Cloudera Runtime 7.2.12

Securing Apache Hive

Date published: 2019-08-21
Date modified: 2021-10-20

https://docs.cloudera.com/

Legal Notice

Contents

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.

External table access

As administrator, you must set up Apache Ranger to allow users to access external tables.

External tables reside by default in /warehouse/tablespace/external on HDFS in CDP Private Cloud Base). To specify some other location of the external table, you need to include the specification in the table creation statement as shown in the following example:

```
CREATE EXTERNAL TABLE my_external_table (a string, b string)
LOCATION '/users/andrena';
```

Hive assigns a default permission of 777 to the hive user, sets a umask to restrict subdirectories, and provides a default ACL to give Hive read and write access to all subdirectories. External tables in CDP Private Cloud Base must be secured using Ranger.

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"
```

Securing HiveServer using LDAP

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

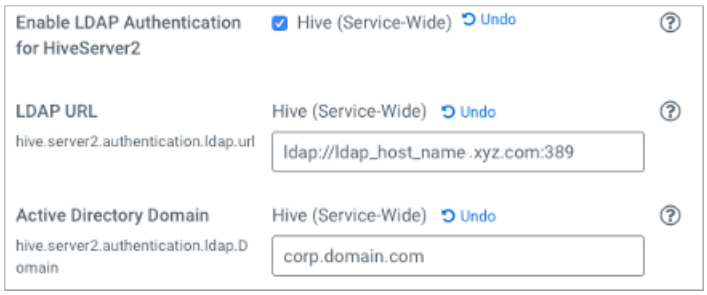
About this task

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.

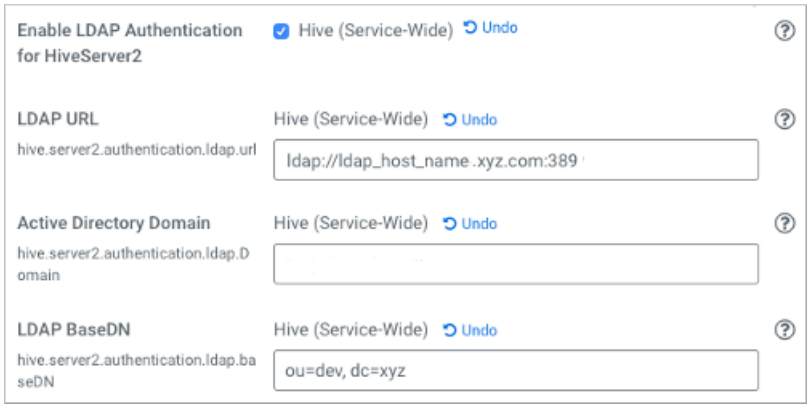
Procedure

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.



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



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=LDA
P_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_endpoi
nt>; \
  <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=<truststo
re_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/_H
OST@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.

Securing an endpoint under AutoTLS

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 not 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 is you are using a self-signed certificate.

Token-based authentication for Cloudera Data Warehouse integrations

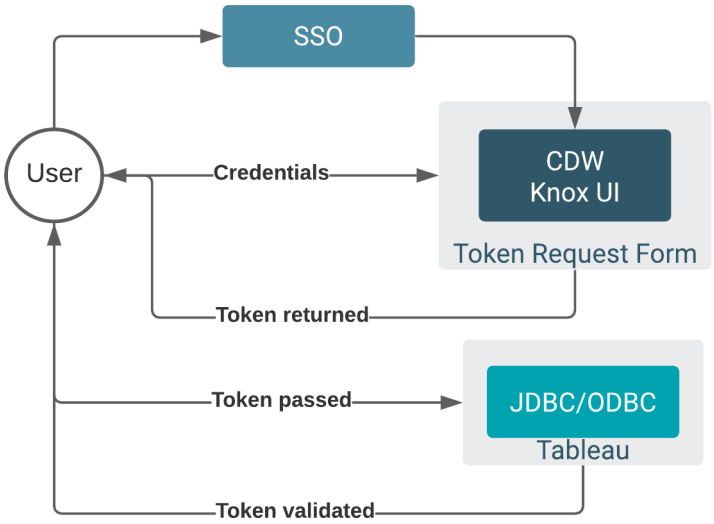
Using a token, you can sign on to use Hive and Impala in Cloudera Data Warehouse for a period of time instead of entering your single-sign on (SSO) credentials every time you need to run a query.

You use the credentials only when starting a session; no need to enter credentials again. To create a token from the CDW UI, you first sign-in using SSO, using the multi-factor authentication setup by corporate IT administrators.

You obtain a token to use for the initial log in, set the lifespan of the access period, and submit the token for remote login to CDW from a JDBC client.

Token-based authentication for CDW integrations is in a Technical Preview state and requires an entitlement.

To create a token, you sign into CDP using SSO credentials, following the corporate multi-factor authentication process. A token generation UI appears in the open SSO session. You generate a token, with an expiration time. You use the token in JDBC/ODBC connection strings to connect to a Hive or Impala Virtual Warehouse from JDBC clients, such as Beeline or Tableau.



Signing on and running queries

You need to meet certain prerequisites before attempting to use token-based authentication for running Hive or Impala queries: local prerequisites for running queries from a node in the cluster, or remote prerequisites for running queries from outside the cluster.

Local prerequisites

- Knox is running on the cluster.

Remote prerequisites

- A JDBC/ODBC driver runs remotely on a server, for example the Tableau server.
- A thin client connects to the remote server and serves up a UI for entering a user name and password to connect.

Acquiring an integration token

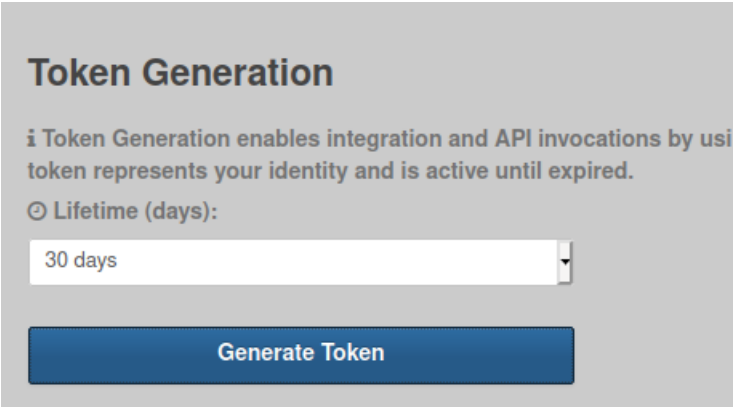
You need to obtain a JSON web token (JWT) and set the lifespan of that token. A token provides access only to the Virtual Warehouse where you obtained it.

About this task

You can access Hive and Impala only in the environment where you obtained the token.

Procedure

1. Sign into CDP using SSO credentials following the corporate multi-factor authentication process.
   Token Generation appears in the open SSO session.
2. In Token Generation, choose the lifespan of the token, and click Generate Token.



3. Copy the Passcode token to use in the next procedure.

   For security reasons, after you close Token Generation, you cannot see the token again.

Using the Passcode token

After copying the Passcode token, you need to connect from a client to a Hive or Impala Virtual Warehouse in CDW. You see how to construct the connection string that a client uses to connect to CDW using token-based authentication.

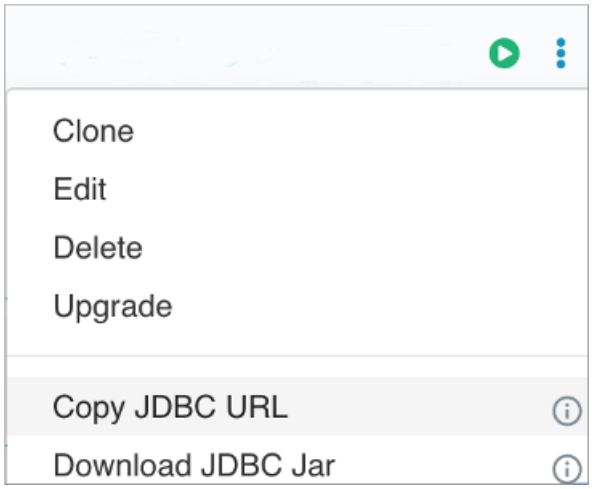
About this task

To create the Beeline connection command, you get the connection string for Hive or Impala to use with the Passcode token.

Procedure

1. On the client end, download the latest version of the Hive JDBC driver from Cloudera Downloads page.
2. Install the driver on the client end.

   Typically, you add the JAR file to the Libraries folder.
3. Log into the CDP web interface and navigate to the Data Warehouse service.
4. In the Data Warehouse service, click Virtual Warehouse in the left navigation panel.
5. In a Virtual Warehouse, click options , and select Copy JDBC URL.



6. Paste the copied JDBC URL into a text file.

```
jdbc:hive2://<your-virtual-warehouse>.<your-environment>.<dwx.company.com>/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3
```

7. Edit the JDBC URL to construct a Beeline command that connects to the Virtual Warehouse, using the word token (literally) as the user name and using the Passcode Token you copied earlier as the password.

```
beeline -u "jdbc:hive2://my-vw.my-env.dwx.mycompany.com/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3" -n token -p {Passcode Token}
```

   After pasting your Passcode Token at the end of the command, the command looks something like this:

```
beeline -u "jdbc:hive2://my-vw.my-env.dwx.mycompany.com/default;transportMode=http;httpPath=cliservice;ssl=true;retries=3" -n token -p 294a07f4-5ae5-40ed-b45c-d4076693ae11
```

8. Execute the beeline command, and then you can run queries on the Hive or Impala Virtual Warehouse.

ODBC sign-on walkthrough

To answer business intelligence questions, you need to see how to sign into CDW and query Hive or Impala.

About this task

In this walkthrough, you see step-by-step how to use token authentication to sign on to CDW over ODBC from a Tableau Server desktop. The token generated for this CDW cluster cannot be used for any other cluster.

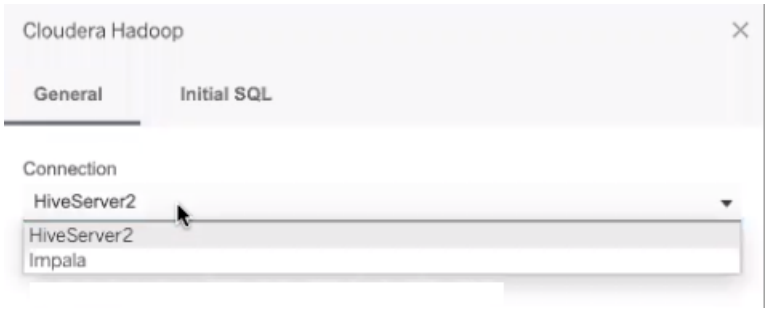
Before you begin

- You started a Virtual Warehouse.
- The CDW cluster is connected to Apache Knox.

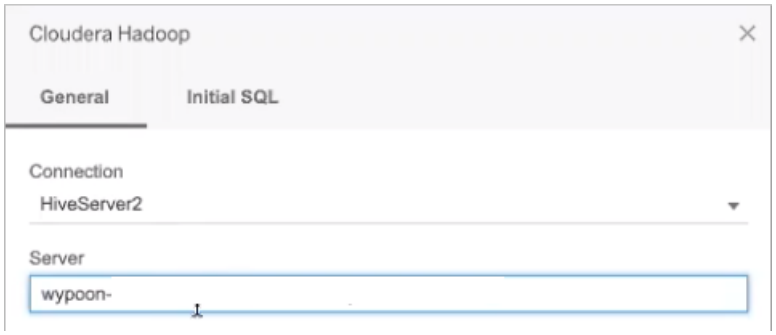
Setting up Tableau

Procedure

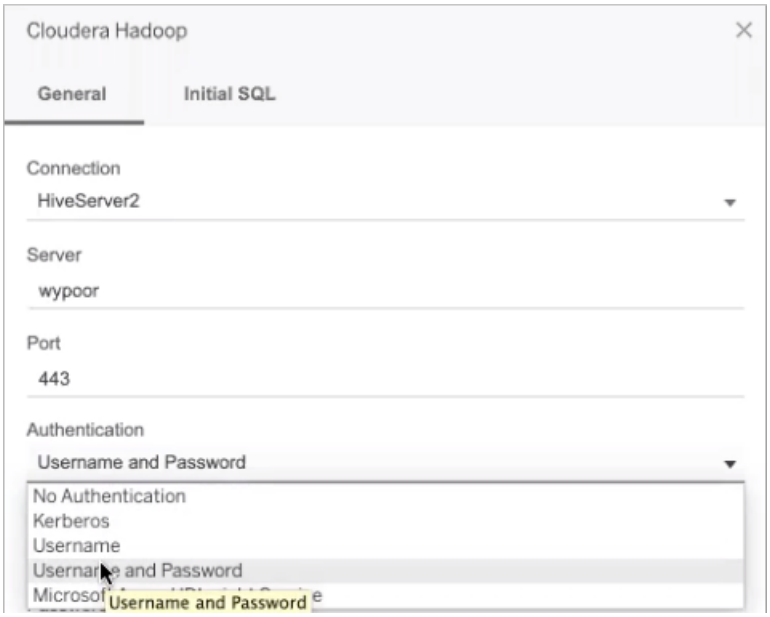
1. In your Tableau Server desktop, on the General table, from the Connection options (HiveServer2 and Impala), select HiveServer2.



2. In your Tableau Server desktop, in Server, paste the name of the HiveServer host from the JDBC URL you copied.



3. Select Port 443, and in Authentication, select the authentication type Username and Password.



4. In Username, type the word token literally, not your user name.
5. In Passcode, paste the passcode token from the token generation operation you performed earlier.
6. In Transport, select the HTTP transport mode.
7. Click Sign In.

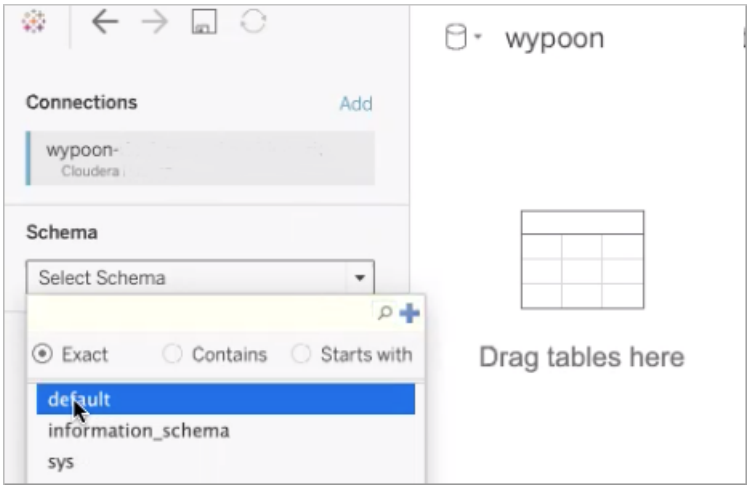
Querying a Hive table from Tableau

About this task

Signing into CDP from the Tableau Server desktop connects you to the Virtual Warehouse and you can load a table into Tableau.

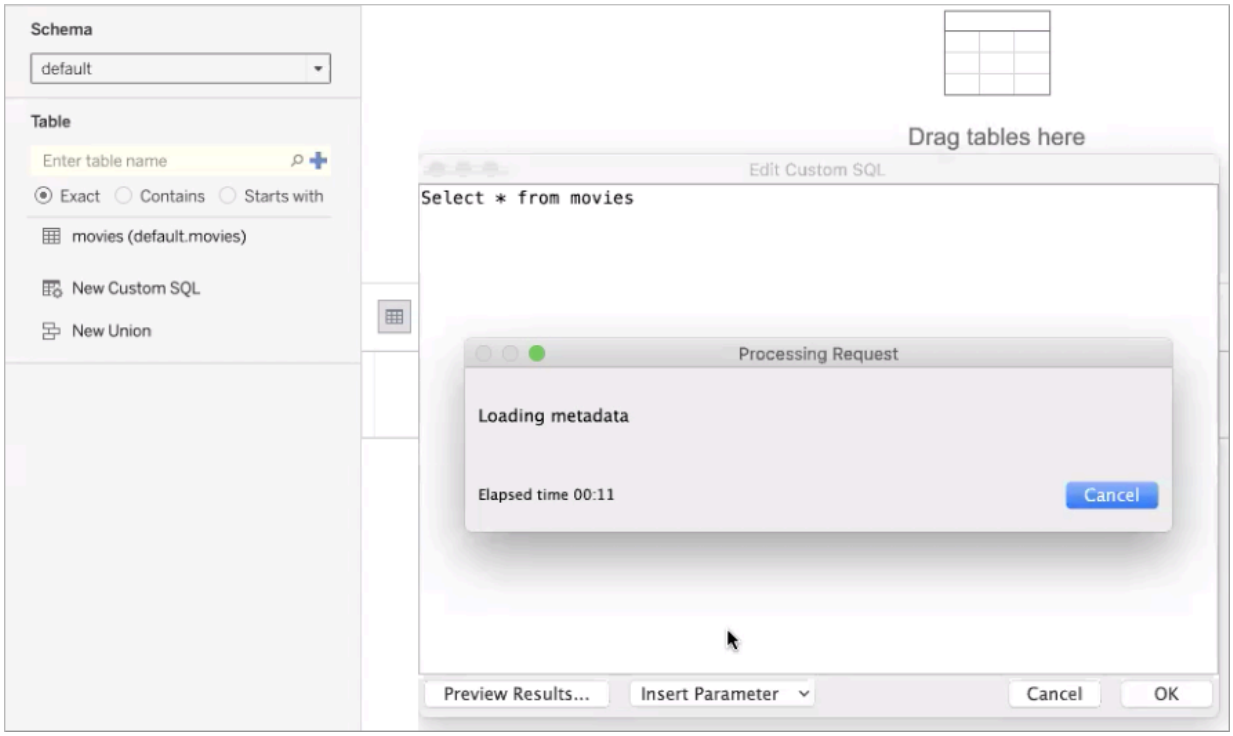
Procedure

1. Go back to the Tableau Server desktop, select the option to use the default schema of Hive in CDW.



2. Load a table called movies, for example.

3. Query the table. selecting all the movies, for example.



The token you generated to connect Tableau to the Virtual Warehouse cannot be used to connect Tableau to any other cluster or Virtual Warehouse.