Cloudera Runtime 7.3.3

Working with Apache Hive metastore

Date published: 2019-08-21
Date modified: 2026-11-13



https://docs.cloudera.com/

Legal Notice

© Cloudera Inc. 2026. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

HMS table storage....4
Hive metastore system database architecture....5
Querying table comments and parameters....5
Hive system database tables....6
Configuring HMS for high availability....7
Hive Metastore leader election....8
Enabling Hive Metastore leader election....9
Hive Metastore leader election properties....10

HMS table storage

You need to understand how the Hive metastore (HMS) stores Hive tables when you run a CREATE TABLE statement or migrate a table to Cloudera. The success or failure of the statement, the resulting table type, and the table location depends on a number of factors.

HMS table transformations

The HMS includes the following Hive metadata about tables that you create:

- A table definition
- Column names
- Data types
- Comments in a central schema repository

When you use the EXTERNAL keyword in the CREATE TABLE statement, HMS stores the table as an external table. When you omit the EXTERNAL keyword and create a managed table, or ingest a managed table, HMS might translate the table into an external table or the table creation can fail, depending on the table properties. An important table property that affects table transformations is the ACID or Non-ACID table type:

Non-ACID

Table properties do not contain any ACID related properties set to true. For example, the table does not contain such properties transactional=true or insert_only=true.

ACID

Table properties do contain one or more ACID properties set to true.

Full ACID

Table properties contain transactional=true but not insert_only=true

Insert-only ACID

Table properties contain insert_only=true.

The following matrix shows the table type and whether or not the location property is supported.

ACID	Managed	Location Property	Comments	Action
Non-ACID	Yes	Yes	Migrated to Cloudera, for example from an HDP or CDH cluster.	Table stored as external
Non-ACID	Yes	No	Table location is null	Table stored in subdirectory of external warehouse*

* metastore.warehouse.external.dir

HMS detects type of client for interacting with HMS, for example Hive or Spark, and compares the capabilities of the client with the table requirement. HMS performs the following actions, depending on the results of the comparison:

Table requirement	Client meets requirements	Managed Table	ACID table type	Action
Client can write to any type of ACID table	No	Yes	Yes	CREATE TABLE fails
Client can write to full ACID table	No	Yes	insert_only=true	CREATE TABLE fails
Client can write to insert-only ACID table	No	Yes	insert_only=true	CREATE TABLE fails

If, for example, a Spark client does not have the capabilities required, the following type of error message appears:

```
Spark has no access to table `mytable`. Clients can access this table only i
f
they have the following capabilities: CONNECTORREAD,HIVEFULLACIDREAD, HIVE
FULLACIDWRITE,
HIVEMANAGESTATS, HIVECACHEINVALIDATE, . . .
```

Hive metastore system database architecture

Hive Metastore system database provides read-only schemas to inspect catalog metadata securely.

Hive Metastore (HMS) persists metadata such as databases, tables, partitions, and storage descriptors in a backend relational database management system (RDBMS) such as MySQL, PostgreSQL, or Oracle. Although you can run SQL queries directly against that backend database, Cloudera recommends that you avoid direct access.

Direct querying is excluded from standard practices due to the following reasons:

- The HMS backend schema can change across Hive versions and deployments.
- Direct querying bypasses HMS-level abstractions and access controls.
- Certain database backends require vendor-specific SQL and tooling.

Instead, you can run read-only queries through Hive by using the beeline client against the metastore system database, which is typically exposed as the sys database. The sys database contains more than 50 tables and mirrors most Hive Metastore (HMS) backend catalog tables, and the metastore schema upgrade scripts create and refresh the sys database.



Important: You must treat the sys database as a read-only introspection surface. Column availability can vary by Hive version or configuration.

Querying table comments and parameters

Query the table parameters property inside the system database to extract catalog descriptions.

About this task

When you need to find the specific comment or configuration parameters of a table, you can retrieve this metadata from the TABLE_PARAMS table inside the sys database instead of accessing the backend database directly.

Before you begin

Ensure that you have access permissions to run the beeline client.

Procedure

1. Open your terminal interface and run the beeline client.
2. Select the system database by running the following command:

   ```
   USE sys;
   ```

3. Extract the table comments by running the following metadata query:

   ```
   SELECT
     d.name AS db_name,
     t.tbl_name AS tbl_name,
     max(CASE WHEN p.param_key = 'comment' THEN p.param_value END) AS
    table_comment
   ```

```
FROM DBS d
JOIN TBLS t
  ON d.db_id = t.db_id
LEFT JOIN TABLE_PARAMS p
  ON p.tbl_id = t.tbl_id
WHERE t.tbl_type <> 'VIRTUAL_VIEW'
  AND lower(d.name) NOT IN ('sys', 'information_schema')
GROUP BY d.name, t.tbl_name
ORDER BY db_name, tbl_name;
```

Hive system database tables

Sys tables list core metadata catalog definitions and schema properties.

SYS db tables

The following list shows the functional groups of tables in the sys database:

- Catalog — DBS, TBLS, PARTITIONS, COLUMNS_V2, TABLE_PARAMS, DATABASE_PARAMS, PARTITION_PARAMS
- Storage — SDS, SERDES, SERDE_PARAMS, SD_PARAMS
- Authorization — TBL_PRIVS, DB_PRIVS, PART_COL_PRIVS
- Transaction — COMPACTION_QUEUE, TXNS, NOTIFICATION_LOG, REPLICATION_METRICS

Metadata query templates

The following query templates demonstrate how to inspect Hive Metastore metadata safely through the sys database:

Retrieve table information

```
USE sys;
SELECT
  d.name AS db_name,
  t.tbl_name
FROM DBS d
JOIN TBLS t
  ON d.db_id = t.db_id
WHERE t.tbl_type <> 'VIRTUAL_VIEW';
```

Fetch table location

```
USE sys;
SELECT
  d.name AS db_name,
  t.tbl_name,
  s.location
FROM DBS d
JOIN TBLS t
  ON d.db_id = t.db_id
JOIN SDS s
  ON t.sd_id = s.sd_id
WHERE t.tbl_type <> 'VIRTUAL_VIEW';
```

Fetch partition location

```
USE sys;
SELECT
  d.name AS db_name,
  t.tbl_name,
  p.part_name,
  s.location
```

```
FROM DBS d
JOIN TBLS t
  ON d.db_id = t.db_id
JOIN PARTITIONS p
  ON p.tbl_id = t.tbl_id
JOIN SDS s
  ON p.sd_id = s.sd_id
WHERE t.tbl_type <> 'VIRTUAL_VIEW';
```

Configuring HMS for high availability

To provide failover to a secondary Hive metastore if your primary instance goes down, you need to know how to add a Metastore role in Cloudera Manager and configure a property.

About this task

Multiple HMS instances run in active/active mode. The HMS client, such as HiveServer or Spark, selects an instance from the list in the hive.metastore.uris property according to the hive.metastore.uri.selection property. The default value is RANDOM: the client picks an HMS instance at random, and if that instance is down, it picks another instance from the list at random. If you set hive.metastore.uri.selection to SEQUENTIAL, the client picks the first valid instance in the list and uses the next instance as the designated replacement.

Before you begin

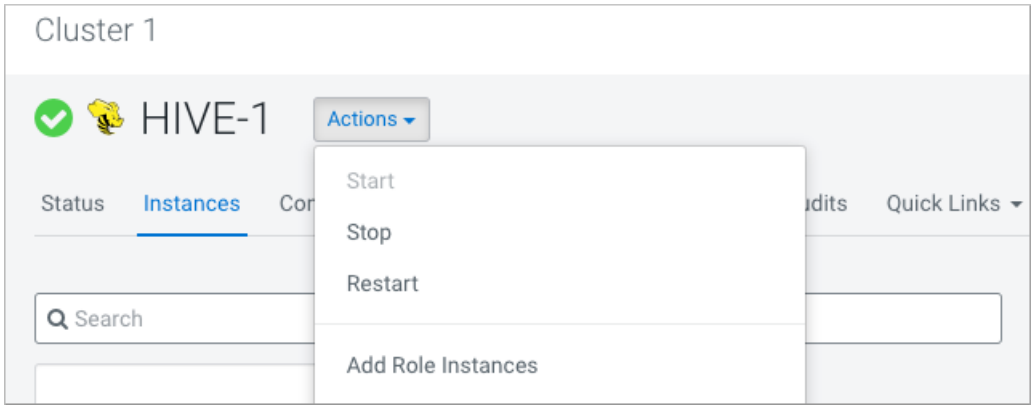
Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

Procedure

1. In Cloudera Manager, click Clusters Hive Configuration .
2. Take one of the following actions:
   - If you have a cluster secured by Kerberos, search for Hive Delegation Token Store, which specifies storage for the Kerberos token as described below.
   - If you have an unsecured cluster, skip the next step.
3. Select org.apache.hadoop.hive.thrift.DBTokenStore, and save the change.



   Storage for the Kerberos delegation token is defined by the hive.cluster.delegation.token.store.class property. The available choices are Zookeeper, the Metastore, and memory. Cloudera recommends using the database by setting the org.apache.hadoop.hive.thrift.DBTokenStore property.
4. Click Instances Actions Add Role Instances



5. In Assign Roles, in Metastore Server, click Select Hosts.
6. In Hosts Selected, scroll and select the host that you want to serve as the backup Metastore, and click OK.
7. Click Continue until you exit the wizard.
8. Start the Metastore role on the host from the Actions menu.

   The hive.metastore.uris property is updated automatically. To verify, go to /etc/hive/config directory in your cluster node and look for the updated property in the hive-site.xml file.
9. To check or change the hive.metastore.uri.selection property, go to Clusters Hive on Tez Configuration in Cloudera Manager, and search for 'Hive Service Advanced Configuration Snippet (Safety Valve) for hive-site.xml'.

   The default value of the hive.metastore.uri.selection property is RANDOM.
10. Add the hive.metastore.uri.selection property and set its value to SEQUENTIAL or RANDOM.

Note: For other non-HS2 clients, identify the hive-site.xml file used by the client and set the configuration to match the settings described above. If the hive.metastore.uri.selection property is not defined in the Hive Metastore (HMS) or HiveServer2 (HS2), the default behavior is RANDOM.

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

Hive Metastore leader election

Leader election lets Hive Metastore (HMS) elect a single instance to run housekeeping tasks, which prevents multiple instances from duplicating the same work and overloading the metastore database.

HMS runs housekeeping tasks such as generating and running compaction tasks, discovering partitions for external tables, cleaning up expired data. When a cluster runs multiple HMS instances, these tasks must run on a single instance. Otherwise, every instance runs the same tasks and duplicates the work.

Duplicated housekeeping work generates a large number of read or write operations against the shared metastore database. In environments that run many concurrent metadata operations, such as replication jobs that use multiple threads to fetch metadata, this additional load can degrade performance for all workloads that share the database.

Tasks that run only on the leader

When leader election is configured, HMS runs the following housekeeping tasks only on the elected leader:

- Compaction tasks, including the initiator, cleaner, and table status updater
- Automatic partition discovery and partition management
- Cleanup of the changed-message (CM) directory
- Transaction and event cleanup tasks that run only in remote server mode

Static configuration compared to dynamic election

HMS supports the following ways to elect the leader:

- **Static leader configuration**

  Assigns a single HMS instance to run housekeeping tasks using the metastore.housekeeping.leader.hostname property. This approach does not support automated failover if the leader stops responding, can be difficult to configure correctly in hybrid environments, and concentrates all tasks on one instance.
- **Dynamic leader election**

  Uses an exclusive Hive lock to elect a leader automatically across active HMS instances. The instance that acquires the lock becomes the leader and runs the housekeeping tasks.

How dynamic leader election works

Dynamic leader election uses a Hive lock as a lease. The HMS instance that acquires the exclusive lock becomes the leader and sends heartbeats to renew the lock before it times out. Each non-leader instance checks the lock periodically and asks HMS to clean up timed-out locks, so that a new leader can be elected if the current leader restarts, is upgraded, or stops responding.

When leadership changes, HMS notifies registered listeners through takeLeadership and loseLeadership events. The new leader starts the housekeeping tasks, and the previous leader stops them, which ensures that the tasks run on only one instance at a time.

HMS records each leader election event in a Hive audit table. You can run a query against this table to identify which instance was the leader at a given time.

Upgrade considerations

Enable dynamic leader election based on the following HMS deployment scenarios:

- **No leader hostname is configured**

  Enable the feature and housekeeping tasks on all instances. Every HMS instance acts as a leader in this setup.
- **A leader hostname is configured and all instances are in a single environment**

  Enable the feature on all instances. During the upgrade, the cluster can temporarily have up to two leaders, but only one leader remains after the upgrade completes.
- **Instances are distributed across multiple environments**

  Enable the feature and housekeeping tasks on all instances in each environment. Cloudera recommends upgrading the environment that contains the leader first. If you upgrade another environment first, the cluster can temporarily have more than one leader, which still functions but increases load on the metastore database.



Tip: To improve the performance of metadata operations, Cloudera recommends setting hive.metastore.try.direct.sql to true on all HMS instances.

Enabling Hive Metastore leader election

Enable dynamic leader election so that Hive Metastore (HMS) uses a Hive lock to elect a leader for housekeeping tasks instead of a statically configured host.

About this task

By default the HMS uses the dynamic election in Cloudera Runtime 7.3.2.10000. To elect the leader dynamically, set metastore.housekeeping.leader.election to lock and enable the supporting properties on all HMS instances so that a new leader can start the housekeeping tasks.

Before you begin

You must have access to the Cloudera Manager Hive Metastore configuration.

Procedure

1. Set the following property in the Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml advanced configurations snippet:

```
metastore.housekeeping.leader.election=lock
```

2. Set the following properties to true on all HMS instances to support high availability for leader election:

```
metastore.compactor.initiator.on=true
metastore.housekeeping.threads.on=true
metastore.compactor.cleaner.on=true
```

3. To track leadership changes, set the metastore.housekeeping.leader.auditTable property to the name of a Hive table.

   HMS creates the table automatically if it does not exist.

4. Save the changes and restart the Hive Metastore service.

Results

HMS instances compete for a Hive lock to elect the leader, and a new leader is elected automatically if the current leader stops responding.

Hive Metastore leader election properties

Properties that control Hive Metastore (HMS) dynamic leader election, including the election mode, the audit table, and lock retry behavior, configured in the Hive Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml advanced configurations snippet.

Table 1: HMS dynamic leader election properties

Property	Description	Default
metastore.housekeeping.leader.auditTable	Specifies a Hive table that tracks leadership changes. The table is in text format with JSON serialization and is created automatically if it does not exist. Cloudera recommends setting this property in a production environment so that you can monitor the leader change history by running a query against the table.	Empty string
metastore.housekeeping.leader.election	Specifies how HMS elects the leader. With the default value, HMS selects the leader based on the metastore.housekeeping.leader.hostname property value. Set the value to lock for HMS to use a Hive lock to elect the leader dynamically.	local
metastore.lock.numretries	Specifies the number of times HMS tries to acquire all the locks if the attempt fails.	100
metastore.lock.sleep.between.retries	Specifies the maximum sleep time between lock retries if the attempt fails.	60s