Cloudera Runtime 7.2.2

Configuring Apache Hive

Date published: 2020-10-07
Date modified:

https://docs.cloudera.com/

Legal Notice

Contents

Configuring legacy CREATE TABLE behavior

After you upgrade to CDP and migrate old tables, you might want to briefly switch to Hive legacy behavior. Legacy behavior might solve compatibility problems with your scripts during data migration, for example, when running ETL.

About this task

By default, executing a CREATE TABLE statement creates a managed Apache Hive 3 table in the Hive metastore. You can change the default behavior to use the legacy CREATE TABLE behavior. When you configure legacy behavior, CREATE TABLE generates external tables. Legacy behavior is recommended only during upgrading due to the advantages of full ACID transactional tables over external tables.

Apache Hive full ACID (transactional) tables deliver better performance, security, and user experience than non-transactional tables. By default, executing a CREATE TABLE statement creates a managed Apache Hive 3 table in the Hive metastore. Hive 3 tables are ACID-compliant, transactional tables having the following full ACID capabilities on data in ORC format only:

- Insert
- Update
- Delete

Using ACID-compliant, transactional tables causes no performance or operational overload. Bucketing is not necessary.

If you are a Spark user, switching to legacy behavior is unnecessary. Calling ‘create table’ from SparkSQL, for example, creates an external table after upgrading to CDP as it did before the upgrade.

Configure legacy CREATE TABLE behavior

When you configure legacy behavior, CREATE TABLE creates an external table in your specified warehouse, which is /warehouse/tablespace/external/hive by default. To configure legacy behavior at the session level, you can pass a property to HiveServer (HS2) in the Beeline connection string when you launch Hive. Alternatively, you can pass the property on the Hive command line to switch to the old behavior. You can also configure legacy create table behavior at the site level by configuring properties in Cloudera Manager. When configured at the site level, legacy behavior persists from session to session.

Related Information

Change DROP behavior

Session-level configuration

About this task

Step 1 describes two ways of configuring legacy CREATE TABLE behavior. You can override the configured legacy behavior as described in step 2 to create a managed table.

Procedure

1. Choose one of the following ways to configure legacy CREATE TABLE behavior:
   - To configure legacy behavior in any JDBC client, include hiveCreateAsExternalLegacy=true in the connection string. For example, in Beeline, include the connection string to launch Hive:

     ```
     beeline -u jdbc:hive2://10.65.13.98:10000/default;hiveCreateAsExternalLe
     gacy=true \
     -n <your user name> -p
     ```

- To configure legacy behavior within an existing beeline session, set hive.create.as.external.legacy=true. For example:

```
hive> SET hive.create.as.external.legacy=true;
```

You can purge the table from the file system and metastore. You can change the DROP behavior, to remove metadata only.

2. Override the configured legacy behavior at the session level (only) to create a managed table by using the MANAGED keyword.

```
CREATE MANAGED TABLE test (id INT);
```

When your session ends, the create legacy behavior also ends. If you issue a CREATE TABLE statement, Hive creates either an insert-only or full ACID table, depending on how you set the following table properties:

- hive.create.as.insert.only
- hive.create.as.acid

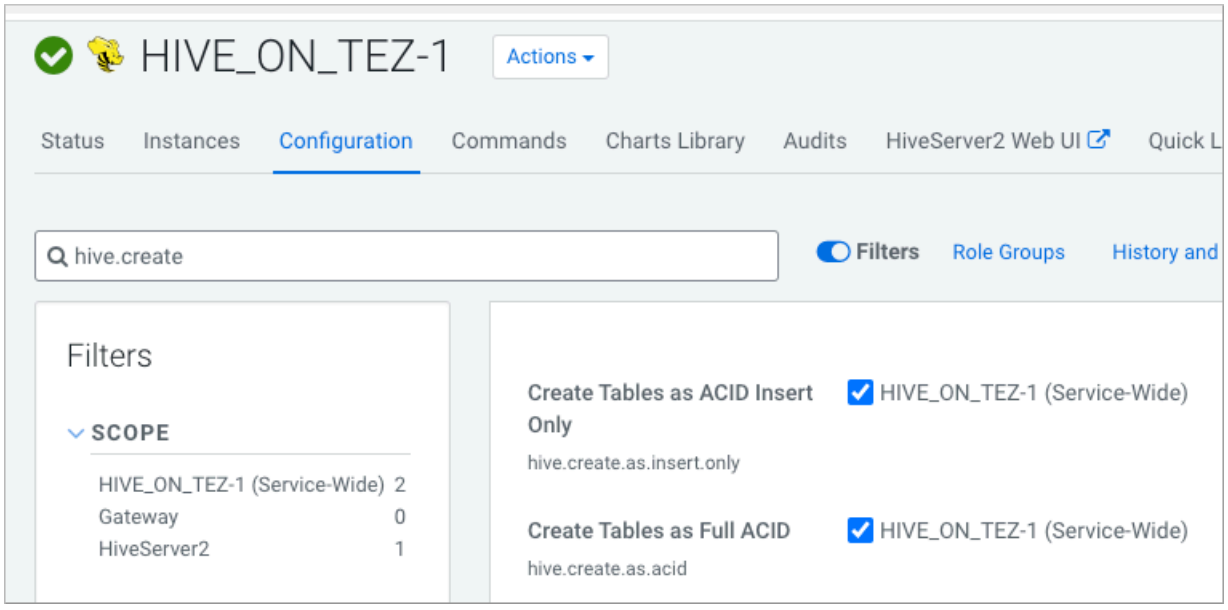
Site-level configuration

About this task

When you configure legacy create table behavior at the site level, the legacy behavior persists from session to session. You configure this behavior at the site level using Cloudera Manager as follows:

Procedure

1. In Cloudera Manager > Clusters > Hive On Tez, search for hive.create.



2. If Create Tables as ACID Insert Only and Create Tables as Full ACID properties appear and are set, uncheck the properties.

Limit concurrent connections

To prevent a rogue application from repeatedly connecting to and monopolizing HiveServer, you can limit concurrent connections to HiveServer.

About this task

As administrator, you can limit concurrent connections using the Cloudera Manager Safety Valve to add one or more of the following properties to the hive-site.xml configuration file:

hive.server2.limit.connections.per.user

Maximum number of HiveServer concurrent connections per user

hive.server2.limit.connections.per.ipaddress

Maximum number of HiveServer concurrent connections per IP address

hive.server2.limit.connections.per.user.ipaddress

Maximum number of HiveServer concurrent connections per user and IP address combination

The default of each parameter is 0. You can change the value of each parameter to any number. You must configure concurrent connections on the server side; therefore, a hive --hiveconf command does not work.

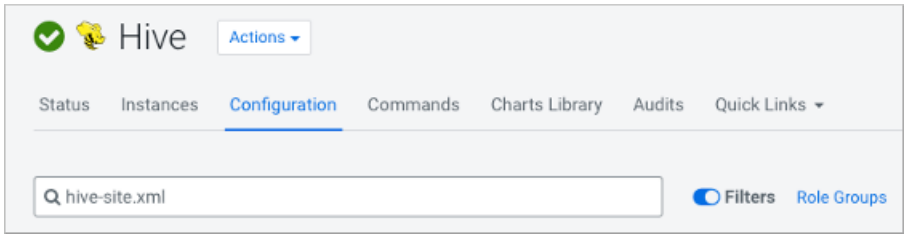
In this task, limit the number of connections per user to 25.

Before you begin

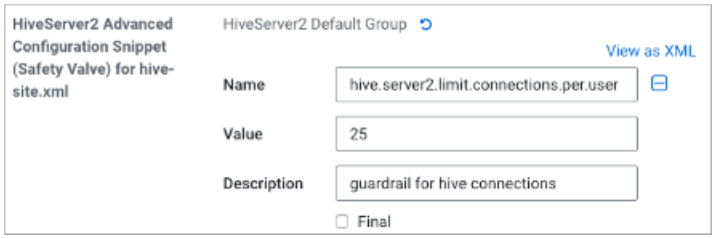
- The following components are running:
  - HiveServer
  - Hive Metastore
  - Hive client
- Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

Procedure

1. In Cloudera Manager Clusters select the Hive service. Click Configuration, and search for hive-site.xml.



2. In HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml, click + and add the hive.server2 .limit.connections.per.user property.
3. Enter a value representing the maximum number of concurrent connections: for example 25.



4. Click Save.

5. Click Actions Deploy Client Configuration .
6. Restart HIVE.

Configuring HiveServer high availability

You need to know how to configure your Hive-on-Tez to use ZooKeeper for HiveServer high availability.

When you add one or more additional HiveServer (HS2) role instances to the Hive-on-Tez service, the multiple role instances can register themselves with ZooKeeper. The JDBC client (client driver) can find a HiveServer through ZooKeeper. Using Beeline, you connect to Hive, and the ZooKeeper discovery mechnism locates and connects to one of the running HiveServer instances.

If more than one HiveServer instance is registered with ZooKeeper, and all instances fail except one, ZooKeeper passes the link to the instance that is running and the client can connect successfully. Failed instances must be restarted manually.

Automatic failover does not occur. If an HS2 instance failed while a client is connected, the session is lost. Since this situation needs to be handed at the client, there is no automatic failover; the client needs to reconnect using ZooKeeper.

Generating statistics

A cost-based optimizer (CBO) generates efficient query plans. Hive does not use the CBO until you generate column statistics for tables. By default, Hive gathers only table statistics. You need to configure Hive to enable gathering of column statistics.

The CBO, powered by Apache Calcite, is a core component in the Hive query processing engine. The CBO optimizes plans for executing a query, calculates the cost, and selects the least expensive plan to use. In addition to increasing the efficiency of execution plans, the CBO conserves resources.

How the CBO works

After parsing a query, a process converts the query to a logical tree (Abstract Syntax Tree) that represents the operations to perform, such as reading a table or performing a JOIN. Calcite applies optimizations, such as query rewrite, JOIN re-ordering, JOIN elimination, and deriving implied predicates to the query to produce logically equivalent plans. Bushy plans provide maximum parallelism. Each logical plan is assigned a cost that is based on distinct, value-based heuristics.

The Calcite plan pruner selects the lowest-cost logical plan. Hive converts the chosen logical plan to a physical operator tree, optimizes the tree, and converts the tree to a Tez job for execution on the Hadoop cluster.

Explain plans

You can generate explain plans by running the EXPLAIN query command. An explain plan shows you the execution plan of a query by revealing the operations that occur when you run the query. Having a better understanding of the plan, you might rewrite the query or change Tez configuration parameters.

Set up the cost-based optimizer and statistics

You can use the cost-based optimizer (CBO) and statistics to develop efficient query execution plans that can improve performance. You must generate column statistics to make CBO functional.

About this task

In this task, you enable and configure the cost-based optimizer (CBO) and configure Hive to gather column statistics as well as table statistics for evaluating query performance. Column and table statistics are critical for estimating predicate selectivity and the cost of the plan. Certain advanced rewrites require column statistics.

In this task, you check, and set the following properties:

- hive.stats.autogather

  Controls collection of table-level statistics.
- hive.stats.fetch.column.stats

  Controls collection of column-level statistics.
- hive.compute.query.using.stats

  Instructs Hive to use statistics when generating query plans.

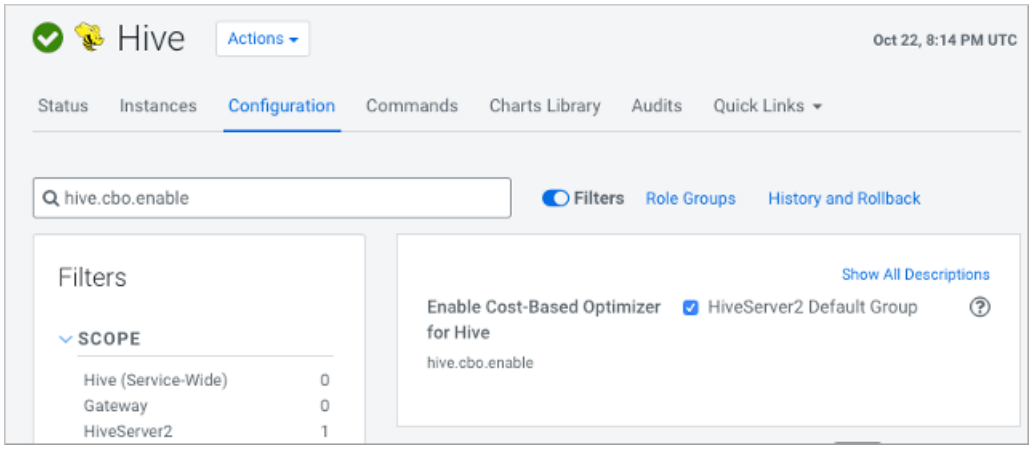
You can manually generate the table-level statistics for newly created tables and table partitions using the ANALYZE TABLE statement.

Before you begin

- The following components are running:
  - HiveServer
  - Hive Metastore
  - Hive clients
- Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

Procedure

1. In Cloudera Manager Clusters select the Hive service, for example, HIVE.
2. On the Configuration tab, search for hive.cbo.enable.



   If the property is not visible in your version of Cloudera Manager, add the property to Hive site using the Cloudera Manager Safety Valve (see links below). Set the property to enabled.
3. Accept the default (enabled), or check to enable the hive.cbo.enable property for the HiveServer Default Group.
4. Search for and enable, if necessary, hive.stats.fetch.column.stats.
5. Search for and enable, if necessary, hive.compute.query.using.stats.
6. In Cloudera Manager Clusters Hive Actions select Restart.

Related Information

Example of using the Cloudera Manager Safety Valve
Custom Configuration (about Cloudera Manager Safety Valve)

Generate and view Apache Hive statistics

You can use statistics to optimize queries for improved performance. The cost-based optimizer (CBO) also uses statistics to compare query plans and choose the best one. By viewing statistics instead of running a query, you can often get answers to your data questions faster.

About this task

This task shows how to generate different types of statistics about a table.

Procedure

1. Launch a Hive shell and log in.
2. Gather table statistics for the non-partitioned table mytable:

```
ANALYZE TABLE mytable COMPUTE STATISTICS;
```

3. View table statistics you generated:

```
DESCRIBE EXTENDED mytable;
```

4. Gather column-level statistics for the table:

```
ANALYZE TABLE mytable COMPUTE STATISTICS FOR COLUMNS;
```

5. View column statistics for the col_name column in my_table in the my_db database:

```
DESCRIBE FORMATTED my_db.my_table col_name;
```

Related Information

Apache Hive Wiki language reference
Apache Hive Wiki - Statistics in Hive

Statistics generation and viewing commands

You can manually generate table and column statistics, and then view statistics using Hive queries. By default, Hive generates table statistics, but not column statistics, which you must generate manually to make cost-based optimization (CBO) functional.

Commands for generating statistics

The following ANALYZE TABLE command generates statistics for tables and columns:

ANALYZE TABLE [table_name] COMPUTE STATISTICS;

Gathers table statistics for non-partitioned tables.

ANALYZE TABLE [table_name] PARTITION(partition_column) COMPUTE STATISTICS;

Gathers table statistics for partitioned tables.

ANALYZE TABLE [table_name] COMPUTE STATISTICS for COLUMNS [comma_separated_column_list];

Gathers column statistics for the entire table.

ANALYZE TABLE partition2 (col1="x") COMPUTE STATISTICS for COLUMNS;

Gathers statistics for the partition2 column on a table partitioned on col1 with key x.

Commands for viewing statistics

You can use the following commands to view table and column statistics:

DESCRIBE [EXTENDED] table_name;

View table statistics. The EXTENDED keyword can be used only if the hive.stats.autogather property is enabled in the hive-site.xml configuration file. Use the Cloudera Manager Safety Valve feature (see link below).

DESCRIBE FORMATTED [db_name.]table_name [column_name] [PARTITION (partition_spec)];

View column statistics.

Related Information

Example of using the Cloudera Manager Safety Valve

Custom Configuration (about Cloudera Manager Safety Valve)

Removing scratch directories

You need to know how to periodically clear scratch directories used by Apache Hive to prevent problems, such as failing jobs.

About this task

Scratch directories where Hive stores intermediate, or temporary, files accumulate too much data over time and overflow. You can configure Hive to remove scratch directories periodically and without user intervention. Using Cloudera Manager, you add the following properties as shown in the procedure:

hive.start.cleanup.scratchdir

Value: true

Cleans up the Hive scratch directory while starting the HiveServer.

hive.server2.clear.dangling.scratchdir

Value: true

Starts a thread in HiveServer to clear out the dangling directories from the file system, such as HDFS.

hive.server2.clear.dangling.scratchdir.interval

Example Value: 1800s

Procedure

1. In Cloudera Manager, click Clusters Hive on Tez Configuration . Clusters > Hive on Tez > 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. InName enter the property name and in value enter the value.