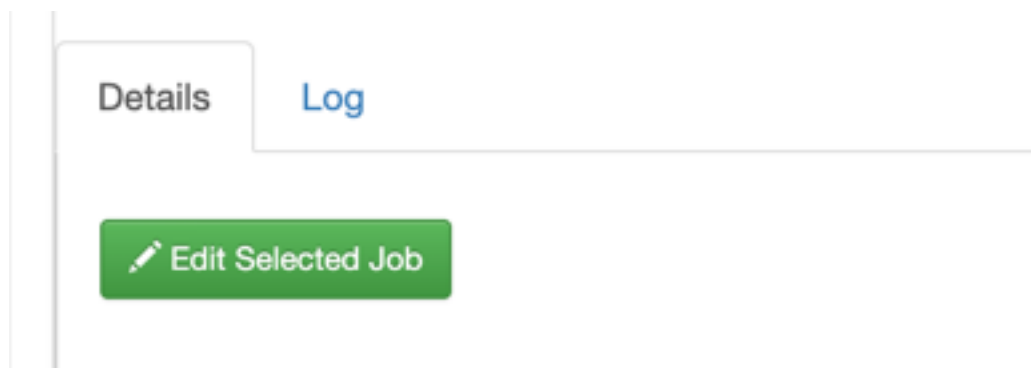
By default, Running Jobs are displayed on the **SQL Jobs** page.



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.

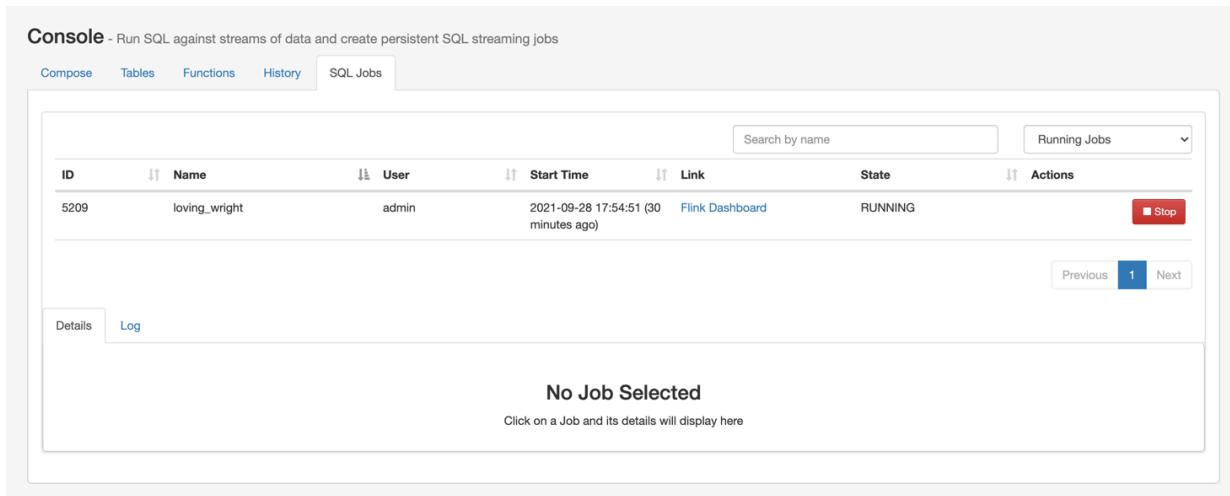
Restarting a stopped SQL Stream job

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.

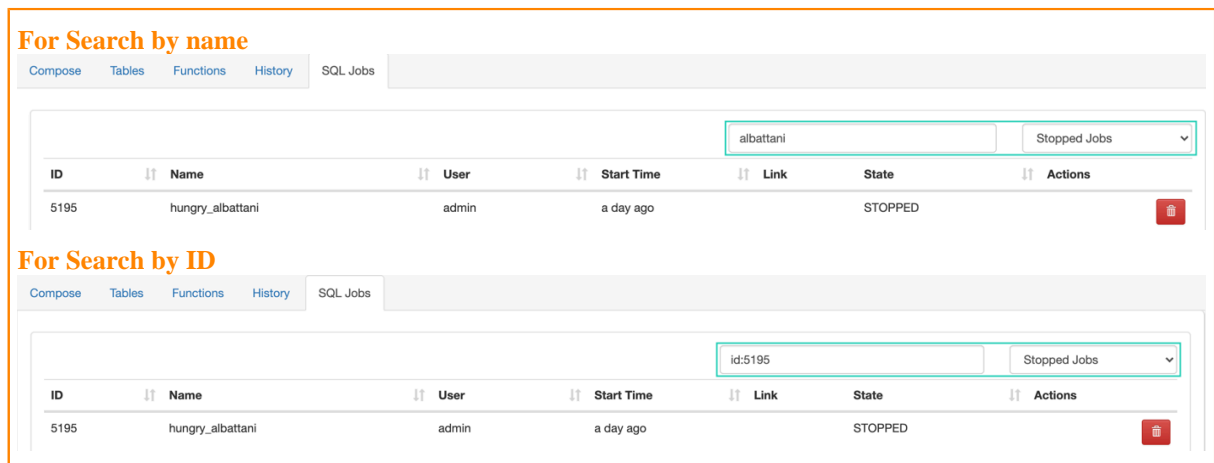
By default, Running Jobs are displayed on the **SQL Jobs** page.



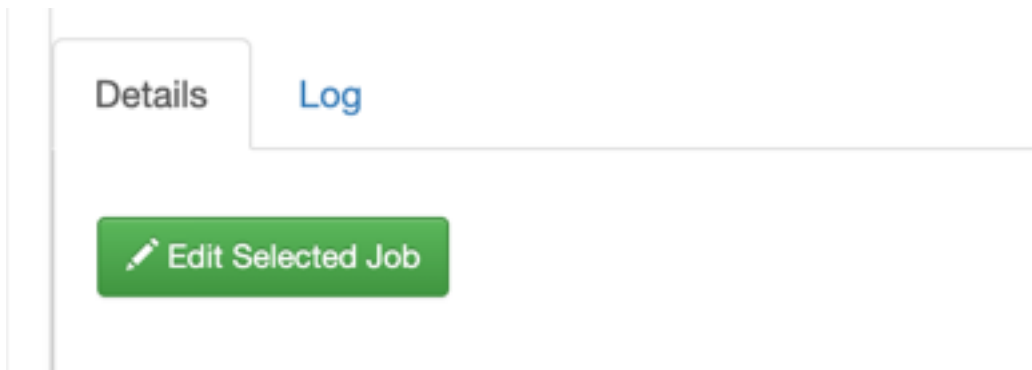
3. Select Stopped Jobs from the drop-down menu.

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.



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.

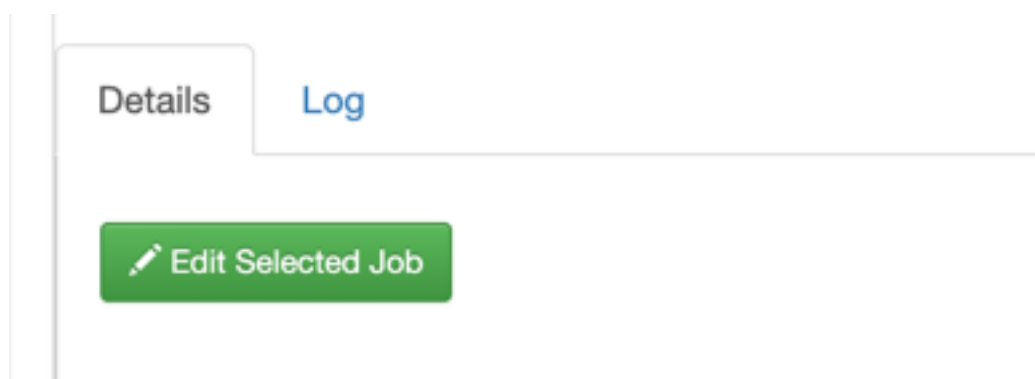
Editing a SQL Stream job

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 displayed on the **SQL Jobs** page.

4. Click on Delete under Actions.

Compose Tables Functions History SQL Jobs							
				Search by name		Stopped Jobs	
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

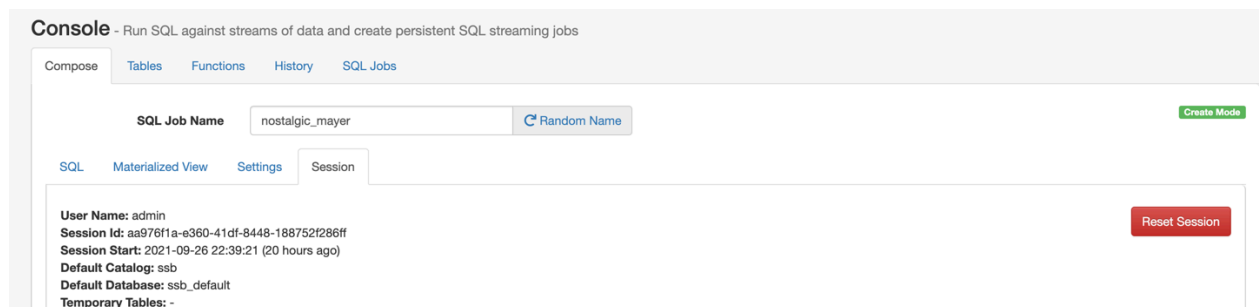
Sample results are displayed in the results window. If there is no data meeting the SQL query, sampling stops after a few attempts.

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.



1. Navigate to the **Streaming SQL Console**.
 - a. Go to your cluster in Cloudera Manager.
 - b. Click on SQL Stream Builder from the list of Services.
 - c. Click on the SQLStreamBuilder Console.

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. Go to your cluster in Cloudera Manager.
 - b. Click on SQL Stream Builder from the list of Services.
 - c. Click on the SQLStreamBuilder Console.

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. Go to your cluster in Cloudera Manager.
 - b. Click on SQL Stream Builder from the list of Services.
 - c. Click on the SQLStreamBuilder Console.

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.