

Cloudera Streaming Analytics Operator 1.4.0

Cloudera SQL Stream Builder Resource Management

Date published: 2024-06-15

Date modified: 2025-10-24

The Cloudera logo is displayed in a bold, orange, sans-serif font. The word "CLOUDERA" is written in all caps, with a stylized 'E' that has a horizontal bar extending to the right.

<https://docs.cloudera.com/>

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

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

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

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

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

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

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

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

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

Contents

Managing session cluster state.....	4
SSB service configurations.....	4
Security configurations.....	5
Database configuration.....	8
Customize container images.....	8

Managing session cluster state

Get information about existing session clusters, and terminated them from the Cloudera SQL Stream Builder UI.

Cloudera SQL Stream Builder displays information about existing session clusters, and allows for those clusters to be terminated from the UI.

1. After logging in to Cloudera SQL Stream Builder, use the Sessions button in the top right corner of the UI.
2. Click on Session Cluster.
3. If a session cluster is running, the dialog window will show its state, and you can click Stop to terminate it.

If no sessions clusters are running, the dialog will display Session cluster is not running. Execute a job in session-mode to start the session cluster.

SSB service configurations

Configuration variables for the Cloudera SQL Stream Builder.

The following configurations can be set in the values.yaml of the Cloudera Streaming Analytics Operator for Kubernetes:

Configuration	Description
fernetKey	A 32-byte long string that is used to encrypt sensitive data, such as credentials and SQL queries.
userManagement	Enable and disable user registration, and configure default users to be registered automatically.
rbac	Use this if you decide not to use default RBAC resources to create your own before installing the Cloudera Streaming Analytics Operator for Kubernetes.
service	The Service created for SSB-SSE. By default it is set to ClusterIP type.
ingress	Specification of the Ingress to be created.
storageConfiguration	Connection configuration to various storage solutions (for example S3). This data is stored as a KubernetesSecret, since it can contain sensitive values. Use this configuration to set up durable storage for Flink (for example, to create checkpoints). You can also use this configuration for artifact storage for Cloudera SQL Stream Builder will to store UDFs and custom connectors.
podVolumes	Customized volume configuration for both Cloudera SQL Stream Builder and Flink pods. You can specify ConfigMaps, Secrets to be mounted in these Pods. This configuration can be useful if you want to mount Hadoop configurations, such as hdfs-site.xml, core-site.xml or Kerberos keytabs.
podVolumeMounts	Mount point configuration for the Volumes created with podVolumes.
ssbConfiguration	Configuration overrides for SSB to set up remote artifact storage, job notifications, and sampling Kafka.
flinkConfiguration	Path of a default flink-conf.yaml that will be applied to all Flink deployments created by Cloudera SQL Stream Builder.

Configuration	Description
database	By default, Cloudera SQL Stream Builder will create a Postgres database, but you can override it with this property if you already have a database you wish to use with Cloudera SQL Stream Builder data. Supported database types are MySQL/MariaDB, Oracle, and PostgreSQL. (See <i>Cloudera SQL Stream Builder database support</i>)

Security configurations

Configuring Flink for securing your workloads and the Cloudera SQL Stream Builder.

Cloudera SQL Stream Builder enables you to deploy the Flink SQL jobs in an easier, however you need to ensure the proper security of your workloads and Cloudera SQL Stream Builder. This means that the following tools are available for you to secure your `FlinkDeployment` in your Kubernetes cluster setup.

Fernet key

When deploying Cloudera SQL Stream Builder, you must specify a `fernetKey`, which will be used for encrypting the job definition for all Flink jobs started with Cloudera SQL Stream Builder. The job definitions may include sensitive data, such as the DDL of tables that can contain username and password as part of the connector configuration or authentication information for connected storages. Sensitive information will be encrypted by the Fernet key.

The specified Fernet key is created as a Kubernetes Secret in the same namespace where the Cloudera Streaming Analytics Operator for Kubernetes is installed, and will be automatically mounted by Cloudera SQL Stream Builder and Flink pods.

User management

When installing Cloudera SQL Stream Builder, a default user (`admin/admin`) is created automatically, and registration is enabled. You have the option to enable or disable user registration, and you can also modify the default user(s) based on your requirements.

LDAP authentication

To enable LDAP, create a secret with the parameters of the LDAP connection before starting helm install.

Example for secure setup with LDAP:

```
kubectl create secret generic ssb-ldap -n flink \
  --from-literal=SSB_LDAP_URL=ldaps://ldap.example.com:636 \
  --from-literal=SSB_LDAP_BASE_DN=dc=example,dc=com \
  --from-literal=SSB_LDAP_USERNAME=cn=admin,dc=example,dc=com \
  --from-literal=SSB_LDAP_PASSWORD=password \
  --from-literal=SSB_LDAP_USER_DN_PATTERNS=uid={0},ou=people \
  --from-literal=SSB_LDAP_USER_SEARCH_BASE= \
  --from-literal=SSB_LDAP_USER_SEARCH_FILTER= \
  --from-literal=SSB_LDAP_GROUP_SEARCH_BASE= \
  --from-literal=SSB_LDAP_GROUP_SEARCH_FILTER= \
  --from-file=ldap_truststore.jks=[** YOUR PATH **]/truststore.jks \
  --from-literal=SSB_LDAP_SSL_TRUSTSTORE_PASSWORD=changeit
```

In the `values.yaml` file, set `ldap` to true, specify your secret, and list the usernames for the admin users.

```
ssb:
  userManagement:
    type: ldap
    ldap:
      secure: true
```

```
secretRef: ssb-ldap
admins:
  - admin
```

Ingress

By default Cloudera SQL Stream Builder does not set up any Ingress. This can be changed using the ingress configuration. The Ingress resource is created in the same namespace as Cloudera SQL Stream Builder.

Ingress can be used to easily enable TLS/HTTPs to Cloudera SQL Stream Builder, but it can also be used to set up authentication. For more information see *Ingress*.

Storage configuration

It is recommended to add some kind of persistent data storage for Flink to be able to save checkpoints and savepoints. In most cases this is some kind of blob storage (for example, S3) that needs authentication to access.

You can use the storageConfiguration configuration to set up the storage for Cloudera SQL Stream Builder. The configuration should be a valid flink-conf.yaml file, which can contain sensitive data, such as s3.access-key, s3.secret-key, and so on.

The Helm chart creates a Secret in the same namespace as Cloudera SQL Stream Builder, and Cloudera SQL Stream Builder creates a new Secret for each new Flink deployment created by the user.

Volume mounts

It is possible to mount existing volumes to Cloudera SQL Stream Builder and all created Flink pods using the podVolumes and podVolumeMounts configurations. You need to ensure that these volumes exist in the namespace of the Cloudera SQL Stream Builder and Flink pods that will be created by SSB.

The configurations can be used to mount ConfigMaps, Secret, or any kind of volumes to the SSB and Flink pods. For example, you can mount hive-site.xml, core-site.xml, krb5.conf and some keytabs as ConfigMaps and Secrets to be able to connect to Hive with SSB/Flink.

You can also configure custom truststores using a Kubernetes Secret.

Example:

```
kubectl -n flink create secret generic custom-truststore \
  --from-file=truststore.jks
```

In the values.yaml file add the configuration:

```
podVolumes:
  create: false
  data:
    - name: hadoop-conf-volume
      configMap:
        name: hadoop-conf
    - name: custom-truststore-volume
      secret:
        secretName: custom-truststore
podVolumeMounts:
  create: false
  data:
    - name: hadoop-conf-volume
      mountPath: /etc/hadoop/conf
      readOnly: true
    - name: custom-truststore-volume
      mountPath: /opt/flink/tls/
      readOnly: true
```

If you mounted a custom truststore, you can reference it when creating a table from SSB. Provide the truststore properties according to the connector type.

An example for the Kafka connector:

```
CREATE TABLE `example_kafka` (
  `id` INT,
  `name` STRING
) WITH (
  'connector' = 'kafka',
  'topic' = 'example_topic',
  'properties.bootstrap.servers' = 'kafka-ssl.example.com:9092',
  'properties.security.protocol' = 'SSL',
  'properties.ssl.truststore.location' = '/opt/flink/tls/truststore.jks',
  'properties.ssl.truststore.password' = 'changeit'
);
```

Kerberos configuration

To enable Kerberos authentication, you need to add the Hadoop dependencies to the Cloudera Streaming Analytics images as described in the *Customize container images* section.

After adding the dependencies, you need to ensure that the Hadoop configuration and krb5.conf files are added as a configmap using the following commands:

```
kubectl -n flink create configmap hadoop-conf --from-file core-site.xml=core-site.xml --from-file hdfs-site.xml=hdfs-site.xml
```

```
kubectl -n flink create configmap krb5-conf --from-file krb5.conf=krb5.conf
```

When the configmaps are in place, the following configuration properties should be updated in the values.yml file for the configuration files to be mounted on the containers:

```
ssb:
  podVolumes:
    create: true
    data:
      - name: hadoop-conf-volume
        configMap:
          name: hadoop-conf
      - name: krb5-conf-volume
        configMap:
          name: krb5-conf
  podVolumeMounts:
    create: true
    data:
      - name: hadoop-conf-volume
        mountPath: /etc/hadoop/conf
        readOnly: true
      - name: krb5-conf-volume
        mountPath: /etc/krb5.conf
        subPath: krb5.conf
```

After setting up images and configurations, you can use Streaming SQL Console to specify your keytabs in the Keytab Manager. Once the keytab is successfully validated, Kerberos will be automatically configured when a new job is deployed.

Related Information

[Customize container images](#)

[Cloudera SQL Stream Builder database support](#)

[Routing with ingress](#)

Database configuration

Configure the Cloudera SQL Stream Builder's database.

By default, the Helm chart automatically creates a PostgreSQL Deployment and Service using a public image. The Helm chart also creates a PersistentVolumeClaim, which will persist throughout Helm uninstalls and installs using the "helm.sh/resource-policy": "keep" annotation.

If you remove this PersistentVolumeClaim, on the next SSB restart all the data saved by Cloudera SQL Stream Builder will be lost, and the database will be re-created.

It is recommended that you use an already existing external database, and configure Cloudera SQL Stream Builder to connect to it, instead of using the default created database. You can use MySQL/MariaDB, Oracle or PostgreSQL as a database. (See *Cloudera SQL Stream Builder database support*.)

You can use the following example configuration to override the default database configuration:

```
ssb:
  database:
    create: false
    auth:
      type: postgresql
      jdbcUrl: "jdbc:postgresql://postgres-host:5432/ssb_admin"
      username: test_user
      password: test_password
```

The data provided here is kept secure as a Kubernetes secret.

Customize container images

Updating container images to use Kudu, Hive, HBase, and HDFS with the Cloudera SQL Stream Builder.

To be able to use Kudu, Hive, HBase or HDFS, you need to update the images supplied to you, and add the required JAR files and dependencies using Dockerfiles.

There are two images you need to update, both of which can be found under the `sqlRunner.image` and `sse.image` configurations. `sqlRunner.image` is the image that will be used for the Flink deployments. This image is responsible for executing the SQL commands. `sse.image` is Cloudera SQL Stream Builder itself.

If you want to use the updated container image, make sure to upload it to a registry your Kubernetes cluster can access, and update the configuration in the `values.yaml` file to point to your new images.

Here is an example of adding Hadoop and Hive to the SQL Runner image:

```
FROM [***REGISTRY HOST***]:[***PORT***]/[***PROJECT***]/ssb-sql-runner:lates
t

ENV CLOUDERA_ARCHIVES "https://archive.cloudera.com"
# Hadoop
ENV HADOOP_VERSION "3.1.1.7.1.9.0-387"
ENV HADOOP_HOME "/opt/hadoop"
RUN rm -rf ${HADOOP_HOME}/ \
  && cd /opt \
  && curl -sL --retry 3 "https://${CLOUDERA_ARCHIVES}/artifacts/build/447
02451/cdh/7.x/redhat8/yum/tars/hadoop/hadoop-client-${HADOOP_VERSION}.tar.gz
" | tar xz \
  && chown -R root:root hadoop-client-${HADOOP_VERSION} \
```

```

    && ln -sfn hadoop-client-`${HADOOP_VERSION}` hadoop \
    && rm -rf `${HADOOP_HOME}`/share/doc \
    && find /opt/ -name *-sources.jar -delete
ENV HADOOP_CONF_DIR "${HADOOP_HOME}/etc/hadoop"
ENV PATH="${HADOOP_HOME}/bin:${PATH}"
ENV HADOOP_CLASSPATH "/opt/hadoop/share/hadoop/client/lib/*"
# Hive
RUN wget https://`${CLOUDERA_ARCHIVES}`/maven/org/apache/flink/flink-sql-conne
ctor-hive-3.1.3_2.12/1.18.0-csaopl.0.0/flink-sql-connector-hive-3.1.3_2.12-1
.18.0-csaopl.0.0.jar \
    -O /opt/flink/lib/flink-sql-connector-hive-3.1.3_2.12-1.18.0-csaopl.0
.0.jar

```

Here is an example of adding Hadoop and Hive to the SSB image:

```

FROM [***REGISTRY HOST***]:[***PORT***]/[***PROJECT***]/ssb-sse:latest

ENV CLOUDERA_ARCHIVES "https://archive.cloudera.com"

ENV HADOOP_VERSION "3.1.1.7.1.9.0-387"
ENV HADOOP_HOME "/opt/hadoop"
RUN rm -rf `${HADOOP_HOME}`/ \
    && cd /opt \
    && curl -sL --retry 3 "https://`${CLOUDERA_ARCHIVES}`/artifacts/build/447
02451/cdh/7.x/redhat8/yum/tars/hadoop/hadoop-client-`${HADOOP_VERSION}`.tar.gz
" | tar xz \
    && chown -R root:root hadoop-client-`${HADOOP_VERSION}` \
    && ln -sfn hadoop-client-`${HADOOP_VERSION}` hadoop \
    && rm -rf `${HADOOP_HOME}`/share/doc \
    && find /opt/ -name *-sources.jar -delete
ENV HADOOP_CONF_DIR "${HADOOP_HOME}/etc/hadoop"
ENV PATH="${HADOOP_HOME}/bin:${PATH}"

# Only copy Hadoop jars that are required for SSB to communicate with Hive
RUN cp "${HADOOP_HOME}/share/hadoop/client/lib/hadoop-common-`${HADOOP_VER
SION}`.jar" /opt/cloudera/ssb-sse/lib/ \
    && cp "${HADOOP_HOME}/share/hadoop/client/lib/hadoop-auth-`${HADOOP_VERSI
ON}`.jar" /opt/cloudera/ssb-sse/lib/ \
    && cp "${HADOOP_HOME}/share/hadoop/client/lib/hadoop-mapreduce-client-
core-`${HADOOP_VERSION}`.jar" /opt/cloudera/ssb-sse/lib/ \

# Hive
RUN wget https://`${CLOUDERA_ARCHIVES}`/maven/org/apache/flink/flink-sql-conn
ector-hive-3.1.3_2.12/1.18.0-csaopl.0.0/flink-sql-connector-hive-3.1.3_2.12-
1.18.0-csaopl.0.0.jar \
    -O /opt/cloudera/ssb-sse/lib/flink-sql-connector-hive-3.1.3_2.12-1.18.0.
jar

```