

Security

Date published: 2019-12-17

Date modified: 2023-02-23

CLOUdera

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

Securing Apache Flink.....	4
Authentication and encryption for Flink.....	4
Enabling security for Apache Flink.....	4
Configuring custom Kerberos principal for Apache Flink.....	6
Enabling SPNEGO authentication for Flink Dashboard.....	6
Enabling Knox authentication for Flink Dashboard.....	7
Enabling Knox Auto Discovery for Flink.....	7
Accessing the Flink Dashboard through Knox.....	8
Configuring Ranger policies for Flink.....	8
Adding Flink group to Kafka policies.....	9
Adding Flink group to Schema Registry policies.....	9
Adding Flink group to Kudu policies.....	9
Securing Apache Flink jobs.....	10
Using EncryptTool for Flink properties.....	11
 Securing SQL Stream Builder.....	 12
Authentication in SSB.....	12
Enabling Kerberos authentication.....	14
Enabling Knox authentication.....	17
Uploading or unlocking your keytab.....	27
Encryption in SSB.....	29
Enabling TLS for database connection.....	31
Configuring Ranger policies for SSB.....	31
Adding SSB user to Kafka policies.....	32
Adding SSB user to Schema Registry policies.....	32
Adding SSB user to Hive policies.....	32
Adding SSB user to Kudu policies.....	33

Securing Apache Flink

Authentication and encryption for Flink

You must use authentication and encryption to secure your data and data sources. You can use Kerberos and TLS/SSL authentication to secure your Flink jobs. The administrator should provide your keystore and truststore credentials for your Cloudera user.

Authentication

While meeting the security requirements for various connectors is an ongoing effort, Flink provides first-class support for Kerberos authentication only.

The primary goals of the Flink Kerberos security infrastructure are:

- to enable secure data access for jobs within a cluster through connectors (for example, Kafka)
- to authenticate to Hadoop components (for example, HDFS, HBase, Zookeeper)
- to enable secure SPNEGO for Global Dashboard access

In a production deployment scenario, streaming jobs usually run for long periods of time. Authentication is mandatory to secure data sources throughout the lifetime of a job. Kerberos keytabs do not expire in that timeframe, unlike a Hadoop delegation token or ticket cache entry. Cloudera recommends using keytabs for long-running production deployments.

Encryption (TLS)

Flink differentiates between internal and external connectivity in case of encryption.

Internal connectivity refers to all connections made between Flink processes. Because internal communication is mutually authenticated, keystore and truststore typically contain the same dedicated certificate. The certificate can use wildcard hostnames or addresses because the certificate is expected to be a shared secret and hostnames are not verified.

External connectivity refers to all connections made from the outside to Flink processes. When Flink applications are running on CDP Private Cloud Base clusters, the Flink web dashboard is accessible through the tracking URL of the YARN proxy. Depending on the security setup in YARN, the proxy itself can enforce authentication (SPNEGO) and encryption (TLS) already for YARN jobs. This can be sufficient when CDP perimeter is protected by a firewall from external user access. If there is no such protection available, additional TLS configuration is required to protect REST endpoints with TLS.

For more information, see the Apache Flink [documentation](#).

Enabling security for Apache Flink

Since Flink is essentially just a YARN application, you mainly need to configure service level security settings for the Flink Dashboard and Gateway in Cloudera Manager. You can configure security during the installation or later in the Configuration menu for Flink.

Kerberos

Kerberos authentication can be enabled for Flink by simply checking the corresponding checkbox in the service wizard while adding the service or later in the service configuration page in Cloudera Manager. The service wizard in Cloudera Manager enables the Kerberos service, and no further action is required to be able to use the authentication with Flink.

For more information about enabling Kerberos authentication using the service wizard, see the [Cloudera Manager documentation](#).

TLS encryption

If AutoTLS is enabled on the cluster, the TLS-related configuration fields are auto-populated for the Flink Dashboard and Gateway. You can set `{{CM_AUTO_TLS}}` as value for the security properties when using AutoTLS in Cloudera Manager. If AutoTLS is not used, the settings have to be configured manually.

Before you begin

Ensure that you have set up TLS for Cloudera Manager:

- Generate TLS certificates
- Configure TLS for Admin Console and Agents
- Enable server certificate verification on Agents
- Configure agent certificate authentication
- Configure agent certificate authentication

Procedure

1. Click Flink service on your Cluster.
2. Click the Configuration tab.
3. Select Category > Security.
All the security related properties are displayed.
4. Edit the security properties according to the cluster configuration.



Note: You need to provide the keystore and truststore information for the Flink Dashboard and the Gateway as well.

Security property	Description
Enable TLS/SSL for Flink Dashboard	Select the checkbox to enable TLS/SSL for Flink Dashboard to encrypt communication between the clients and Flink Dashboard.
Flink Dashboard TLS/SSL Server JKS Keystore File Location	Path to the keystore file containing the server certificate and private key used for TLS/SSL. The keystore must be in JKS format.
Flink Dashboard TLS/SSL Server JKS Keystore File Password	Password for the Flink Dashboard JKS keystore file.
Flink Dashboard TLS/SSL Server JKS Keystore Key Password	Password that protects the private key contained in the JKS keystore.
Flink Dashboard TLS/SSL Client Trust Store File	Location of the truststore file on disk. The truststore file must be in JKS format. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
Flink Dashboard TLS/SSL Client Trust Store Password	Password for the Flink Dashboard TLS/SSL Certificate Trust Store File. Provides optional integrity checking of the file. This password is not required to access the trust store, this field can be left blank.
Gateway TLS/SSL Client Trust Store File	Location of the truststore file on disk. The truststore file must be in JKS format. This is used when Gateway is the client in a TLS/SSL connection. If this parameter is not provided, the default list of well-known certificate authorities is used instead.
Gateway TLS/SSL Client Trust Store Password	The password for the Gateway TLS/SSL Certificate Trust Store. Provides optional integrity checking of the file. This password is not required to access the trust store, this field can be left blank.

5. Click Save Changes.

Related Information

[Secure Tutorial](#)

Configuring custom Kerberos principal for Apache Flink

The Kerberos principal for Flink is configured by default to use the same service principal as the default process user. However, you can change the default setting by providing a custom principal in Cloudera Manager.

About this task



Important: Cloudera Manager configures CDP services to use the default Kerberos principal names. Cloudera recommends that you do not change the default Kerberos principal names. If it is unavoidable to do so, contact Cloudera Professional Services because it requires extensive additional custom configuration.

Procedure

1. Go to your Cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Go to the Configuration tab.
4. Search for the Kerberos principal by entering kerberos in the search field.
5. Provide a custom name to the Kerberos Principal property.
6. Click Save Changes.
7. Click Actions > Restart next to the Flink service name to restart the service.

Enabling SPNEGO authentication for Flink Dashboard

You must manually configure the SPNEGO authentication for Flink Dashboard in Cloudera Manager to enable secure access for users as by default the authentication is turned off.

Enabling SPNEGO authentication for Flink Dashboard

1. Go to your cluster in Cloudera Manager.
2. Select Flink from the list of services.
3. Select the Configuration tab.
4. Filter to Scope > Flink Dashboard.
5. Search for Use SPNEGO Authentication.
6. Select the checkbox to enable SPNEGO authentication for Flink Dashboard.
7. Click Save Changes.

You need to restart the Flink service to finalize the configuration.

8. Click on Actions > Restart next to the Flink service name.

Providing user credentials for flink list

The Flink CLI uses the Flink Dashboard when you use the flink list command. In this case, the Flink CLI connects to the Flink Dashboard and lists the running and scheduled applications. The connection between the CLI and Dashboard requires user credentials for the SPNEGO authentication.

The following methods can be used to provide the user credentials:

- You can use the kinit command and the custom ticket cache file:

1. Connect to your host using ssh.

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

You are prompted to provide your password.

2. Run the kinit command to obtain a valid TGT.

```
kinit <your_principal>
```

In this case, the flink list command reads the TGT from the default ticket cache file of the user.



Note: When you need to use custom ticket cache file, you must set the path of the cache directory:

```
KRB5CCNAME=/path/to/custom/ticket/cache
```

- You can provide the keytab file and login principals directly to the flink list command:

1. Connect to your host using ssh.

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

You are prompted to provide your password.

2. Run the flink list command using your keytab information:

```
flink list -yD security.kerberos.login.keytab=<your_keytab_file_name> -  
yD security.kerberos.login.principal=<your_principal>
```

Enabling Knox authentication for Flink Dashboard

You can use Knox authentication for Flink Dashboard to provide integration with customer Single Sign-On (SSO) solutions. Knox uses Kerberos (SPNEGO) to strongly authenticate itself towards the services.

Before you begin

- Install and configure Knox on your cluster. For more information, see the [Installing Apache Knox](#) documentation.
- Enable Kerberos authentication for Flink and the Flink Dashboard. For more information, see [Enabling security for Apache Flink](#) and [Enabling SPENGO authentication for Flink Dashboard](#) sections.

Enabling Knox Auto Discovery for Flink

The Auto Discovery feature of Knox is supported for Flink. This means that you only need to enable the Knox Auto Discovery feature for Flink, and Cloudera Manager provides and manages all the required service definition files.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Select SQL Stream Builder from the list of services.
3. Click Configuration.
4. Search for flink in the Search field.
The Enable Auto Discovery (cdp-proxy) - Flink property is listed.
5. Check the Auto Discovery (cdp-proxy) property box for the Enable Auto Discovery (cdp-proxy) - Flink property.
6. Click on Save changes
The Refresh needed indicator appears beside the Knox service name.
7. Refresh Knox.

Results

Knox is enabled for the Flink Dashboard.

What to do next

Complete the steps in *Accessing the Flink Dashboard through Knox* section to open the Flink Dashboard.

Related Information

[Accessing the Flink Dashboard through Knox](#)

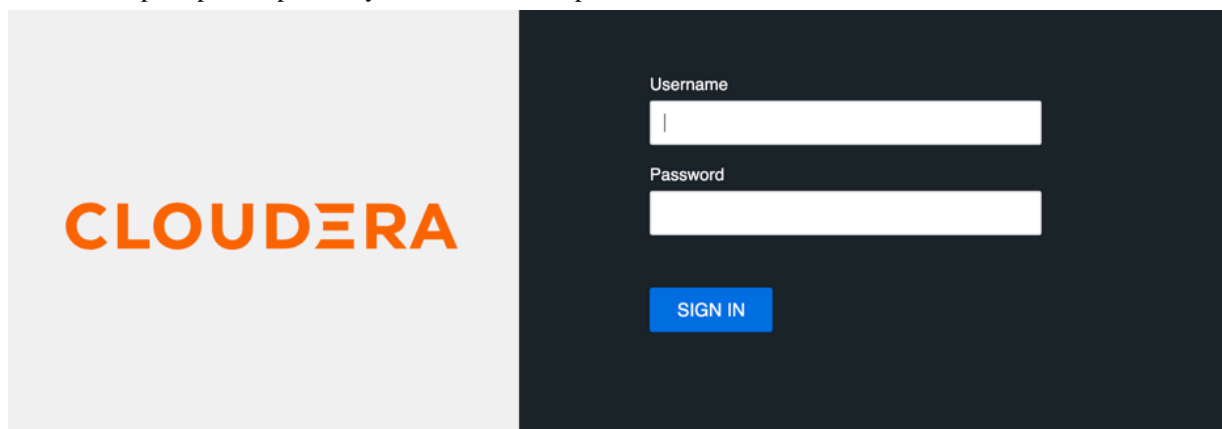
Accessing the Flink Dashboard through Knox

After enabling Knox authentication either manually or using the Auto Discovery feature, you can only reach the Flink Dashboard page through the Knox Gateway.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Click on Knox from the list of Services.
3. Select Knox Gateway Home.

You will be prompted to provide your username and password.



4. Click cdp-proxy under Topologies.
Flink Dashboard should be listed under the cdp-proxy.
5. Click on Flink Dashboard.
You are redirected to the Flink Dashboard page.

Configuring Ranger policies for Flink

You must add Flink users to the Ranger policies that are used by Kafka, Schema Registry, and Kudu to provide access to topics, schemas and tables provided by the components.

Before you begin

Install Apache Ranger on your cluster. For more information, see the [Production Installation](#) documentation.

You can reach the Ranger User Interface through Cloudera Manager:

1. Go to your cluster in Cloudera Manager.
2. Select Ranger from the list of services.
3. Click on Ranger Admin Web UI.

You are redirected to the Ranger Service Manager.

You need to create a Flink user group, and add the Flink users to set the required permissions in a group level:

1. Create a Flink user group in the Ranger Service Manager.
 - a. Click Settings > Users/Groups/Roles.
 - b. Select Groups tab.
 - c. Click on Add New Group.
 - d. Provide a Name to the group and a Description.
 - e. Click Save.
2. Add new users to the Flink group.
 - a. Click Settings > Users/Groups/Roles.
 - b. Select Users tab.
 - c. Click on Add New User.
 - d. Provide the basic information about the user.
 - e. Select a Role to the user.
 - f. Select the created Flink group.
 - g. Click Save.

Adding Flink group to Kafka policies

You must add the Flink group to the following policies:

- all-consumergroup
 - all-topic
1. Select cm_kafka from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the list of Kafka policies page.
 2. Click on the edit button of the all-consumergroup policy.
 3. Add the Flink group to the Select Group field under the Allow Conditions setting.
 4. Click Save.

You are redirected to the list of Kafka policies page
 5. Click on + More... to check if the Flink group is listed under the Groups for the consumergroup policy.
 6. Add the Flink user to the following policy with the above steps as well:
 - all-topic

Adding Flink group to Schema Registry policies

You must add the Flink group to the following policy:

- all-schema-group, schema-metadata, schema-branch, schema-version
1. Select cm_schema-registry from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the list of Schema Registry policies page.
 2. Click on the edit button of the all-schema-group, schema-metadata, schema-branch, schema-version policy.
 3. Add the Flink user to the Select Group field under the Allow Conditions setting.
 4. Click Save.

You are redirected to the list of Schema Registry policies page.
 5. Click on + More... to check if the Flink group is listed under the Groups for the schema-group, schema-metadata, schema-branch, schema-version policy.

Adding Flink group to Kudu policies

You must create a policy to grant access to Kudu tables for the Flink group.

1. Select `cm_kudu` from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the Create Policy page.

2. Click on Add New Policy.
3. Provide a name to the Policy Name field.
4. Provide a prefix for the Databases you want to add to the policy or type `*` to select all.
5. Provide a prefix for the table you want to add to the policy or type `*` to select all.
6. Provide a prefix for the column you want to add to the policy or type `*` to select all.
7. Add the Flink user to the Select User field under the Allow Conditions setting.
8. Click on the plus icon to Add Permissions to the Permissions field.
9. Click on the specific permissions or Select All.
10. Click on Add at the bottom of the page.

You are redirected to the list of Kudu policies page where the created policy should be listed.

11. Click on + More... to check if the Flink group is listed under the Groups for the created policy.

Securing Apache Flink jobs

Submitting Flink jobs in a secure environment requires every security parameter for authentication, authorization and other connector related security settings. You should prepare your keystore and keytab files for Flink and for also the chosen connector component.

The following example shows the security parameters that are needed to submit a Flink job:

```
flink run -d -p 2 \
-yD security.kerberos.login.keytab=test.keytab \
-yD security.kerberos.login.principal=test \
-yD security.ssl.internal.enabled=true \
-yD security.ssl.internal.keystore=keystore.jks \
-yD security.ssl.internal.key-password='cat pwd.txt' \
-yD security.ssl.internal.keystore-password='cat pwd.txt' \
-yD security.ssl.internal.truststore=keystore.jks \
-yD security.ssl.internal.truststore-password='cat pwd.txt' \
-yt keystore.jks \
flink-secure-tutorial-1.0-SNAPSHOT.jar \
--kafkaTopic flink \
--hdfsOutput hdfs:///tmp/flink-secure-tutorial \
--kafka.bootstrap.servers <broker_host>:9093 \
--kafka.security.protocol SASL_SSL \
--kafka.sasl.kerberos.service.name kafka \
--kafka.ssl.truststore.location /etc/cdep-ssl-conf/CA_STANDARD/truststore.jks
```

The Kerberos and TLS properties are user specific parameters. Generally the cluster administrator provides the Kerberos keytab and TLS certificate to the user. In case you did not receive the keytab and keystore file from the administrator, you can use the following commands:

```
> ktutil
ktutil: add_entry -password -p test -k 1 -e des3-cbc-sha1
Password for test@:
ktutil: wkt test.keytab
ktutil: quit
```

```
keytool -genkeypair -alias flink.internal -keystore keystore.jks -dname "CN=flink.internal" -storepass 'cat pwd.txt' -keyalg RSA -keysize 4096 -storetype PKCS12
```



Note: The keytool can be accessed at `/usr/java/default/bin/keytool` if the `JAVA_HOME` is not set globally on the host.

The full explanation of the properties used in the example can be found in the Secure Tutorial. It also includes how to enable security features step-by-step for Flink applications that are running on secured CDP Private Cloud Base environments.

Related Information

[Secure Tutorial](#)

Using EncryptTool for Flink properties

Cloudera Streaming Analytics offers EncryptTool to further protect your user information and configurations when communicating with Flink using the command line. After generating a master key to the user, you need to manually encrypt the parameters and Flink automatically decrypts the protected values. You also must enable EncryptTool protection in the configuration file for Flink.

About this task

In addition to the functionality provided by the vanilla version of Apache Flink, CSA includes a solution to protect for sensitive properties in the configuration file and the dynamic properties. This way passwords in clear text to Flink can be avoided.

There are two actions available through the `flink-encrypt-tool` command line client to use the EncryptTool:

- `generate-key`: generating master key per user. The master key is saved to an arbitrary filesystem location specified by the user, by default to the HDFS home folder of the user. It is the responsibility to protect the privileges of the key, so that it is only accessible by them. EncryptTool assumes that all sensitive properties are protected using the same key in a single configuration file.
- `encrypt`: encrypting configuration property. The configuration properties have to be manually encrypted and updated in the configuration file or supplied in encrypted format via dynamic properties.

Flink automatically decrypts the values based on the configuration object during runtime with the privileges of the user that has submitted the Flink job, so the visibility of the key has to be set up accordingly.

Users can override the default key location by setting the following property in the `flink-conf.yaml`:

```
security.encrypt-tool.key.location:  
hdfs:///user/alice/myencryptionkey
```

In order for the `flink-encrypt-tool` to use the modified configuration file, one can set the following environment variable: `export FLINK_CONF_DIR=/path/to/modified-flink-conf-dir`

Procedure

1. Generate master key using `generate-key` action.
2. Define a secure location for the master key.
3. Use `encrypt` action to get the encrypted value for each sensitive key.
4. Update the configuration.

What to do next

Once the encryption of each property is performed and saved also set the following flag to indicate that the configuration encryption is enabled: `security.encrypt-tool.enabled: true`

**Note:**

Currently the tool only supports all or nothing protection. This means that once it is enabled, the following configuration values have to be encrypted if specified:

- security.ssl.internal.truststore-password
- security.ssl.internal.keystore-password
- security.ssl.internal.key-password
- security.ssl.truststore-password
- security.ssl.keystore-password
- security.ssl.key-password
- security.ssl.rest.truststore-password
- security.ssl.rest.keystore-password
- security.ssl.rest.key-password

Securing SQL Stream Builder

Authentication in SSB

You can authenticate users to access the Streaming SQL Console using Kerberos or Knox authentication.

On a non-Kerberized cluster, you need to register a new account to access the Streaming SQL Console. Provide a Username, Password, and your First Name and Last Name to create an account.



SSB Registration

Username *

Password *

First Name

Last Name

Already have an account? [Login](#)

After creating an account, you can log in to the Streaming SQL Console by providing the registered Username and Password.



SSB Login

Username *

Password *

Sign In

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



Note: You can later change your password by clicking on your username on the main menu of Streaming SQL Console, and by selecting the Profile tab.

Change Password

Current Password *

New Password *

New Password Again *

Change Password

Enabling Kerberos authentication

You need to enable Kerberos authentication in Cloudera Manager as well as directly for your browser to securely reach the Streaming SQL Console, and to use Knox authentication.

When Kerberos authentication is set up for SSB, and an unauthorized user wants to reach the Streaming SQL Console, the following error message appears:

401 Not Authenticated

Kerberos authentication is enabled, SPNEGO is required to access this page.

1. Go to your cluster in Cloudera Manager.
2. Click SQL Stream Builder from the list of services.
3. Go to the Configuration tab.
4. Select Category > Security .
5. Type *kerberos* in the search field.
6. Select the Enable Kerberos authentication setting.
7. Open a terminal window.

8. Configure your browser for Kerberos authentication:**For Mozilla Firefox**

- a. Load the about:config page to open the low level Firefox configuration.
- b. Search for network.negotiate-auth.trusted-uris preference.
- c. Open the network.negotiate-auth.trusted-uris preference.
- d. Enter the hostnames of the SQL Stream Console are protected by Kerberos HTTP SPNEGO.
- e. Click Ok.

For Internet Explorer

- a. Configure the Local Intranet Domain
 1. Click on the Settings icon in Internet Explorer.
 2. Go to Internet options > Security.
 3. Select Local Intranet zone.
 4. Click on Sites.
 5. Review that the following options are checked:
 - Include all local (intranet) sites not listed in other zones
 - Include all sites that bypass the proxy server are checked
 6. Click Advanced.
 7. Enter the hostnames and domains of the SQL Stream Console that are protected by Kerberos HTTP SPNEGO.
 8. Click Ok.
- b. Configure the Intranet Authentication
 1. Click on the Settings icon in Internet Explorer.
 2. Go to Internet options > Security.
 3. Select Local Intranet zone.
 4. Click on Custom level.

The Security Settings - Local Intranet Zone dialog box opens.
 5. Scroll down to the User Authentication options.
 6. Select Automatic logon only in Intranet Zone.
 7. Click Ok.
- c. Verify the Proxy Settings
 1. Make sure that you enabled a proxy server.
 2. Click on the Settings icon in Internet Explorer.
 3. Go to Internet options > Connections.
 4. Select LAN settings.
 5. Confirm that the proxy server Address and Port number are correct.
 6. Click Advanced.

The Proxy Settings dialog box opens.
 7. Add the Streaming SQL Console domains that are protected by Kerberos to the Exceptions field.
 8. Click Ok.

For Google Chrome

- Windows
 - a. Open Control Panel > Internet Options > Security.
 - b. Perform the steps from the Internet Explorer configuration.
- MacOS
 - a. Open a terminal window.

- b. Copy and paste the following command:

```
defaults write com.google.Chrome AuthServerAllowlist  
"*<host_domain>,*<host_domain1>,*<host_domain2>"
```

```
sudo scp <your_hostname>:/etc/krb5.conf /etc/krb5.conf
```

You will be prompted to provide your password.

```
kinit <username>
```

Configuring custom Kerberos principal for SQL Stream Builder

The Kerberos principal for SQL Stream Builder is configured by default to use the same service principal as the default process user. However, you can change the default setting by providing a custom principal in Cloudera Manager.

About this task



Important: Cloudera Manager configures CDP services to use the default Kerberos principal names. Cloudera recommends that you do not change the default Kerberos principal names. If it is unavoidable to do so, contact Cloudera Professional Services because it requires extensive additional custom configuration.

Procedure

1. Go to your Cluster in Cloudera Manager.
2. Select SQL Stream Builder from the list of services.
3. Go to the Configuration tab.
4. Search for the Kerberos principal by entering kerberos in the search field.
5. Provide a custom name to the Kerberos Principal property.
6. Click Save Changes.
7. Click Actions > Restart next to the SQL Stream Builder service name to restart the service

Enabling Knox authentication

You can use Knox authentication for SQL Stream Builder (SSB) to provide integration with customer Single Sign-On (SSO) solutions. Knox uses Kerberos (SPNEGO) to strongly authenticate itself towards the services.

Apache Knox Gateway is used to help ensure perimeter security for SSB. With Knox, enterprises can confidently extend the SSB UI and API endpoints to new users without Kerberos complexities. Knox provides a central gateway and has varying degrees of authorization, authentication, SSL, and SSO capabilities to enable a single access point for SSB.

Before you begin

- Install and configure Knox on your cluster. For more information, see the [Installing Apache Knox](#) documentation.
- Enable Kerberos authentication for SSB. For more information, see [Enabling Kerberos authentication](#) section.

When using SSB on CDP Private Cloud Base, the Auto Discovery feature of Knox is not supported. This means you must manually configure Knox by adding SSB as a custom service to the cdp-proxy configuration. When using CDP Private Cloud Base 7.1.7 an additional configuration step is required to add the service definitions to Knox.

Enabling Knox Auto Discovery for MVE without Load Balancer

When using SQL Stream Builder (SSB), the Auto Discovery feature of Knox is supported for the Materialized View Engine (MVE). This means that you need to enable the Knox Auto Discovery feature for the MVE if you plan to use SSB without Load Balancer, and Cloudera Manager provides and manages all the required service definition files. In case the Load Balancer is enabled, you need to manually add the service definition to Knox.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Select Knox from the list of services.
3. Select Configuration.
4. Search for mve in the Search field.
5. Check the Enable Auto Discovery (cdp-proxy-api) - SQL Streaming Builder - Materialized View Engine API property.
6. Click Save Changes.
The Refresh needed indicator appears beside the Knox service name.
7. Refresh Knox.

What to do next

If you use SSB on CDP Private Cloud Base 7.1.7, add the SSB service definitions to Knox, and then configure the default topologies in Cloudera Manager. If you use SSB on CDP Private Cloud Base 7.1.8, you do not need to add the service definitions, which means you can continue with configuring the default topologies of Knox in Cloudera Manager.

Extending Knox with the SSB service definitions in CDP Private Cloud Base 7.1.7

When using CDP Private Cloud Base 7.1.7, you must create service definitions for the SSB engines and APIs in the Knox Admin UI.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Select Knox from the list of services.
3. Select Knox Gateway Home.
4. Open the General Proxy Information.
5. Click Admin UI URL.
You are redirected to the Knox Manager page.
6. Click Service Definitions under Resource Types.

7. Click on the plus icon to add the SSB service definitions.
The Create a New Service Definition window appears.

Create a New Service Definition
✕

At a minimum you must modify the service's name and role attributes!

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <serviceDefinition>
3   <service name="your_service_name" role="YOUR_ROLE" version="1.0.0">
4     <dispatch classname="org.apache.knox.gateway.dispatch.DefaultDispatch" use-two-way-ssl="false" />
5     <routes />
6   </service>
7   <rules />
8 </serviceDefinition>
9

```

Cancel
Ok

8. Delete the default text from the window.
9. Create the service definitions for SSB.

For Without Load Balancer

- a. Copy the following XML entry for the SSB-SSE-UI service definition:

```

<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
  <serviceDefinition>
    <service name="ssb-sse-ui" role="SSB-SSE-UI" version="1.7.0">
      <metadata>
        <context>/ssb-sse-ui</context>
        <description>SSB-SSE-UI</description>
        <shortDesc>SSB-SSE-UI</shortDesc>
        <type>UI</type>
      </metadata>
      <dispatch classname="org.apache.knox.gateway.dispatch.ConfigurableDispatch" use-two-way-ssl="false">
        <param>
          <name>responseExcludeHeaders</name>
          <value>CONTENT-LENGTH, WWW-AUTHENTICATE</value>
        </param>
      </dispatch>
      <routes>
        <route path="/ssb-sse-ui/" />
        <route path="/ssb-sse-ui/**" />
        <route path="/ssb-sse-ui/**?*" />
        <route path="/ssb-sse-ui/swagger/**" />
      </routes>
    </service>
  </serviceDefinition>
</serviceDefinitions>

```

```

        <rewrite apply="SSB-SSE-UI/filter/outbound/swagger/body" t
o="response.body"/>
      </route>
    </routes>
  </service>
  <rules>
    <rule name="SSB-SSE-UI/ssb-sse-ui/inbound/root" pattern="*://*:*/
*/ssb-sse-ui/">
      <rewrite template="{ $serviceUrl[SSB-SSE-UI] }"/>
    </rule>
    <rule name="SSB-SSE-UI/ssb-sse-ui/inbound/path" pattern="*://*:*/
**/ssb-sse-ui/{**}">
      <rewrite template="{ $serviceUrl[SSB-SSE-UI] }/{**}"/>
    </rule>
    <rule name="SSB-SSE-UI/ssb-sse-ui/inbound/query" pattern="*://*:*/
**/ssb-sse-ui/{**}?{**}">
      <rewrite template="{ $serviceUrl[SSB-SSE-UI] }/{**}?{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/api" pattern="
/api/{**}"/>
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/api/{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/ui" pattern
="*://*:*/ui/{**}">
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/ui/{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/frontend-ui" p
attern="/ui/">
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/ui/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/internal" p
attern="/internal/">
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/internal/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/swagger-url"
pattern="/swagger/{**}">
      <rewrite template="{ $frontend[path] }/ssb-sse-ui/swagger/{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/swagger-confi
g-url" pattern="/swagger-ui/{**}">
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/swagger-ui/{**}"
/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/html/websocket"
pattern="/websocket/">
      <rewrite template="{ $frontend[url] }/ssb-sse-ws/websocket/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/json/root">
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI/rule/outbound/json/path">
      <rewrite template="{ $frontend[url] }/ssb-sse-ui/{**}"/>
    </rule>
    <filter name="SSB-SSE-UI/filter/outbound/swagger/body">
      <content type="*/json">
        <apply path="$.servers[0].url" rule="SSB-SSE-UI/rule/outb
ound/json/root"/>
        <apply path="$.oauth2RedirectUrl" rule="SSB-SSE-UI/rule/o
utbound/json/path"/>
      </content>
    </filter>
  </rules>
</serviceDefinition>

```

```
</serviceDefinitions>
```

- b. Paste it to the New Service Definition window.
- c. Click Ok.
- d. Click on the plus icon to create a new service definition.
- e. Copy the following XML entry for the SSB-SSE-WS service definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
  <serviceDefinition>
    <service name="ssb-sse-ws" role="SSB-SSE-WS" version="1.7.0">
      <metadata>
        <context>/ssb-sse-ws</context>
        <description>SSB-SSE-WS</description>
        <shortDesc>SSB-SSE-WS</shortDesc>
        <type>API</type>
      </metadata>
      <dispatch classname="org.apache.knox.gateway.dispatch.ConfigurableDispatch" use-two-way-ssl="false">
        <param>
          <name>responseExcludeHeaders</name>
          <value>CONTENT-LENGTH,WWW-AUTHENTICATE</value>
        </param>
        <param>
          <name>requestExcludeHeaders</name>
          <value>Cookie,Origin</value>
        </param>
      </dispatch>
      <routes>
        <route path="/ssb-sse-ws/**">
          <rewrite apply="SSB-SSE-WS/rule/inbound1" to="request.url"/>
        </route>
      </routes>
    </service>
  </serviceDefinition>
</serviceDefinitions>
```

- f. Paste it to the New Service Definition window.
- g. Click Ok.
- h. Click on the plus icon to create a new service definition.
- i. Copy the following XML entry for the SSB-SSE-API service definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
  <serviceDefinition>
    <service name="ssb-sse-api" role="SSB-SSE-API" version="1.5.0.0">
      <metadata>
        <context>/ssb-sse-api</context>
        <description>Streaming SQL Builder - Streaming SQL Engine API</description>
        <shortDesc>SSB - SSE API</shortDesc>
        <type>API</type>
      </metadata>
      <routes>
        <route path="/ssb-sse-api/api/**?*">
```

```

        <rewrite apply="SSB-SSE-API/ssb-sse-api/api/path" t
o="request.url"/>
      </route>
    </routes>
  </service>
  <rules>
    <rule dir="IN" name="SSB-SSE-API/ssb-sse-api/api/path" pat
tern="*:*//*:*/**/ssb-sse-api/api/{path=**}?{**}">
      <rewrite template="{ $serviceUrl[SSB-SSE-API]}/api/{pat
h=**}?{**}"/>
    </rule>
  </rules>
</serviceDefinition>
</serviceDefinitions>

```

- j.** Paste it to the New Service Definition window.
- k.** Click Ok.

For With Load Balancer

- a.** Copy the following XML entry for the SSB-SSE-UI-LB service definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
  <serviceDefinition>
    <service name="ssb-sse-ui-lb" role="SSB-SSE-UI-LB" version="1.7.0"
">
    <dispatch classname="org.apache.knox.gateway.dispatch.ConfigurableDispatch" use-two-way-ssl="false">
      <param>
        <name>responseExcludeHeaders</name>
        <value>CONTENT-LENGTH,WWW-AUTHENTICATE</value>
      </param>
    </dispatch>
    <metadata>
      <context>/ssb-sse-ui-lb</context>
      <description>SSB-SSE-UI-LB</description>
      <shortDesc>SSB-SSE-UI-LB</shortDesc>
      <type>UI</type>
    </metadata>
    <routes>
      <route path="/ssb-sse-ui-lb/">
      <route path="/ssb-sse-ui-lb/**/">
      <route path="/ssb-sse-ui-lb/**?*">
      <route path="/ssb-sse-ui-lb/swagger/**">
        <rewrite apply="SSB-SSE-UI-LB/filter/outbound/swagger/body" to="response.body"/>
      </route>
    </routes>
  </service>
  <rules>
    <rule name="SSB-SSE-UI-LB/ssb-sse-ui-lb/inbound/root" pattern="*:*/*:*/*/**/ssb-sse-ui-lb/">
      <rewrite template="{ $serviceUrl[SSB-SSE-UI-LB] }/">
    </rule>
    <rule name="SSB-SSE-UI-LB/ssb-sse-ui-lb/inbound/path" pattern="*:*/*:*/*/**/ssb-sse-ui-lb/{**}">
      <rewrite template="{ $serviceUrl[SSB-SSE-UI-LB] }/{**}"/>
    </rule>
    <rule name="SSB-SSE-UI-LB/ssb-sse-ui-lb/inbound/query" pattern="*:*/*:*/*/**/ssb-sse-ui-lb/{**}?{**}">
      <rewrite template="{ $serviceUrl[SSB-SSE-UI-LB] }/{**}?{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/api" pattern="/api/{**}"/>
```

```

        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/api/{**}"/>
    >
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/ui" pattern="*://*:*/ui/{**}">
        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/ui/{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/frontend-ui" pattern="/ui/">
        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/ui/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/internal" pattern="/internal/">
        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/internal/">
    >
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/swagger-url" pattern="/swagger/{**}">
        <rewrite template="{ $frontend[path] }/ssb-sse-ui-lb/swagger/{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/swagger-config-url" pattern="/swagger-ui/{**}">
        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/swagger-ui/{**}"/>
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/html/websocket" pattern="/websocket/">
        <rewrite template="{ $frontend[url] }/ssb-sse-ws-lb/websocket/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/json/root">
        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/">
    </rule>
    <rule dir="OUT" name="SSB-SSE-UI-LB/rule/outbound/json/path">
        <rewrite template="{ $frontend[url] }/ssb-sse-ui-lb/{**}"/>
    </rule>
    <filter name="SSB-SSE-UI-LB/filter/outbound/swagger/body">
        <content type="*/json">
            <apply path="$.servers[0].url" rule="SSB-SSE-UI-LB/rule/outbound/json/root"/>
            <apply path="$.oauth2RedirectUrl" rule="SSB-SSE-UI-LB/rule/outbound/json/path"/>
        </content>
    </filter>
</rules>
</serviceDefinition>
</serviceDefinitions>

```

- b.** Paste it to the New Service Definition window.
- c.** Click Ok.
- d.** Click on the plus icon to create a new service definition.
- e.** Copy the following XML entry for the SSB-SSE-WS-LB service definition:

```

<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
    <serviceDefinition>
        <service name="ssb-sse-ws-lb" role="SSB-SSE-WS-LB" version="1.7.0">
            <dispatch classname="org.apache.knox.gateway.dispatch.ConfigurableDispatch" use-two-way-ssl="false">
                <param>
                    <name>responseExcludeHeaders</name>

```

```

        <value>CONTENT-LENGTH, WWW-AUTHENTICATE</value>
      </param>
    <param>
      <name>requestExcludeHeaders</name>
      <value>Cookie, Origin</value>
    </param>
  </dispatch>
  <metadata>
    <context>/ssb-sse-ws-lb</context>
    <description>SSB-SSE-WS-LB</description>
    <shortDesc>SSB-SSE-WS-LB</shortDesc>
    <type>API</type>
  </metadata>
  <routes>
    <route path="/ssb-sse-ws-lb/**">
      <rewrite apply="SSB-SSE-WS-LB/rule/inbound1" to="request.
url"/>
    </route>
  </routes>
</service>
<rules>
  <rule dir="IN" name="SSB-SSE-WS-LB/rule/inbound1" pattern="*://
*:*/**/ssb-sse-ws-lb/{path=**}">
    <rewrite template="{ $serviceUrl[SSB-SSE-WS-LB]}/{path=**}"/>
  </rule>
</rules>
</serviceDefinition>
</serviceDefinitions>

```

- f. Paste it to the New Service Definition window.
- g. Click Ok.
- h. Click on the plus icon to create a new service definition.
- i. Copy the following XML entry for the SSB-MVE-API-LB service definition:

```

<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
  <serviceDefinition>
    <service name="ssb-mve-api-lb" role="SSB-MVE-API-LB" version="1.
7.0">
      <metadata>
        <context>/ssb-mve-api-lb</context>
        <description>Streaming SQL Builder - Materialized View Engine
API LB</description>
        <shortDesc>SSB - MVE API LB</shortDesc>
        <type>API</type>
      </metadata>
      <routes>
        <route path="/ssb-mve-api-lb/**">
          <rewrite apply="SSB-MVE-API-LB/ssb-mve-api-lb/path" to="r
equest.url"/>
        </route>
      </routes>
    </service>
    <rules>
      <rule dir="IN" name="SSB-MVE-API-LB/ssb-mve-api-lb/path" pattern
="*://*:*/**/ssb-mve-api-lb/{path=**}?{**}">
        <rewrite template="{ $serviceUrl[SSB-MVE-API-LB]}/{path=**}?
{**}"/>
      </rule>
    </rules>
  </serviceDefinition>

```



```
</serviceDefinitions>
```

10. In the list of Service definitions, you should be able to see the SSB entries.

What to do next

After adding the SSB service definitions using the Knox Admin page, you need to configure the default topologies in Cloudera Manager.

Adding SSB services to the default topologies

You must add the SSB services to the Knox default topologies in Cloudera Manager.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Click on Knox from the list of Services.
3. Select Configuration.
4. Search for Knox Simplified Topology Management.
5. Add the following entries to the Knox Simplified Topology Management - cdp-proxy:

For Without Load Balancer

```
SSB-SSE-UI:url=https://[***STREAMING SQL ENGINE HOST***]:18121
SSB-SSE-UI:httpclient.connectionTimeout=5m
SSB-SSE-UI:httpclient.socketTimeout=5m
SSB-SSE-WS:url=wss://[***STREAMING SQL ENGINE HOST***]:18121
```

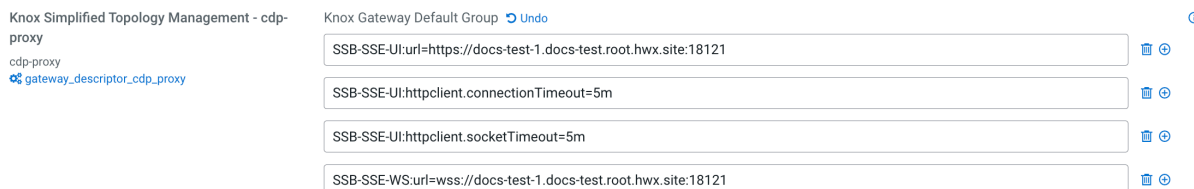
For With Load Balancer and without TLS

```
SSB-SSE-UI-LB:url=https://[***STREAMING SQL ENGINE HOST***]:8080
SSB-SSE-UI-LB:httpclient.connectionTimeout=5m
SSB-SSE-UI-LB:httpclient.socketTimeout=5m
SSB-SSE-WS-LB:url=wss://[***STREAMING SQL ENGINE HOST***]:8080
```

For With Load Balancer and with TLS

```
SSB-SSE-UI-LB:url=https://[***STREAMING SQL ENGINE HOST***]:8445
SSB-SSE-UI-LB:httpclient.connectionTimeout=5m
SSB-SSE-UI-LB:httpclient.socketTimeout=5m
SSB-SSE-WS-LB:url=wss://[***STREAMING SQL ENGINE HOST***]:8445
```

You need to add the hostname to the entries as shown in the following example:



6. Add the following entries to the Knox Simplified Topology Management - cdp-proxy-api:

```
SSB-SSE-API:url=https://[***STREAMING SQL ENGINE HOST***]:18121
```

The port for the SSB-SSE-API remains the same regardless of TLS configuration.

7. Add the following entries to the Knox Simplified Topology Management - cdp-proxy-api if you are using a Load Balanced SSB:

For Without TLS

```
SSB-MVE-API-LB:url=https://[***SSB MV HOST***]:8081
```

For With TLS

```
SSB-MVE-API-LB:url=https://[***SSB MV HOST***]:8444
```

8. Click Save changes.
The Refresh needed indicator appears beside the Knox service name.
9. Refresh Knox.

What to do next

When the default topologies are configured, you need to define the proxy paths for SSB in Cloudera Manager both for CDP Private Cloud Base 7.1.7 and 7.1.8.

Defining Knox proxy paths for SSB

You must provide the Knox proxy paths for YARN and the Materialized View API in Cloudera Manager to authenticate the user when accessing the Materialized Views and the Resource Manager through the Streaming SQL Console.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Click on SQL Stream Builder from the list of Services.
3. Select Configuration.
4. Search for Knox proxy path for YARN.
5. Add the following URL path:

```
https://[***KNOX GATEWAY HOST***]/gateway/cdp-proxy/yarnuiv2/proxy
```

6. Search for Knox proxy path for Materialized View Engine.
7. Add the following URL path:

```
https://[***KNOX GATEWAY HOST***]/gateway/cdp-proxy-api/ssb-mve-api
```

8. Restart the SQL Stream Builder service.

What to do next

After configuring the Knox service for SQL Stream Builder, you can reach the Streaming SQL Console by completing the steps in *Accessing the Streaming SQL Builder through Knox* section.

Accessing the Streaming SQL Console through Knox

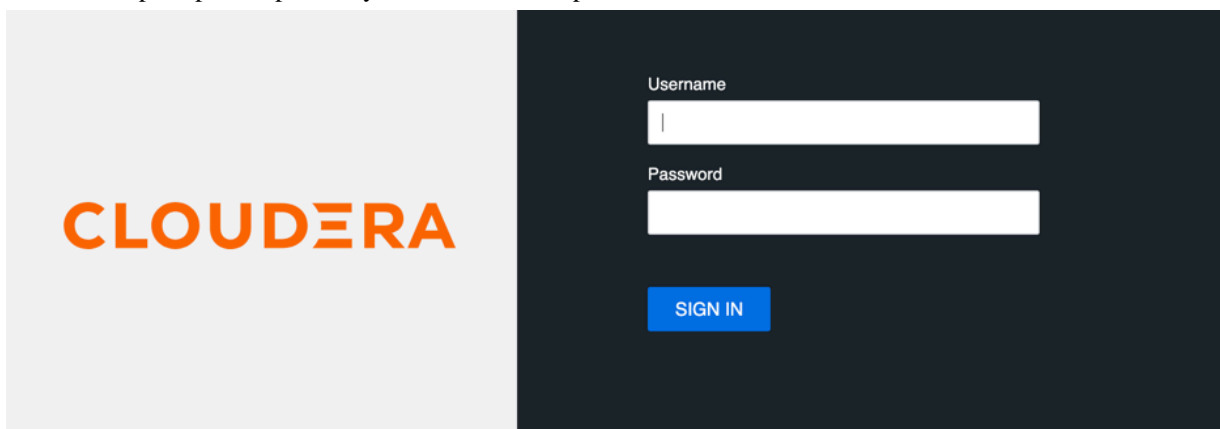
After configuring Knox and SSB, you should check if the SSO authentication works for the Streaming SQL Console.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Click on Knox from the list of Services.

3. Select Knox Gateway Home.

You will be prompted to provide your username and password.



4. Click cdp-proxy under Topologies.

SSB Console should be listed under the cdp-proxy.

5. Click SSB Console.

You are redirected to the Streaming SQL Console page.



Note: In case SSB Console does not appear under the cdp-proxy topologies, try to clean up the deployments folder of Knox with the following command:

```
rm -rf /var/lib/knox/gateway/data/deployments/*
```

After running the command, restart the Knox service.

Uploading or unlocking your keytab

After setting Kerberos or Knox authentication for SSB, you need to unlock the user specific keytabs on the Streaming SQL Console by providing your keytab password or uploading the keytab file. You cannot submit jobs until the keytab stays locked.

Procedure

1. Navigate to the Streaming SQL Console.

- Go to your cluster in Cloudera Manager.
- Select SQL Stream Builder from the list of services.
- Click SQLStreamBuilder Console .

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

2. Click your username on the sidebar of the Streaming SQL Console.

3. Click Manage keytab.

The **Keytab Manager** window appears.

You can either unlock the keytab already existing on the cluster, or you can directly upload your keytab file in the SQL Stream Builder.

- a) Unlock your keytab by providing the Principal Name and Password, and clicking Unlock Keytab. The Principal Name and Password should be the same as the workload username and password set for the Streaming Analytics cluster.

Keytab Manager

Unlock

Upload

Principal Name *

Password *

Lock Keytab

Cancel

Unlock Keytab

- b) Upload your keytab by clicking on the Upload tab, uploading the keytab file directly to the Console, and clicking Unlock Keytab.

Keytab Manager

Unlock Upload

Principal Name *



Choose File

No file chosen

Lock Keytab

Cancel

Upload Keytab

Encryption in SSB

When auto-TLS is disabled for the SQL Stream Builder (SSB) service, you must manually set the TLS properties for SSB in Cloudera Manager.

Before you begin

Ensure that you have set up Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)) for Cloudera Manager:

- Generated TLS certificates
- Configured TLS for Admin Console and Agents
- Enabled server certificate verification on Agents
- Configured agent certificate authentication
- Configured TLS encryption on the agent listening port

For more information, see the [Cloudera Manager](#) documentation.



Note: You can also configure the security parameters when adding SQL Stream Builder as a service using the Add Service Wizard.

Procedure

1. Click SQL Builder service on your Cluster.

2. Go to the Configuration tab.
3. Select Category > Security.
All the security related properties are displayed.
4. Edit the security properties according to the cluster configuration.



Note: You need to provide the keystore and truststore information for the Materialized View Engine, the SQL Stream Engine, and for the Streaming SQL Console.

Materialized View Engine	
Enable TLS/SSL for Materialized View Engine	Select the option to encrypt communication between clients and Materialized View Engine using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Materialized View Engine TLS/SSL Server JKS Keystore File Location	Path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Materialized View Engine is acting as a TLS/SSL server. The keystore must be in JKS format.
Materialized View Engine TLS/SSL Server JKS Keystore File Password	Password for the Materialized View Engine JKS keystore file.
Materialized View Engine TLS/SSL Server JKS Keystore Key Password	Password that protects the private key contained in the JKS keystore.
Materialized View Engine TLS/SSL Client Trust Store File	Location of the truststore on disk. The truststore must be in JKS format. If this parameter is not provided, the default list of known certificate authorities is used instead.
Materialized View Engine TLS/SSL Client Trust Store Password	The password for the Materialized View Engine TLS/SSL Certificate truststore file. This password is not mandatory to access the truststore; this field is optional. This password provides optional integrity checking of the file. The contents of truststores are certificates, and certificates are public information.

SQL Stream Engine	
Enable TLS/SSL for Streaming SQL Engine	Select the option to encrypt communication between clients and Streaming SQL Engine using Transport Layer Security (TLS) (formerly known as Secure Socket Layer (SSL)).
Streaming SQL Engine TLS/SSL Server JKS Keystore File Location	Path to the TLS/SSL keystore file containing the server certificate and private key used for TLS/SSL. Used when Streaming SQL Engine is acting as a TLS/SSL server. The keystore must be in JKS format.
Streaming SQL Engine TLS/SSL Server JKS Keystore File Password	Password for the Streaming SQL Engine JKS keystore file.
Streaming SQL Engine TLS/SSL Server JKS Keystore Key Password	Password that protects the private key contained in the JKS keystore used when Streaming SQL Engine is acting as a TLS/SSL server.
Streaming SQL Engine TLS/SSL Client Trust Store File	Location on disk of the truststore, in .jks format, used to confirm the authenticity of TLS/SSL servers that Streaming SQL Engine might connect to. This is used when Streaming SQL Engine is the client in a TLS/SSL connection. This truststore must contain the certificate(s) used to sign the service(s) connected to. If this parameter is not provided, the default list of known certificate authorities is used instead.
Streaming SQL Engine TLS/SSL Client Trust Store Password	Password for the Streaming SQL Engine TLS/SSL Certificate Trust Store file. This password is not mandatory to access the trust store; this field is optional. This password provides optional integrity checking of the file. The contents of truststores are certificates, and certificates are public information.

5. Click Save Changes.

Enabling TLS for database connection

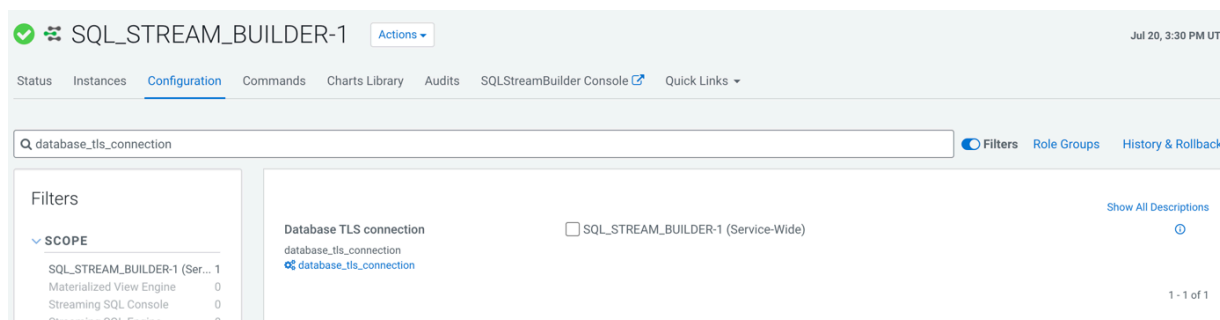
You can enable encrypted communication between SSB and the configured databases using the Database connection property in Cloudera Manager.



Note: When you are using Auto-TLS the configuration for the Database TLS connection is not enabled automatically.

Procedure

1. Go to your cluster in Cloudera Manager.
2. Click SQL Stream Builder from the list of services.
3. Select Configuration tab.
4. Add database_tls_connection to the Search box.



5. Select Database TLS connection to enable encryption for the databases.
6. Click Save changes.
7. Restart the SQL Stream Builder service.
 - a) Click Action > Restart next to the SQL Stream Builder service name.

Results

Encrypted communication is enabled between SSB and the configured databases.

What to do next

You must rotate the Certificate Authority and Host Certificates to update the security requirements of your cluster. For more information, see the [Rotate Auto-TLS Certificate Authority and Host Certificates](#) in Cloudera Manager documentation.

Configuring Ranger policies for SSB

You must add SQL Stream Builder (SSB) service user named ssb to the Ranger policies that are used by Kafka, Schema Registry, Hive and Kudu to provide access to topics, schemas and tables provided by the components.

Before you begin

- Install Apache Ranger on your cluster. For more information, see the [Production Installation](#) documentation.
- Add the required Ranger policies for Flink. For more information, see [Configuring Ranger policies for Flink](#) documentation.

You can reach the Ranger User Interface through Cloudera Manager:

1. Go to your cluster in Cloudera Manager.
2. Select Ranger from the list of services.
3. Click on Ranger Admin Web UI.

You are redirected to the Ranger Admin Web UI.

Adding SSB user to Kafka policies

You must add the ssb user to the following policies:

- all-consumergroup
- all-topic

1. Select `cm_kafka` from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the list of Kafka policies page.

2. Click on the edit button of the *all-consumergroup* policy.
3. Add the SSB user to the Select User field under the Allow Conditions setting.
4. Click Save.

You are redirected to the list of Kafka policies page

5. Click on + More... to check if the SSB user is listed under the Users for the consumergroup policy.
6. Add the ssb user to the following policy with the above steps as well:
 - all-topic

Adding SSB user to Schema Registry policies

You must add the ssb user to the following policy:

- all-schema-group, schema-metadata, schema-branch, schema-version

1. Select `cm_schema-registry` from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the list of Schema Registry policies page.

2. Click on the edit button of the all-schema-group, schema-metadata, schema-branch, schema-version policy.
3. Add the ssb user to the Select User field under the Allow Conditions setting.
4. Click Save.

You are redirected to the list of Schema Registry policies page.

5. Click on + More... to check if the SSB user is listed under the Users for the schema-group, schema-metadata, schema-branch, schema-version policy.

Adding SSB user to Hive policies

You must add the ssb user to the following policy:

- all-global
- all-database, table, column
- all-database, table
- all-database
- all-hiveservice
- all-database, udf
- all-url

1. Select `cm_hadoopsql` from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the list of Hadoop SQL policies page.

2. Click on the edit button of the all-global policy.
3. Add the SSB user to the Select User field under the Allow Conditions setting.
4. Click Save.

You are redirected to the list of Hadoop SQL policies page.

5. Click on + More... to check if the SSB user is listed under the Users for the all-global policy.

6. Add the ssb user to the following policy with the above steps as well:

- all-database, table, column
- all-database, table
- all-database
- all-hiveservice
- all-database, udf
- all-url

Adding SSB user to Kudu policies

You must create a policy to grant access to Kudu tables for the ssb user.

1. Select `cm_kudu` from the Service Manager home page on the Ranger Admin Web UI.

You are redirected to the Create Policy page.

2. Click on Add New Policy.

3. Provide a name to the Policy Name field.

4. Provide a prefix for the Databases you want to add to the policy or type `*` to select all.

5. Provide a prefix for the table you want to add to the policy or type `*` to select all.

6. Provide a prefix for the column you want to add to the policy or type `*` to select all.

7. Add the ssb user to the Select User field under the Allow Conditions setting.

8. Click on the plus icon to Add Permissions to the Permissions field.

9. Click on the specific permissions or Select All.

10. Click on Add at the bottom of the page.

You are redirected to the list of Kudu policies page where the created policy should be listed.

11. Click on + More... to check if the ssb user is listed under the Users for the created policy.