

Getting Started with Cloudera SQL Stream Builder

Date published:

Date modified:

The Cloudera logo, featuring the word "CLOUDERA" in a bold, orange, sans-serif font. The letter "E" is stylized with a horizontal bar through its center.

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

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

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

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

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

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

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

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

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

Contents

Submitting a SQL job.....	4
Extensibility.....	5
Managing project files.....	6
Iceberg tables and connectors in Cloudera SQL Stream Builder.....	8

Submitting a SQL job

Sign in to Cloudera SQL Stream Builder and run a new job to confirm launching Flink jobs.

What is Cloudera SQL Stream Builder?

The Cloudera SQL Stream Builder is a comprehensive interactive user interface for creating stateful stream processing jobs using SQL. By using SQL, you can simply and easily declare expressions that filter, aggregate, route, and otherwise mutate streams of data. Cloudera SQL Stream Builder is a job management interface that you can use to compose and run SQL on streams, as well as to create durable data APIs for the results.

For more information on Cloudera SQL Stream Builder, see [Cloudera SQL Stream Builder Overview](#).



Note: This documentation details the differences in setup and operation of Cloudera SQL Stream Builder for Cloudera Streaming Analytics Operator for Kubernetes. For detailed information on using Cloudera SQL Stream Builder, see [Using Cloudera SQL Stream Builder](#).

To access the User Interface (UI) of Cloudera SQL Stream Builder, called Streaming SQL Console, run the following command to set up the port-forwarding:

```
kubectl -n [*** NAME SPACE ***] port-forward service/ssb-sse 18121:18121
```

This enables access to Streaming SQL Console at <http://localhost:18121> (or your machine's IP address if it differs from localhost).



SSB Login

Username *

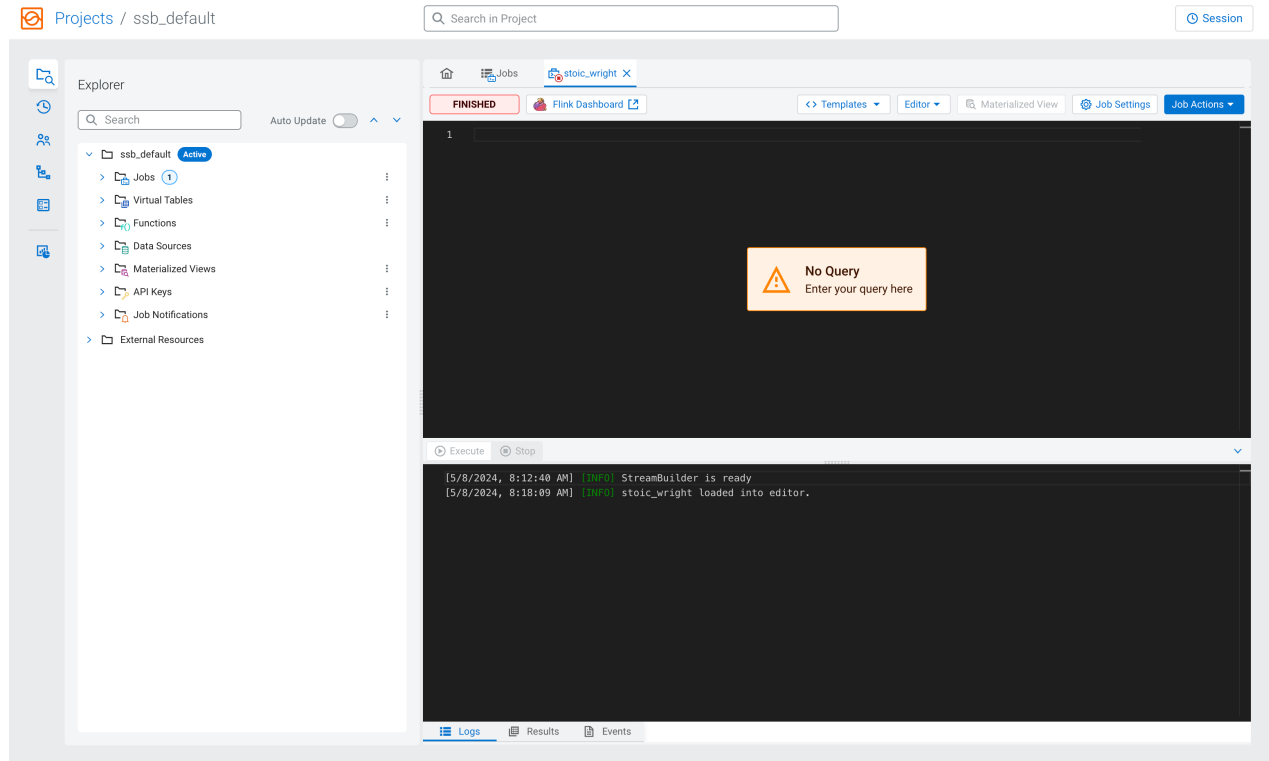
Password *

Sign In

Don't have an account? [Create new account](#)

You can sign in to Streaming SQL Console using the default username and password (admin - admin), or you can register with a new account.

After logging in, you are redirected to the Projects homepage of Streaming SQL Console. To start a simple SQL job, open the `ssb_default` project. Click  to create a new job with a randomly generated name or you can provide your own.



Copy and paste the following SQL query to the SQL Editor, and click Execute:

```
CREATE TABLE blackhole (col1 STRING) WITH ('connector'='print');
CREATE TABLE datagen (col1 STRING) WITH ('connector'='datagen');
INSERT INTO blackhole SELECT * FROM datagen;
```

This job generates random rows and writes the generated data to the TaskManager logs. You can use this example to verify that SSB can successfully launch a Flink job.

Extensibility

Extend Cloudera SQL Stream Builder on Cloudera Streaming Analytics Operator for Kubernetes with project files, custom connector JARs, and optional container images.

Project files

Upload JARs, TLS certificates, configuration directories, and other artifacts as project files and reference them from Flink SQL DDL and connector definitions. Configure `ssb.storageConfiguration` in the Helm chart so uploaded files are staged for Flink job pods. For procedures, see [Managing project files](#).

Bundled connectors

The default Cloudera Streaming Analytics Operator for Kubernetes deployment uses Flink and Cloudera SQL Stream Builder images that include Kafka, JDBC, Hive, Kudu, Iceberg, filesystem connectors, and other bundled connectors. You register tables with Flink SQL DDL and console templates.

Custom connectors and optional images

To use connectors or libraries that are not in the default image, upload JARs as project files or build optional connector images as described in [Customizing container images](#). Register custom tables with Flink SQL DDL.

Mounted configuration and credentials

You can still mount Hadoop configuration, Kerberos keytabs, and other files with podVolumes and podVolumeMounts in the Helm chart when you need cluster-wide paths on Flink or Cloudera SQL Stream Builder pods. See [SSB service configurations](#).

Managing project files

Upload, organize, and reference project-scoped files such as TLS certificates, connector JARs, and configuration bundles from Cloudera SQL Stream Builder.

About this task

Project files are artifacts stored with a project. You can upload certificates, Hive or Iceberg configuration directories, connector JARs, and other files that Flink jobs and Cloudera SQL Stream Builder resources need at runtime. Files appear under the Files node in the Project Explorer.

Configure remote artifact storage in the Helm chart with `ssb.storageConfiguration` so uploaded files are available to Flink job pods. See [SSB service configurations](#) and [Extensibility](#).

You can create folders only at the project root. Nested folders are not supported. Upload one or more files at a time to the root or to an existing folder. A file at the project root cannot use the same name as a folder at the root, and vice versa.

Procedure

1. Open a project in Cloudera SQL Stream Builder.
2. In the Project Manager, select Explorer.
3. Expand Files.

Create a folder

4. On the Files node, open the action menu and click Create Folder.

The Create Folder dialog opens.

5. Enter a Folder Name and click Create.

Folder names cannot contain slashes. The new folder appears under Files.

Upload files

6. On the Files node or on a folder under Files, open the action menu and click Upload Files.

The Upload Files dialog opens.

7. Optional: In Folder, choose Root or an existing folder.

When you start upload from a folder node, that folder is selected by default.

8. Click Select Files, choose one or more files, and click Upload.

Uploaded files appear in the Explorer tree under Files or inside the target folder.

Delete files or folders

9. On a file or folder under Files, open the action menu and click Delete.

Confirm the deletion when prompted. Deleting a folder removes the folder and its files from the project.



Warning: If a running job or data source still references a deleted file, update those resources before you delete the file.

What to do next

Reference project files in SQL and configuration properties

Some connector and catalog properties accept a project file token instead of a fixed path on the Flink image. In the console UI, use the following format:

- File at the project root: `ssb.project-file:FILENAME`
- File inside a folder: `ssb.project-file:FOLDERNAME/FILENAME`
- Project folder (for example a Hive configuration directory uploaded as a folder): `ssb.project-file:FOLDERNAME`

At job submission, Cloudera SQL Stream Builder resolves the token to the staged project file path for the Flink job. This pattern is commonly used for properties such as `hive-conf-dir` on Iceberg catalogs when the Hive configuration directory is uploaded as a project folder.

In `CREATE TABLE`, `CREATE VIEW`, and catalog DDL, use a string literal of the form `'${ssb.project-file:PATH}'`. The `PATH` follows the same rules as in the console (root file, file in a folder, or a configuration folder). You can also use these placeholders in Flink job configuration properties where supported.

The following example creates a Kafka table that references a TLS truststore at the project root:

```
CREATE TABLE kafka_ssl (
  id STRING,
  payload STRING
) WITH (
  'connector' = 'kafka',
  'topic' = 'events',
  'properties.ssl.truststore.location' = '${ssb.project-file:client.truststore.jks}',
  'properties.security.protocol' = 'SSL'
);
```

The following example registers an Iceberg catalog that uses a Hive configuration folder uploaded as `hive-conf`:

```
CREATE CATALOG iceberg_hive WITH (
  'type' = 'iceberg',
  'catalog-type' = 'hive',
  'hive-conf-dir' = '${ssb.project-file:hive-conf}'
);
```

What to do next

Select project files in the UI

Path fields in selected resource editors support either a project file or a filesystem path on the Flink image. You cannot combine both for the same field.

When a selector is available, choose an uploaded project file from the list. To use a file that is already on the Flink image, enter the full path in the text field instead.

Kafka data sources expose this pattern for TLS material when custom truststore settings are enabled:

- Kafka TrustStore
- Kafka KeyStore (Optional)

Similar selectors appear on other path properties that reference project files. When you upgrade from an earlier Cloudera SQL Stream Builder release, existing path-based values are preserved until you change them to a project file.

Iceberg tables and connectors in Cloudera SQL Stream Builder

The Iceberg connector and catalog in Cloudera SQL Stream Builder support Kubernetes deployments, including Kafka to Iceberg upsert pipelines.

Prerequisites

Iceberg, Hive, Hadoop, and Kudu connectors ship in the alternative Flink image. Enable it during install as described in *Installation overview*. Cloudera SQL Stream Builder exposes Iceberg as a first-class catalog type under Data Sources and Catalogs.

Kafka to Iceberg live upsert

Streaming upsert into Iceberg tables requires exactly-once checkpointing. Without checkpointing, the Iceberg sink cannot commit incremental files reliably and the job may fail or produce duplicates.

Enable checkpointing in the Flink configuration for the job, for example:

```
execution.checkpointing.interval: 60s
state.checkpoints.dir: s3://<bucket>/checkpoints/
state.savepoints.dir: s3://<bucket>/savepoints/
```

Configure durable checkpoint storage through `ssb.storageConfiguration` or per-job Advanced Settings before you start a Kafka to Iceberg upsert pipeline.

Iceberg table and catalog properties

When you create an Iceberg table in Cloudera SQL Stream Builder, set catalog and table options that match your object store and metastore:

- For S3-backed catalogs, use the alternative image and ensure Hadoop S3A settings are available through `storageConfiguration` or mounted `core-site.xml`.
- Specify the Iceberg catalog type (for example Hive or Hadoop) and warehouse location in the catalog DDL.
- For upsert mode, define primary key columns and set the Iceberg write mode to upsert in the table DDL or sink configuration.

Iceberg catalogs that use s3a paths resolve the filesystem implementation correctly when the alternative image is enabled.