Cloudera Runtime 7.1.0

Securing Apache Hive

Date published: 2019-08-21
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Authorizing Apache Hive Access

As administrator, you need to understand the insecure Hive default authorization for running Hive queries. You need to know your security options: to set up Ranger or Storage Based Authorization (SBA), which is based on impersonation and HDFS access control lists (ACLs), or a combination of these methods.

To limit Apache Hive access to approved users. Cloudera recommends Ranger. Authorization is the process that checks user permissions to perform select operations, such as creating, reading, and writing data, as well as editing table metadata. Apache Ranger provides centralized authorization for all Cloudera Runtime Services.

You can set up Ranger to protect managed, ACID tables or external tables using a Hadoop SQL policy. You can protect external table data on the file system by using an HDFS policy in Ranger.

You can set up SBA plus HDFS ACLs to protect external tables and external table data.

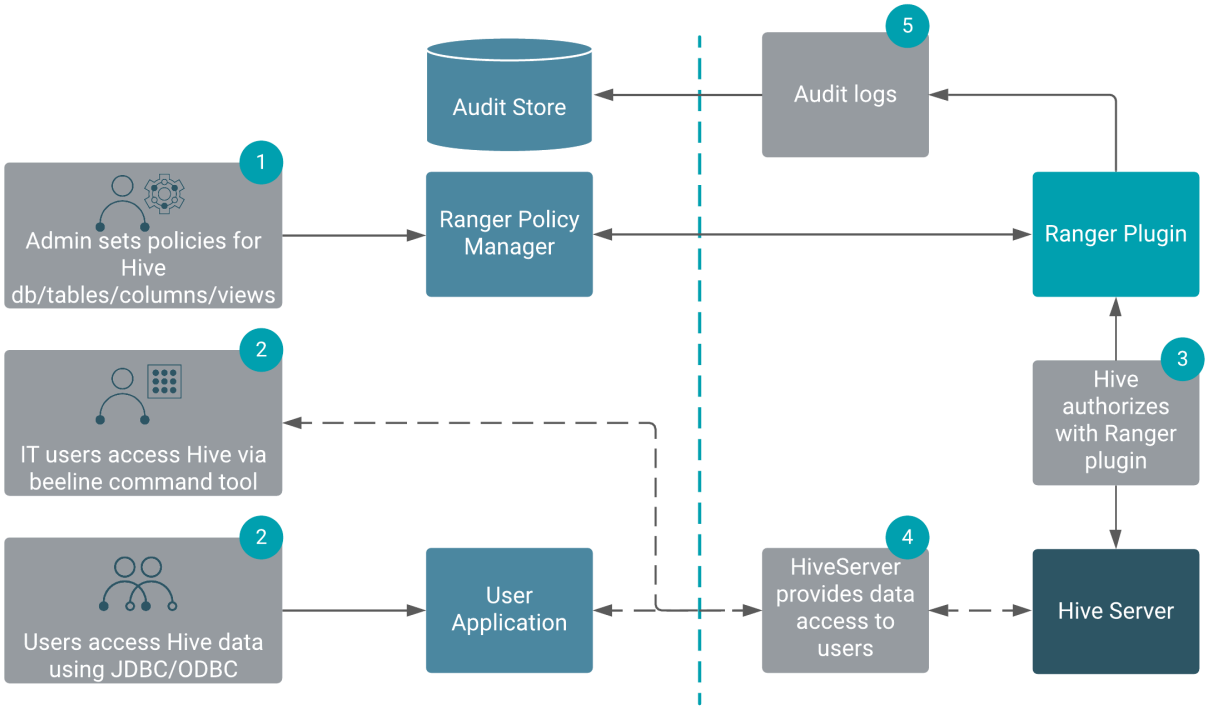
Preloaded Ranger Policies

In Ranger, preloaded Hive policies are available by default. Users covered by these policies can perform Hive operations. All users need to use the default database, perform basic operations such as listing database names, and query the information schema. To provide this access, preloaded default database tables columns and information_ schema database policies are enabled for group public (all users). Keeping these policies enabled for group public is recommended. For example, if the default database tables columns policy is disabled preventing use of the default database, the following error appears:

```
hive> USE default;
Error: Error while compiling statement: FAILED: HiveAccessControlException
Permission denied: user [hive] does not have [USE] privilege on [default]
```

Apache Ranger policy authorization

Apache Ranger provides centralized policy management for authorization and auditing of all Cloudera Runtime services, including Hive. All Cloudera Runtime services are installed with a Ranger plugin used to intercept authorization requests for that service, as shown in the following illustration.



Authorizing Hive through Ranger instead of using SBA is highly recommended.

Storage based authorization

Storage-based authorization is LDAP-based. As the name implies, storage-based authorization relies on the authorization provided by the storage layer. In CDP Private Cloud Base, the storage layer is HDFS, which provides both POSIX and ACL permissions. Hive is one of many Cloudera Runtime services that share storage on HDFS. By enabling this model on the Hive Metastore Server, Hadoop administrators can provide consistent data and metadata authorization. The model controls access to metadata and checks permissions on the corresponding directories of the HDFS file system. Traditional POSIX permissions for the HDFS directories where tables reside determine access to those tables. This authorization model doesn't support column-level security.

In addition to the traditional POSIX permissions model, HDFS also provides ACLs, or access control lists, as described in ACLs on HDFS. An ACL consists of a set of ACL entries, and each entry names a specific user or group and grants or denies read, write, and execute permissions for the specified user or group. These ACLs are also based on POSIX specifications, and they are compatible with the traditional POSIX permissions model.

HDFS ACL permissions provide administrators with authentication control over databases, tables, and table partitions on the HDFS file system. For example, an administrator can create a role with a set of grants on specific HDFS tables, then grant the role to a group of users. Roles allow administrators to easily reuse permission grants. Cloudera recommends relying on POSIX permissions and a small number of ACLs to augment the POSIX permissions for exceptions and edge cases.

A file with an ACL incurs additional memory cost to the NameNode due to the alternate algorithm used for permission checks on such files.

Authorization model comparison

In addition to Apache Ranger, Hive supports storage-based authorization (SBA) for external tables. Ranger and SBA can co-exist in CDP Private Cloud Base. The following table compares authorization models:

Authorization model	Secure?	Fine-grained authorization (column, row level)	Privilege management using GRANT/REVOKE statements	Centralized management GUI
Apache Ranger	Secure	Yes	Yes	Yes
Storage-based	Secure	No authorization at SQL layer in HiveServer. Provides Metastore server authorization for the Metastore API only.	No. Table privilege based on HDFS permission	No
Hive default	Not secure. No restriction on which users can run GRANT statements	Yes	Yes	No

When you run grant/revoke commands and Apache Ranger is enabled, a Ranger policy is created/removed.

Related Information

HDFS ACLS

Configure a Resource-based Policy: Hive

Row-level Filtering and Column Masking in Hive

Query Hive

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 on HDFS in CDP Private Cloud Base. Only the Hive service can own and interact with files in this directory. Storage-based authorization (SBA) does not work for giving users access to ACID tables.

Ranger is the only available authorization mechanism that Cloudera recommends for ACID tables.

External table access

As administrator, you must set up one of several authorization options to allow users to access external tables.

External tables reside by default in /warehouse/tablespace/external on HDFS (CDP Private Cloud Base) or S3 (CDP Public Cloud). 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 support the following permissions and authorization models:

- SBA
- SBA and Ranger
- Ranger

You can use the mixed mode, SBA and Ranger, for low-level analytical processing of external tables.

Using the SBA permissions model

You must add Access ACLs to allow groups or users to create databases and tables in the space governed by SBA. You are authorized to query a table if you have file-level access to the underlying data. You configure impersonation in HiveServer to run operations on behalf of an end user. You cannot use LLAP.

Using the SBA and Ranger example

Assume that you are an administrator who creates a sales database and gives the sales group read-write permissions to the sales directory. This includes Default ACLs for the sales group to read from and write to the database. Users in the sales group set doAs=true, and are authorized under SBA to create external tables. Given the ACLs, both Hive and sales users can access all files and partitions.

To restrict certain users from accessing all files and partitions, you can use Ranger. Hive enforces access; however, if you give a sales user fewer options for accessing the tables through SBA, for example by setting a user's HDFS access to tables to read-only, Ranger cannot control that user's access.

Using the Ranger authorization model

If you disable SBA and use only Ranger to give a specific user, who is not in the sales group, permission to create external tables in the sales-report database, the user can log in, and create a database. With Default ACLs in place, sales group users can also access the table.

Related Information

Enabling or disabling impersonation (doas)
HDFS ACLS

Enabling or disabling impersonation (doas)

As administrator, you must understand the permissions model supported in CDP Private Cloud Base and later. If you do not use Apache Ranger for security, you need to add users to an HDFS access control list (ACL) to permit access to the Hive warehouse for running DML queries on external tables. You base your doas configuration on the type of table you generally create.

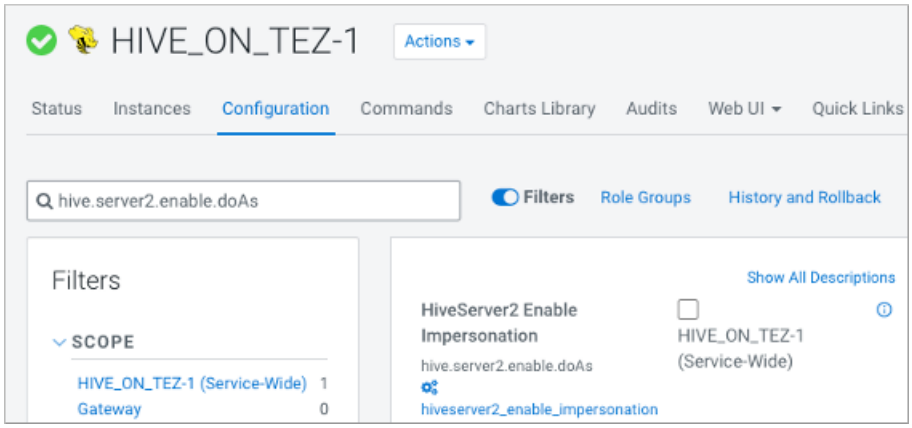
Hive 3 supports the HDFS access control model instead of the legacy Hive permission inheritance based on the hive.warehouse.subdir.inherit.perms parameter setting. In Apache Hive 3, a directory inherits permissions from the Default ACL.

Configure services for this behavior, as described below:

Disable impersonation to use Ranger

When you enable Ranger, you disable user impersonation (doAs=false). This is the Hive default and Ranger is the recommended security model. Managed, ACID tables as well as external tables, secured by Ranger, are supported in this configuration. Impersonation of the end user is disabled, which is the state required by Hive for managing ACID tables.

In Cloudera Manager, click Hive on Tez Configuration and search for (hive.server2.enable.doAs).



Uncheck Hive (Service-Wide) to disable impersonation.

With no impersonation, HiveServer authorizes only the hive user to access Hive tables.

Enable impersonation to use SBA

As administrator, if you do not use the recommended Ranger security, you simply enable the doAs impersonation parameter to use SBA: In Cloudera Manager, click Hive on Tez Configuration and for HiveServer2 Enable Impersonation, check Hive (Service-Wide) to enable impersonation (doAs=true).

External tables are supported in this configuration. ACID, managed tables are not supported. Impersonation of the end user is enabled.

Related Information

Apache Software Foundation HDFS Permissions Guide

HDFS ACLS

Managing YARN queue users

To manage users of secure YARN queues, you need to know how to configure impersonation for the security model you select.

To allow access to YARN queues, as Administrator, you configure HiveServer user impersonation, and another property or not, depending on your security: Ranger or security-based authorization (SBA). In either case, to manage YARN queues, you need the following behavior:

- User submits the query through HiveServer (HS2) to the YARN queue
- Tez app starts for the user
- Access to the YARN queue is checked for this user.

  As administrator, you can allocate resources to different users.

Managing YARN queues under Ranger

When you use Ranger, you configure HiveServer not to use impersonation (doas=false). HiveServer authorizes only the hive user, not the connected end user, to access Hive tables and YARN queues unless you also configure the following parameter:

hive.server2.tez.queue.access.check=true

Managing YARN queues under SBA

As administrator, if you do not use the recommended Ranger security, you simply enable the doAs impersonation (doas=true) parameter to use SBA. This action also causes HiveServer to authorize the connected user who issued the query to access YARN queues while running the Tez application as the hive user.

Configure storage-based authorization

You need to set parameters to enable storage-based authorization (SBA).

About this task

Hive performs authorization checks on the client, rather than the server when you use SBA. This allows malicious users to circumvent these checks. Some metadata operations do not check for authorization. See Apache JIRA HIVE-3009. DDL statements for managing permissions have no effect on storage-based authorization, but they do not return error messages (HIVE-3010).

Before you begin

- You obtained admin role privileges.

Procedure

1. Set authorization configuration parameters to enable storage-based authorization using the Cloudera Manager Safety Valve feature (see link below).

```
<property>
   <name>hive.security.authorization.enabled</name>
   <value>false</value>
</property>

<property>
   <name>hive.security.authorization.manager</name>
   <value>org.apache.hadoop.hive.ql.security.authorization.\
StorageBasedAuthorizationProvider</value>
</property>

<property>
   <name>hive.server2.enable.doAs</name>
   <value>true</value>
</property>

<property>
   <name>hive.metastore.pre.event.listeners</name>
   <name>org.apache.hadoop.hive.ql.security.authorization.\
AuthorizationPreEventListener</name>
</property>

<property>
   <name>hive.security.metastore.authorization.manager</name>
   <value>org.apache.hadoop.hive.ql.security.authorization.\
StorageBasedAuthorizationProvider</value>
</property>
```

2. Determine the required permissions of the tables and databases in your environment.
3. Create a table or database in the Hive, then manually modify the POSIX permissions using the HDFS file system commands.

Related Information

Storage-based authorization information on the Apache Wiki

Storage-based operation permissions

If you use SBA, you need to know which Hive operations have read and write access to your Hive databases and tables.

Table 1: Required Minimum Permissions for Hive Operations

Operation	Database READ Access	Database WRITE Access	Table READ Access	Table WRITE Access
LOAD				X
EXPORT			X	
IMPORT				X
CREATE TABLE		X		
CREATE TABLE AS SELECT		X	X (source table)	
DROP TABLE		X		
SELECT			X	
ALTER TABLE				X
SHOW TABLES	X			

Configure HiveServer for ETL using YARN queues

You need to set several configuration properties to allow placement of the Hive workload on the Yarn queue manager, which is common for running an ETL job. You need to set several parameters that effectively disable the reuse of containers. Each new query gets new containers routed to the appropriate queue.

About this task

Hive configuration properties affect mapping users and groups to YARN queues. You set these properties to use with YARN Placement Rules.

To set Hive properties for YARN queues:

Procedure

1. In Cloudera Manager, click Clusters Hive Configuration .
2. Search for the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml setting.
3. In the Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml setting, click +.
4. In Name enter the property hive.server2.tez.initialize.default.sessions and in value enter false.
5. In Name enter the property hive.server2.tez.queue.access.check and in value enter true.
6. In Name enter the property hive.server2.tez.sessions.custom.queue.allowed and in value enter true.

Connecting to an Apache Hive endpoint through Apache Knox

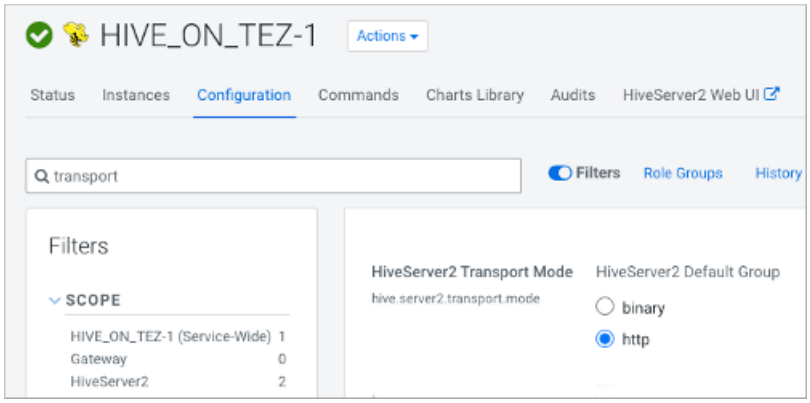
If your cluster uses Apache Knox for perimeter security in CDP Private Cloud Base, you can connect to an Apache Hive endpoint through Knox. You set the HiveServer transport mode and reference your Java keystore.

Before you begin

Automate the creation of an internal certificate authority (CA) using Auto-TLS (see link below). Set up SSL, including trust, for Knox Gateway clients.

Procedure

1. In Cloudera Manager, click Clusters Hive on Tez Configuration , and change the Hive on Tez service transport mode in Cloudera Manager to http.



   KNOX discovers the service automatically and builds a proxy URL for Hive on Tez only when the transport mode is http.

2. Download the Knox Gateway TLS/SSL client trust store JKS file from Knox, and save it locally.

   You can find the location of the JKS file from value of the Knox property gateway.tls.keystore.path.

3. In the Hive connection string, include parameters as follows:

```
jdbc:hive2://<host>:8443/;ssl=true;transportMode=http;  \
httpPath=gateway/cdp-proxy-api/hive;  \
sslTrustStore=/<path to JKS>/bin/certs/gateway-client-trust.jks;  \
trustStorePassword=<Java default password>
```

   In this example, changeit is the Java default password for the trust store.

Apache Spark access to Apache Hive

From Apache Spark, you access ACID tables and external tables in Apache Hive 3 using the Hive Warehouse Connector.

The HiveWarehouseConnector library is a Spark library built on top of Apache Arrow for reading and writing Hive ACID and external tables from Spark.

In CDP Public Cloud, the Hive Warehouse Connector is designed to leverage the LLAP cache and optimized for fast transmission of data using low-latency analytical processing (LLAP). The connector orchestrates a distributed read from LLAP daemons. The read from cache occurs after applying security rules and ACID transformations. CDP Public Cloud uses LLAP to read ACID, or other Hive-managed tables, from Spark. You do not need LLAP to write to

ACID, or other managed tables, from Spark. You do not need LLAP to access external tables from Spark. The HWC library internally uses the Hive Streaming API and LOAD DATA Hive commands to write the data.

In CDP Private Cloud Base, the Hive Warehouse Connector uses JDBC to transmit data.

Related Information

Hive Warehouse Connector for accessing Apache Spark data

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"
```

In CDP Private Cloud Base, under certain circumstances, you can query remote clusters that use a different version of Hive than the version installed on your cluster. You can query the data on the remote cluster and include WRITE operations from the local cluster.

Examples of Supported Queries

```
CREATE TABLE orders_ctas AS SELECT * FROM orders_ext;
```

```
INSERT INTO orders_ctas SELECT * FROM orders_ext;
```

```
INSERT OVERWRITE TABLE orders_ctas SELECT * FROM orders_ext;
```

Related Information

Encrypting Communication

Enable TLS/SSL for HiveServer

Enable SASL in HiveServer

Secure HiveServer using LDAP

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

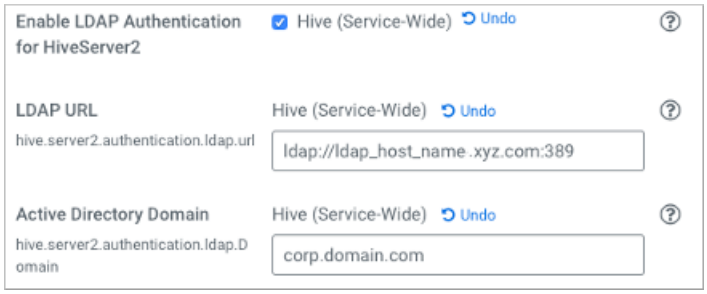
About this task

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.

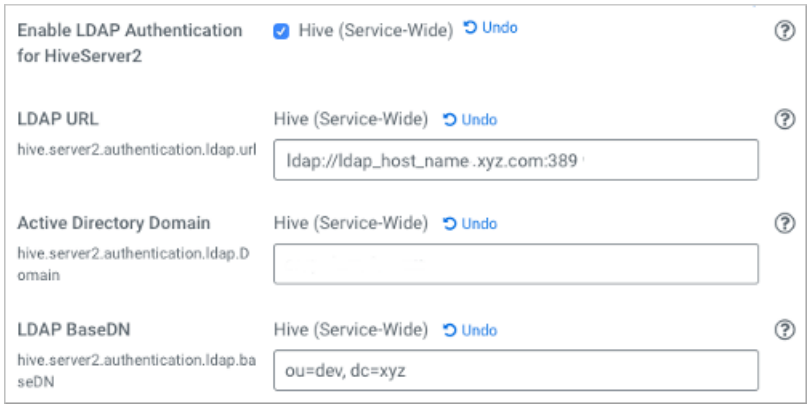
Procedure

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.



   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)
Encrypting Communication

Enable TLS/SSL for HiveServer
Enable SASL in HiveServer

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

CDP Private Cloud Base supports a number of modes for interacting with Hive, including Ranger-based authorization.

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.

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.

Related Information

Encrypting Communication
Enable TLS/SSL for HiveServer
Enable SASL in HiveServer

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.

Encrypting Communication

Encryption between HiveServer2 and its clients is independent from Kerberos authentication. HiveServer supports the following types of encryption between the service and its clients (Beeline, JDBC/ODBC):

- SASL (Simple Authentication and Security Layer)
- TLS/SSL (Transport Layer Security/Secure Sockets Layer)

TLS/SSL requires certificates. SASL QOP encryption does not. SASL QOP is aimed at protecting core Hadoop RPC communications. SASL QOP might cause performance problems when handling large amounts of data. You can configure HiveServer to support TLS/SSL connections from JDBC/ODBC clients using Cloudera Manager.

Client connections to HiveServer2 over TLS/SSL

A client connecting to a HiveServer2 over TLS/SSL must access the trust store on HiveServer to establish a chain of trust and verify server certificate authenticity. The trust store is typically not password protected. The trust store might be password protected to prevent its contents from being modified. However, password protected trust stores can be read from without using the password.

The client needs the path to the trust store when attempting to connect to HiveServer2 using TLS/SSL. You can specify the trust store in one of the following ways:

- Pass the path to the trust store each time you connect to HiveServer in the JDBC connection string:

```
jdbc:hive2://fqdn.example.com:10000/default;ssl=true;\
 sslTrustStore=$JAVA_HOME/jre/lib/security/jssecacerts;trustStorePassword
=extraneous
```

- Set the path to the trust store one time in the Java system javax.net.ssl.trustStore property:

```
java -Djavax.net.ssl.trustStore=/usr/java/jdk1.8.0_141-cloudera/jre/lib/
security/jssecacerts \
 -Djavax.net.ssl.trustStorePassword=extraneous MyClass \
 jdbc:hive2://fqdn.example.com:10000/default;ssl=true
```

Enable TLS/SSL for HiveServer

You can secure client-server communications using symmetric-key encryption in the TLS/SSL (Transport Layer Security/Secure Sockets Layer) protocol. To encrypt data exchanged between HiveServer and its clients, you can use Cloudera Manager to configure TLS/SSL.

Before you begin

- HiveServer has the necessary server key, certificate, keystore, and trust store set up on the host system.

- The hostname variable ($(hostname -f)-server.jks) was used with Java keytool commands to create keystore, as shown in this example:

```
$ sudo keytool -genkeypair -alias $(hostname -f)-server -keyalg RSA -key
store \
    /opt/cloudera/security/pki/$(hostname -f)-server.jks -keysize 2048 -
dname \
    "CN=$(hostname -f),OU=dept-name-optional,O=company-
name,L=city,ST=state,C=two-digit-nation" \
    -storepass password -keypass password
```

About this task

On the beeline command line, the JDBC URL requirements include specifying ssl=true;sslTrustStore=<path_to_truststore>. Truststore password requirements depend on the version of Java running in the cluster:

- Java 11: the truststore format has changed to PKCS and the truststore password is required; otherwise, the connection fails.
- Java 8: The trust store password does not need to be specified.

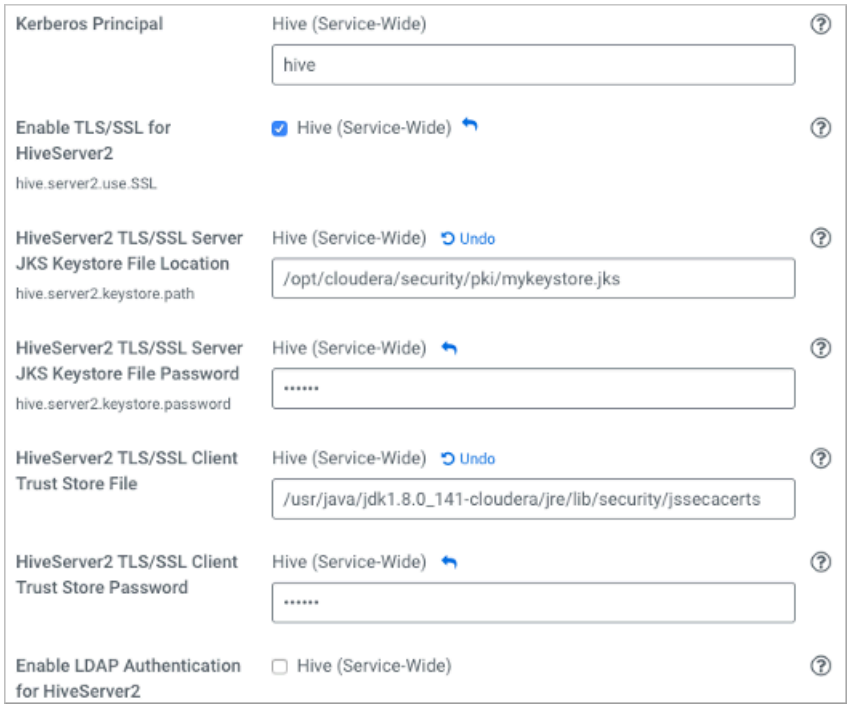
Procedure

1. In Cloudera Manager, navigate to Clusters Hive Configuration .
2. In Filters, select HIVE for the scope.
3. Select Security for the category.
4. Accept the default Enable TLS/SSL for HiveServer2, which is checked for Hive (Service-Wide).
5. Enter the path to the Java keystore on the host system.
   /opt/cloudera/security/pki/*keystore_name*.jks
6. Enter the password for the keystore you used on the Java keytool command-line when the key and keystore were created.

   The password for the keystore must match the password for the key.

7. Enter the path to the Java trust store on the host system.

   Cloudera clusters are typically configured to use the alternative trust store, jssecacerts, set up at $JAVA_HOME/jre/lib/security/jssecacerts.



8. Click Save Changes.
9. Restart the Hive service.
10. Construct a connection string for encrypting communications using TLS/SSL.

```
jdbc:hive2://#<host>:#<port>/#<dbName>;ssl=true;sslTrustStore=#<ssl_trus
tstore_path>; \
trustStorePassword=#<truststore_password>;#<otherSessionConfs>?#<hiveCon
fs>#<hiveVars>
```

Enable SASL in HiveServer

You can provide a Quality of Protection (QOP) that is higher than the cluster-wide default using SASL (Simple Authentication and Security Layer).

About this task

HiveServer2 by default uses hadoop.rpc.protection for its QOP value. Setting hadoop.rpc.protection to a higher level than HiveServer (HS2) does not usually make sense. HiveServer ignores hadoop.rpc.protection in favor of hive.server2.thrift.sasl.qop.

You can determine the value of hadoop.rpc.protection: In Cloudera Manager, click Clusters HDFS Configuration Hadoop , and search for hadoop.rpc.protection.

If you want to provide a higher QOP than the default, set one of the SASL Quality of Protection (QOP) levels as shown in the following table:

auth	Default. Authentication only.
auth-int	Authentication with integrity protection. Signed message digests (checksums) verify the integrity of messages sent between client and server.

auth-conf	Authentication with confidentiality (transport-layer encryption) and integrity. Applicable only if HiveServer is configured to use Kerberos authentication.

Procedure

1. In Cloudera Manager, navigate to Clusters Hive Configuration .
2. In HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site click + to add a property and value.
3. Specify the QOP auth-conf setting for the SASL QOP property.
   For example,

   Name:hive.server2.thrift.sasl.qop

   Value: auth-conf
4. Click Save Changes.
5. Restart the Hive service.
6. Construct a connection string for encrypting communications using SASL.

   ```
   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.

Secure Hive Metastore

Cloudera recommends using Apache Ranger policies to secure Hive data in Hive MetaStore. You need to perform a few actions to prevent users from bypassing HiveServer to access the Hive metastore and the Hive metastore database.

Procedure

1. Add a firewall rule on the metastore service host to allow access to the metastore port only from the HiveServer2 host. You can do this using iptables.
2. Grant access to the metastore database only from the metastore service host.
   For example, in MySQL: GRANT ALL PRIVILEGES ON metastore.* TO 'hive'@'metastorehost'; where metastorehost is the host where the metastore service is running.
3. Make sure users who are not administrators cannot log into the HiveServer host.