Cloudera Streaming Analytics 1.4.1

Security

Date published: 2019-12-17
Date modified: 2021-07-20

CLOUDERA

https://docs.cloudera.com/

Legal Notice

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

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.

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.

The Auto Discovery for Knox service is not yet available for the Flink Dashboard. This means you must manually configure Knox with the following steps:

- Add the Flink Dashboard as a custom service to the cdp-proxy and cdp-proxi-api configurations
- Create the Flink Dashboard service definitions in Knox

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.

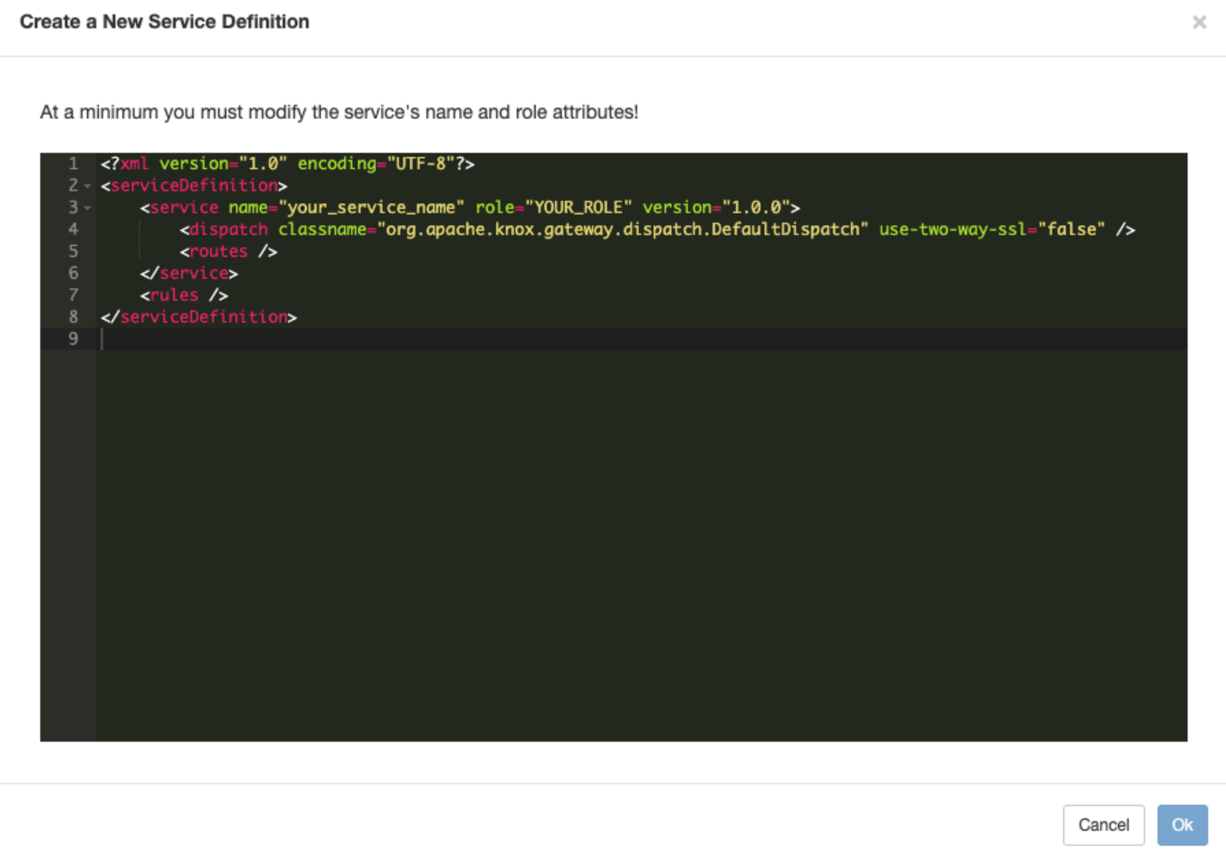
Adding Flink Dashboard to Knox Topology Management

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!

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinition>
    <service name="your_service_name" role="YOUR_ROLE" version="1.0.0">
        <dispatch classname="org.apache.knox.gateway.dispatch.DefaultDispatch" use-two-way-ssl="false" />
        <routes />
    </service>
    <rules />
</serviceDefinition>
```

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

   a. Copy the following XML entry for the FLINK service definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinitions>
   <serviceDefinition>
      <service name="flink" role="FLINK" version="1.12.1">
         <dispatch classname="org.apache.knox.gateway.dispatch.Configu
rableDispatch" use-two-way-ssl="false">
            <param>
               <name>responseExcludeHeaders</name>
               <value>CONTENT-LENGTH,Www-Authenticate</value>
            </param>
         </dispatch>
         <metadata>
            <context>/flink/</context>
            <description>The Flink Dashboard acts as a single UI for all
 the Flink jobs running on the YARN cluster.</description>
            <shortDesc>Flink Dashboard</shortDesc>
            <type>UI</type>
         </metadata>
         <routes>
            <route path="/flink/"/>
            <route path="/flink/**"/>
            <route path="/flink/**?**"/>
            <route path="/flink/jobs/overview">
```

```
               <rewrite apply="FLINK/flink/outbound/json" to="response
.body"/>
            </route>
         </routes>
      </service>
      <rules>
         <rule name="FLINK/flink/inbound/root" pattern="*://*:*/**/flink
/">
            <rewrite template="{$serviceUrl[FLINK]}/"/>
         </rule>
         <rule name="FLINK/flink/inbound/path" pattern="*://*:*/**/flin
k/{**}">
            <rewrite template="{$serviceUrl[FLINK]}/{**}"/>
         </rule>
         <rule name="FLINK/flink/inbound/query" pattern="*://*:*/**/fl
ink/{path=**}?{**}">
            <rewrite template="{$serviceUrl[FLINK]}/{path=**}?{**}"/>
         </rule>
         <rule dir="OUT" name="FLINK/flink/outbound/links">
            <match pattern="*://*:*/proxy/{**}"/>
            <rewrite template="{$frontend[url]}/yarnuiv2/proxy/{**}/"/>
         </rule>
         <filter name="FLINK/flink/outbound/json">
            <content type="*/json">
               <apply path="$.jobs[*].cluster.url" rule="FLINK/flink/o
utbound/links"/>
            </content>
         </filter>
      </rules>
   </serviceDefinition>
</serviceDefinitions>
```

b. Paste it to the New Service Definition window.

c. Click Ok.

In the list of Service definitions, you should be able to see the Flink service definition.

Adding Flink Dashboard to Knox Topology Management

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 entry to the Knox Simplified Topology Management - cdp-proxy:

```
FLINK:url=https://<your_hostname>:18211
```

6. Click on Save changes.

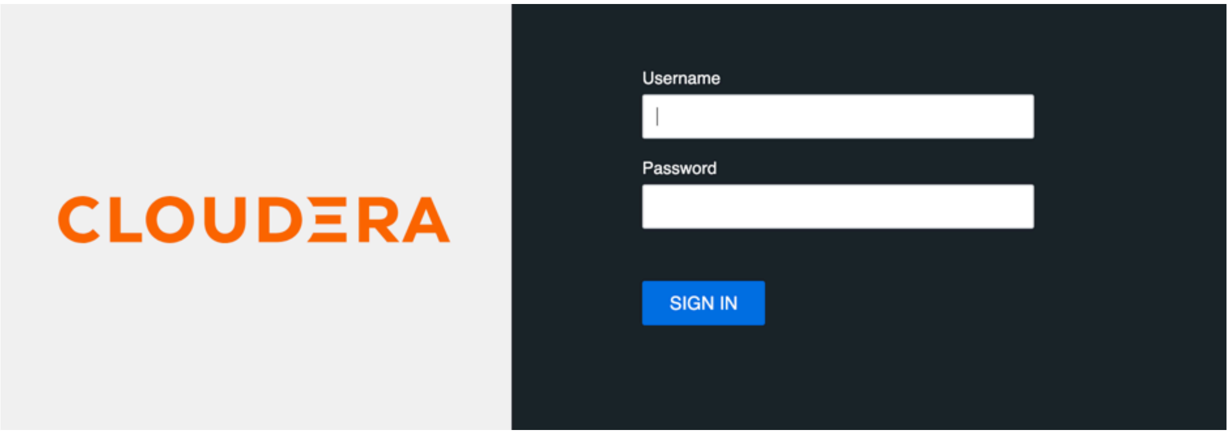
   The Refresh needed indicator appears beside the Knox service name.
7. Refresh Knox.

Reaching the Flink Dashboard through Knox

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. Clink 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. Clink 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 select all by typing *.
5. Provide a prefix for the table you want to add to the policy or select all by typing *.
6. Provide a prefix for the column you want to add to the policy or select all by typing *.
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 -storetyp
e 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