

Securing Streaming Analytics Clusters

Date published: 2019-12-16

Date modified: 2026-09-30



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

Securing Apache Flink.....	4
Enabling security for Apache Flink.....	4
Securing Apache Flink jobs.....	5
Securing Flink external REST endpoints with TLS.....	7
Configuring TLS/SSL encryption manually for Flink services.....	8

Securing Apache Flink

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 [Enabling Kerberos Authentication for Cloudera](#).

TLS encryption

If AutoTLS is enabled on the cluster, the TLS-related configuration fields can be auto-populated for the Flink History Server (global Flink Dashboard). Flink is deployed as a Gateway-type service, so Cloudera Manager does not create keystores for Flink internal or REST endpoints. You can set `{{CM_AUTO_TLS}}` as the value for Dashboard and Gateway security properties when using AutoTLS in Cloudera Manager. Configure internal and REST TLS for running Flink jobs separately. 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.

Security property	Description
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.

What to do next

To configure TLS on Flink external REST endpoints for direct client access to running jobs, see [Securing Flink external REST endpoints with TLS](#). To configure TLS for Flink internal and REST services cluster-wide with manually distributed keystores, see [Configuring TLS/SSL encryption manually for Flink services](#).

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.

Running secure Flink Deployments

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 Cloudera Private Cloud Base environments.

For TLS on external REST endpoints when clients connect directly to Flink, see [Securing Flink external REST endpoints with TLS](#).

Using SPNEGO authentication with Kerberos for Flink Deployments

By default on a kerberized cluster the Flink dashboard is secured by the YARN proxy, which means direct external access to the Flink deployments should be disabled.

Alternatively, it is possible to directly enable SPNEGO for the Flink Dashboard. To secure a session cluster, use the following configuration options:

```
flink-yarn-session -d -nm [***CLUSTER-NAME***] \
-Dsecurity.kerberos.login.use-ticket-cache=true \
-Dsecurity.spnego.auth.enabled=true \
-Dsecurity.spnego.auth.keytab=[***SERVICE-KEYTAB***] \
-Dsecurity.spnego.auth.principal=[***HTTP PRINCIPAL***] \
-Dyarn.ship-files=[***SERVICE-KEYTAB***]
```



Note:

Using the HTTP principal is required for SPNEGO to work correctly.

The security.spnego.auth.principal configuration option can also be set to the * wildcard character, so that the HTTP principal can be used for multiple hosts that are included in the keytab.



Important:

The provided keytab must include all the hosts where the cluster can be deployed to.

To run a Flink job on the previously created session cluster, use the following command:

```
flink run \
-d -t yarn-session \
-yid [***YARN APPLICATION ID***] \
-yD security.spnego.auth.enabled=true \
-yD security.kerberos.login.keytab=[***USER-KEYTAB***] \
-yD security.kerberos.login.principal=[***USER-PRINCIPAL***] \
[***FULL PATH***][***FILENAME***].jar
```

Securing Flink external REST endpoints with TLS

Configure TLS for Flink external REST endpoints when clients connect directly to the Flink REST API or web dashboard. Internal RPC and data transport traffic uses separate security.`security.ssl.internal.*` settings.

About this task

Flink exposes two classes of TLS-protected endpoints:

- Internal endpoints — RPC, blob server, and data transport between Flink processes. Configure these with `security.ssl.internal.*` properties when you submit jobs. For an example, see [Securing Apache Flink jobs](#).
- External REST endpoints — the REST API and web dashboard that operators and clients reach from outside the Flink process. Configure these with `security.ssl.rest.*` properties.

On a Kerberos-enabled cluster, the YARN proxy or Knox Gateway typically fronts the Flink dashboard. If you disable direct external access, you might not need explicit REST TLS on every job. Enable REST TLS when users, monitoring tools, or automation connect directly to the JobManager REST port or TaskManager REST endpoints.



Note: Service-level TLS for the Flink History Server (global Flink Dashboard) in Cloudera Manager is separate from per-job internal and REST TLS. For cluster-wide Dashboard TLS, see [Enabling security for Apache Flink](#). Flink is deployed as a Gateway-type service, so when Auto-TLS is enabled, Cloudera Manager can auto-populate keystore and truststore paths for the Flink History Server only. You must create and distribute your own keystores for Flink internal endpoints and external REST endpoints on running jobs. To configure TLS for all Flink services cluster-wide, see [Configuring TLS/SSL encryption manually for Flink services](#).

Procedure

1. Enable REST TLS on the Flink cluster or job submission.

Set the following Flink configuration properties on the JobManager and TaskManagers. You can pass them with `-yD` on `flink run` or in a `flink-conf.yaml` fragment shipped with the application:

```
security.ssl.rest.enabled: true
security.ssl.rest.keystore: /path/to/rest-keystore.jks
security.ssl.rest.keystore-password: <password>
security.ssl.rest.key-password: <password>
security.ssl.rest.truststore: /path/to/rest-truststore.jks
security.ssl.rest.truststore-password: <password>
```

Optional properties include `security.ssl.rest.keystore-type`, `security.ssl.rest.truststore-type`, `security.ssl.rest.authentication-enabled` for mutual TLS, and `security.ssl.rest.cert.fingerprint` to pin the REST certificate.

2. Align TLS protocol settings with your cluster policy.

Cloudera Streaming Analytics services honor the central TLS protocol and cipher settings from Cloudera Manager. The Flink `security.ssl.protocol` property accepts a comma-separated list such as `TLSv1.3`. Ensure client libraries and browsers support the configured protocol.

3. Distribute keystore and truststore files to hosts that run Flink containers.

Ship keystores with `-yt` on `flink run` or place them on a path accessible to YARN containers, similar to internal TLS setup in [Securing Apache Flink jobs](#).

4. Verify REST connectivity over HTTPS.

Confirm that the Flink dashboard and REST API respond on the HTTPS port and that clients trust the REST certificate. If you access Flink through Knox Gateway, configure Knox Gateway to trust the Flink REST certificate or terminate TLS at the gateway.

What to do next

For a full secured Flink deployment walkthrough, see the [Flink secure tutorial](#) and [Securing Apache Flink jobs](#).

Configuring TLS/SSL encryption manually for Flink services

Enable TLS for Flink internal and REST endpoints across the cluster when Cloudera Manager cannot yet generate Flink keystores automatically. Configure central protocol and cipher settings, distribute host keystores, and apply Flink safety-valve properties.

About this task

Starting in Cloudera Runtime 7.3.2 SP1 (7.3.2), Cloudera Manager can control TLS protocols and cipher suites for Flink, Cloudera SQL Stream Builder, and Knox Gateway when you enable the CMF_FF_TLS_ADVANCED_CONTROL feature flag. For background on the flag and global TLS settings, see [Transport Layer Security Support and Custom Protocol Configuration](#).

Flink is deployed as a Gateway-type service. Cloudera Manager can auto-generate keystores for Cloudera SQL Stream Builder, but Flink keystore generation in Cloudera Manager is not available in this release. Until that support ships, use this procedure to configure TLS for Flink internal and REST services cluster-wide. For per-job REST TLS when you submit applications, see [Securing Flink external REST endpoints with TLS](#).

Procedure

1. Enable advanced TLS control and configure protocol and cipher settings for Flink and Cloudera SQL Stream Builder.
 - a) Enable the CMF_FF_TLS_ADVANCED_CONTROL feature flag on the Cloudera Manager Server host, as described in [Transport Layer Security Support and Custom Protocol Configuration](#).
 - b) On the Flink and Cloudera SQL Stream Builder service configuration pages in Cloudera Manager, set Supported SSL/TLS versions and TLS Cipher List to match your cluster policy.

Cloudera Manager writes these values to the corresponding service configuration properties and propagates them into `flink-conf.yaml` as `security.ssl.protocol` and `security.ssl.algorithms`. Flink and Cloudera SQL Stream Builder ports use these settings during the TLS handshake.

2. Generate a signed keystore for each host that runs Flink containers.

For each host, create a keystore that meets the following requirements:

- Sign the certificate with your root CA.
- Include a Subject Alternative Name (SAN) that lists the hostname, fully qualified domain name (FQDN), and IP addresses used by Flink services on that host.
- Set Extended Key Usage (EKU) to include both `serverAuth` and `clientAuth`.

3. Distribute the host-specific keystore files to every Flink node.

Copy each keystore to the same persistent directory path on every host and use the same file name on all nodes. Flink shares a single `flink-conf.yaml` configuration, so every Flink service on every node must resolve the keystore from an identical path.

4. Configure Flink to use the distributed keystores for internal and REST TLS.
 - a) In Cloudera Manager, open the Flink service Configuration page.
 - b) Locate `flink-conf/flink-conf.yaml_service_safety_valve` and add the following properties. Replace `YOUR_KEYSTORE_PASSWORD` and `YOUR_KEYSTORE_COMMON_PATH` with the password and path you used when you distributed the keystores in the previous step. Use the same values on every node.
 - c) Read the global truststore path and password from the deployed Flink configuration, for example:

```
grep truststore /etc/flink/conf/flink-conf.yaml
```

- d) Substitute those values for CM_GLOBAL_TRUSTSTORE_PATH and CM_GLOBAL_TRUSTSTORE_PASSWORD in the safety-valve block below.

```
security.ssl.rest.enabled: 'true'
security.ssl.rest.key-password: YOUR_KEYSTORE_PASSWORD
security.ssl.rest.keystore: YOUR_KEYSTORE_COMMON_PATH
security.ssl.rest.keystore-password: YOUR_KEYSTORE_PASSWORD
security.ssl.internal.enabled: 'true'
security.ssl.internal.key-password: YOUR_KEYSTORE_PASSWORD
security.ssl.internal.keystore: YOUR_KEYSTORE_COMMON_PATH
security.ssl.internal.keystore-password: YOUR_KEYSTORE_PASSWORD
security.ssl.internal.truststore: CM_GLOBAL_TRUSTSTORE_PATH
security.ssl.internal.truststore-password: CM_GLOBAL_TRUSTSTORE_PASSWORD
```

When Cloudera Manager supports automatic Flink keystore generation in a future release, you will not need this manual safety-valve configuration.

5. Add your root CA certificate to the Cloudera Manager global truststore if it is not already present.
 - a) Go to AdministrationSecurityUpdate Auto-TLS Truststore.
 - b) Leave Certificate Location empty.
 - c) Choose Certificate File and upload your root CA certificate from your local machine.
 - d) Set Update mode to Append to existing truststore.

For more information, see [Updating the Auto-TLS Global Truststore](#).

6. Deploy the client configuration and restart dependent services.
 - a) Deploy the client configuration for the cluster.
 - b) Restart the following services:
 - Flink
 - YARN
 - Cloudera SQL Stream Builder
 - Knox Gateway

7. Verify encrypted Flink endpoints.

Submit or restart Flink jobs and confirm that Flink services expose encrypted internal and REST ports using the TLS settings you configured. Clients must trust the certificates in your distributed keystores and the Cloudera Manager global truststore.

What to do next

For Flink History Server (global Dashboard) TLS through Cloudera Manager, see [Enabling security for Apache Flink](#).