

Securing Apache Hive

Date published: 2021-02-22

Date modified: 2021-06-08



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

Transactional table access.....	4
Hive authentication.....	4
Secure HiveServer using LDAP.....	4
Client connections to HiveServer.....	5
JDBC connection string syntax.....	7
Under AutoTLS secure a HiveServer JDBC/ODBC endpoint.....	9

Transactional table access

As administrator, you must enable the Apache Ranger service to authorize users who want to work with transactional tables. These types of tables are the default, ACID-compliant tables in Hive 3 and later.

ACID tables reside by default in `/warehouse/tablespace/managed/hive`. Only the Hive service can own and interact with files in this directory. Ranger is the only available authorization mechanism that Cloudera recommends for ACID tables.

Hive authentication

HiveServer supports authentication of clients using Kerberos or user/password validation backed by LDAP.

If you configure HiveServer to use Kerberos authentication, HiveServer acquires a Kerberos ticket during startup. HiveServer requires a principal and keytab file specified in the configuration. Client applications (for example, JDBC or Beeline) must have a valid Kerberos ticket before initiating a connection to HiveServer2. JDBC-based clients must include `principal=<hive.server2.authentication.principal>` in the JDBC connection string. For example:

```
String url = "jdbc:hive2://node1:10000/default;principal=hive/HiveServerHost@YOUR-REALM.COM"
Connection con = DriverManager.getConnection(url);
```

where `hive` is the principal configured in `hive-site.xml` and `HiveServerHost` is the host where HiveServer is running.

To start Beeline and connect to a secure HiveServer, enter a command as shown in the following example:

```
beeline -u "jdbc:hive2://10.65.13.98:10000/default;principal=hive/_HOST@CLOU
DERA.SITE"
```

Secure HiveServer using LDAP

You can secure the remote client connection to Hive by configuring HiveServer to use authentication with LDAP.

About this task

When you configure HiveServer to use user and password validation backed by LDAP, the Hive client sends a username and password during connection initiation. HiveServer validates these credentials using an external LDAP service. You can enable LDAP Authentication with HiveServer using Active Directory or OpenLDAP.

Procedure

1. In Cloudera Manager, select `Hive-on-Tez Configuration`.
2. Search for `ldap`.
3. Check `Enable LDAP Authentication for HiveServer2 for Hive (Service Wide)`.
4. Enter your LDAP URL in the format `ldap[s]://<host>:<port>`.

LDAP_URL is the access URL for your LDAP server. For example, `ldap://ldap_host_name.xyz.com:389`

5. Enter the Active Directory Domain or LDAP Base DN for your environment.

- Active Directory (AD)
- LDAP_BaseDN

Enter the domain name of the AD server. For example, corp.domain.com.

Enable LDAP Authentication for HiveServer2 ☒ Hive (Service-Wide) [Undo](#) [?](#)

LDAP URL [?](#)
hive.server2.authentication.ldap.url

Active Directory Domain [?](#)
hive.server2.authentication.ldap.Domain

Enter the base LDAP distinguished name (DN) for your LDAP server. For example, ou=dev, dc=xyz.

Enable LDAP Authentication for HiveServer2 ☒ Hive (Service-Wide) [Undo](#) [?](#)

LDAP URL [?](#)
hive.server2.authentication.ldap.url

Active Directory Domain [?](#)
hive.server2.authentication.ldap.Domain

LDAP BaseDN [?](#)
hive.server2.authentication.ldap.baseDN

6. Click Save Changes.
7. Restart the Hive service.
8. Construct the LDAP connection string to connect to HiveServer.
The following simple example is insecure because it sends clear text passwords.

```
String URL = "jdbc:hive2://node1:10000/default;user=LDAP_Userid;password=LDAP_Password";
Connection con = DriverManager.getConnection(url);
```

The following example shows a secure connection string that uses encrypted passwords.

```
String url = "jdbc:hive2://node1:10000/default;ssl=true;sslTrustStore=/my_truststore_path;trustStorePassword=my_truststore_password";
Connection con = DriverManager.getConnection(url);
```

For information about encrypting communication, see links below.

Related Information

[Custom Configuration \(about Cloudera Manager Safety Valve\)](#)

Client connections to HiveServer

You can use Beeline, a JDBC, or an ODBC connection to HiveServer.

JDBC Client-HiveServer Authentication

The JDBC client requires a connection URL as shown below. JDBC-based clients must include a user name and password in the JDBC connection string. For example:

```
String url = "jdbc:hive2://node1:10000/default;user=LDAP_Userid;password=LDAP_Password"
Connection con = DriverManager.getConnection(url);
```

where the LDAP_Userid value is the user ID and LDAP_Password is the password of the client user.

HiveServer modes of operation

HiveServer supports the following modes for interacting with Hive:

Operating Mode	Description
Embedded	The Beeline client and the Hive installation reside on the same host machine or virtual machine. No TCP connectivity is required.
Remote	Use remote mode to support multiple, concurrent clients executing queries against the same remote Hive installation. Remote transport mode supports authentication with LDAP and Kerberos. It also supports encryption with SSL. TCP connectivity is required.

Remote mode: Launch Hive using the following URL:

```
jdbc:hive2://<host>:<port>/<db>.
```

The default HiveServer2 port is 10000.

Embedded mode: Launch Hive using the following URL:

```
jdbc:hive2://
```

Transport Modes

As administrator, you can start HiveServer in one of the following transport modes:

Transport Mode	Description
TCP	HiveServer uses TCP transport for sending and receiving Thrift RPC messages.
HTTP	HiveServer uses HTTP transport for sending and receiving Thrift RPC messages.

Pluggable Authentication Modules in HiveServer

While running in TCP transport mode, HiveServer supports Pluggable Authentication Modules (PAM). Using Pluggable Authentication Modules, you can integrate multiple authentication schemes into a single API. You use the Cloudera Manager Safety Valve technique on [HIVE_ON_TEZ-1 Configuration](#) to set the following properties:

- hive.server2.authentication
Value = CUSTOM
- hive.server2.custom.authentication.class
Value = <the pluggable auth class name>

The class you provide must be a proper implementation of the `org.apache.hive.service.auth.PasswdAuthenticationProvider`. HiveServer calls its `Authenticate(user, passed)` method to authenticate requests. The implementation can optionally extend the Hadoop's `org.apache.hadoop.conf.Configured` class to grab the Hive Configuration object.

HiveServer Trusted Delegation

HiveServer determines the identity of the connecting user from the authentication subsystem (Kerberos or LDAP). Any new session started for this connection runs on behalf of this connecting user. If the server is configured to proxy the user, the identity of the connecting user is used to connect to Hive. Users with Hadoop superuser privileges can request an alternate user for the given session. HiveServer checks that the connecting user can proxy the requested userid, and if so, runs the new session as the alternate user.

JDBC connection string syntax

The JDBC connection string for connecting to a remote Hive client requires a host, port, and Hive database name. You can optionally specify a transport type and authentication.

```
jdbc:hive2://<host>:<port>/<dbName>;<sessionConfs>?<hiveConfs>#<hiveVars>
```

Connection string parameters

The following table describes the parameters for specifying the JDBC connection.

JDBC Parameter	Description	Required
host	The cluster node hosting HiveServer.	yes
port	The port number to which HiveServer listens.	yes
dbName	The name of the Hive database to run the query against.	yes
sessionConfs	Optional configuration parameters for the JDBC/ODBC driver in the following format: <key1>=<value1>;<key2>=<key2>...;	no
hiveConfs	Optional configuration parameters for Hive on the server in the following format: <key1>=<value1>;<key2>=<key2>; ... The configurations last for the duration of the user session.	no
hiveVars	Optional configuration parameters for Hive variables in the following format: <key1>=<value1>;<key2>=<key2>; ... The configurations last for the duration of the user session.	no

TCP and HTTP Transport

The following table shows variables for use in the connection string when you configure HiveServer. The JDBC client and HiveServer can use either HTTP or TCP-based transport to exchange RPC messages. Because the default transport is TCP, there is no need to specify transportMode=binary if TCP transport is desired.

transportMode Variable Value	Description
http	Connect to HiveServer2 using HTTP transport.
binary	Connect to HiveServer2 using TCP transport.

The syntax for using these parameters is:

```
jdbc:hive2://<host>:<port>/<dbName>;transportMode=http;httpPath=<http_endpoint>; \
<otherSessionConfs>?<hiveConfs>#<hiveVars>
```

User Authentication

If configured in remote mode, HiveServer supports Kerberos, LDAP, Pluggable Authentication Modules (PAM), and custom plugins for authenticating the JDBC user connecting to HiveServer. The format of the JDBC connection URL for authentication with Kerberos differs from the format for other authentication models. The following table shows the variables for Kerberos authentication.

User Authentication Variable	Description
principal	A string that uniquely identifies a Kerberos user.
saslQop	Quality of protection for the SASL framework. The level of quality is negotiated between the client and server during authentication. Used by Kerberos authentication with TCP transport.
user	Username for non-Kerberos authentication model.
password	Password for non-Kerberos authentication model.

The syntax for using these parameters is:

```
jdbc:hive://<host>:<port>/<dbName>;principal=<HiveServer2_kerberos_principal>;<otherSessionConfs>?<hiveConfs>#<hiveVars>
```

Transport Layer Security

HiveServer2 supports SSL and Sasl QOP for transport-layer security. The format of the JDBC connection string for SSL uses these variables:

SSL Variable	Description
ssl	Specifies whether to use SSL.
sslTrustStore	The path to the SSL TrustStore.
trustStorePassword	The password to the SSL TrustStore.

The syntax for using the authentication parameters is:

```
jdbc:hive2://<host>:<port>/<dbName>; \
ssl=true;sslTrustStore=<ssl_truststore_path>;trustStorePassword=<truststore_password>; \
<otherSessionConfs>?<hiveConfs>#<hiveVars>
```

When using TCP for transport and Kerberos for security, HiveServer2 uses Sasl QOP for encryption rather than SSL.

Sasl QOP Variable	Description
principal	A string that uniquely identifies a Kerberos user.
saslQop	The level of protection desired. For authentication, checksum, and encryption, specify auth-conf. The other valid values do not provide encryption.

The JDBC connection string for Sasl QOP uses these variables.

```
jdbc:hive2://fqdn.example.com:10000/default;principal=hive/_HOST@EXAMPLE.COM;saslQop=auth-conf
```

The `_HOST` is a wildcard placeholder that gets automatically replaced with the fully qualified domain name (FQDN) of the server running the HiveServer daemon process.

Under AutoTLS secure a HiveServer JDBC/ODBC endpoint

The default cluster configuration for HiveServer (HS2) with AutoTLS secures the HS2 WebUI Port, but NOT the JDBC/ODBC endpoint.

About this task

The default cluster configuration for HS2 with AutoTLS will secure the HS2 Server WebUI Port, but NOT the JDBC/ODBC endpoint.

Assumptions:

- Auto-TLS Self-signed Certificates.
- Proper CA Root certs eliminate the need for any of the following truststore actions.

When HS2 TLS is enabled `hive.server2.use.SSL=true`, the auto-connect feature on gateway servers is not supported. The auto-connect feature uses `/etc/hive/conf/beeline-site.xml` to automatically connect to Cloudera Manager controlled HS2 services. Also, with `hive.server2.use.SSL=true`, ZooKeeper discovery mode is not supported because the HS2 reference stored in ZooKeeper does not include the `ssl=true` and other TLS truststore references (self-signed) needed to connect with TLS.

The `beeline-site.xml` file managed for gateways doesn't include `ssl=true` or a reference to a truststore that includes a CA for the self-signed TLS certificate used by ZooKeeper or HiveServer.

The best practice, under the default configuration, is to have all external clients connect to Hive (JDBC/ODBC) through the Apache Knox proxy. With TLS enabled via Auto-TLS with a self-signed cert, you can use the `jks` file downloaded from Knox as the client trusted CA for the Knox host. That cert will only work for KNOX. And since KNOX and HS2 TLS server certs are from the same CA, Knox connects without adjustments.

To connect through Knox:

Procedure

1. Configure the HS2 transport mode as `http` to support the Knox proxy interface.

```
jdbc:hive2://<host>:8443/;ssl=true;\
transportMode=http;httpPath=gateway/cdp-proxy-api/hive;\
...
```

The TLS Public Certificate in `<path>/bin/certs/gateway-client-trust.jks` will not work.

2. Build a TLS Public Certificate from the self-signed root CA used for the cluster in Cloudera Manager.

```
keytool -import -v -trustcacerts -alias home90-ca -file \
/var/lib/cloudera-scm-agent/agent-cert/cm-auto-global_cacerts.pem \
-keystore <my_cacert>.jks
```

3. Connect to HS2 on the Beeline command line, using the `-u` option.

```
hive -u jdbc:hive2://<HS2 host>:10001/default;ssl=true;\
transportMode=http;httpPath=cliservice;\
principal=hive/_HOST@<realm>;user=<user name>;\
sslTrustStore=<path>/certs/home90_cacert.jks;\
trustStorePassword=changeit
```

The `httpPath` default is configured in Cloudera Manager. The `sslTrustStore` is required if you are using a self-signed certificate.