Cloudera Runtime 7.3.3

Configuring Apache Hive

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

Understanding CREATE TABLE behavior....4
Configuring legacy CREATE TABLE behavior....5
Limiting concurrent connections....6
Hive on Tez configurations....7
Configuring HiveServer high availability using a load balancer....8
Configuring the Hive Delegation Token Store....8
Adding a HiveServer role....9
Configuring the HiveServer load balancer....10
Achieving cross-cluster availability through Hive Load Balancer failover....11
Configuring HiveServer high availability using ZooKeeper....13
Generating Hive statistics....14
Setting up the cost-based optimizer and statistics....14
Generating and viewing Hive statistics....15
Statistics generation and viewing commands....16
Configuring query audit logs to include caller context....17
Removing scratch directories....18

Understanding CREATE TABLE behavior

Hive table creation has changed significantly since Hive 3 to improve useability and functionality. If you are upgrading from CDH or HDP, you must understand the changes affecting legacy table creation behavior.

Hive has changed table creation in the following ways:

- Creates ACID-compliant table, which is the default in Cloudera
- Supports simple writes and inserts
- Writes to multiple partitions
- Inserts multiple data updates in a single SELECT statement
- Eliminates the need for bucketing.

If you have an ETL pipeline that creates tables in Hive, the tables will be created as ACID. Hive now tightly controls access and performs compaction periodically on the tables. Using ACID-compliant, transactional tables causes no performance or operational overload. The way you access managed Hive tables from Spark and other clients changes. In Cloudera, access to external tables requires you to set up security access permissions.

You must understand the behavior of the CREATE TABLE statement in legacy platforms like CDH or HDP and how the behavior changes after you upgrade to Cloudera.

Before upgrading to Cloudera Base on premises

In CDH 5, CDH 6, and HDP 2, by default CREATE TABLE creates a non-ACID managed table in plain text format.

In HDP 3 and CDP 7.1.0 through 7.1.7.x, by default CREATE TABLE creates either a full ACID transactional table in ORC format or insert-only ACID transactional tables for all other table formats.

After upgrading to Cloudera Base on premises

- If you are upgrading from HDP 2, CDH 5, or CDH 6 to CDP 7.1.0 through CDP 7.1.8, by default CREATE TABLE creates a full ACID transactional table in ORC format or insert-only ACID transactional tables for all other table formats.
- If you are upgrading from HDP 3 or CDP 7.1.0 through 7.1.7.x to CDP 7.1.8, the existing behavior persists and CREATE TABLE creates either a full ACID transactional table in ORC format or insert-only ACID transactional tables for all other table formats.

Now that you understand the behavior of the CREATE TABLE statement, you can choose to modify the default table behavior by configuring certain properties. The order of preference for configuration is as follows:

Modify the default CREATE TABLE behavior

Override default behavior when creating the table

Irrespective of the database, session, or site-level settings, you can override the default table behavior by using the MANAGED or EXTERNAL keyword in the CREATE TABLE statement.

```
CREATE [MANAGED][EXTERNAL] TABLE foo (id INT);
```

Set the default table type at a database level

You can use the database property, defaultTableType=EXTERNAL or ACID to specify the default table type to be created using the CREATE TABLE statement. You can specify this property when creating the database or at a later point using the ALTER DATABASE statement. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('defaultTableType'='E
XTERNAL');
```

In this example, tables created under the test_db database using the CREATE TABLE statement creates external tables with the purge fucntionality enabled (external.table.purge = 'true').

You can also choose to configure a database to allow only external tables to be created and prevent creation of ACID tables. While creating a database, you can set the database property, EXTERNAL_TABLES_ONLY=true to ensure that only external tables are created in the database. For example:

```
CREATE DATABASE test_db WITH DBPROPERTIES ('EXTERNAL_TABLES_ONLY
'='true');
```

Set the default table type at a session level

You can configure the CREATE TABLE behavior within an existing beeline session by setting hive.create.as.external.legacy to true or false. Setting the value to true results in configuring the CREATE TABLE statement to create external tables by default.

When the session ends, the default CREATE TABLE behavior also ends.

Set the default table type at a site level

You can configure the CREATE TABLE behavior at the site level by configuring the hive.create.as.insert.only and hive.create.as.acid properties in Cloudera Manager under Hive configuration. When configured at the site level, the behavior persists from session to session. For more information, see Configuring CREATE TABLE behavior.

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 Cloudera as it did before the upgrade. You can connect to Hive using the Hive Warehouse Connector (HWC) to read Hive ACID tables from Spark. To write ACID tables to Hive from Spark, you use the HWC and HWC API. Spark creates an external table with the purge property when you do not use the HWC API. For more information, see Hive Warehouse Connector for accessing Spark data.

Related Information

Configuring legacy CREATE TABLE behavior

Configuring legacy CREATE TABLE behavior

After you upgrade to Cloudera Base on premises and migrate old tables, the legacy CREATE TABLE behavior of Hive is no longer available by default and you might want to switch to the legacy behavior. Legacy behavior might solve compatibility problems with your scripts during data migration, for example, when running ETL.

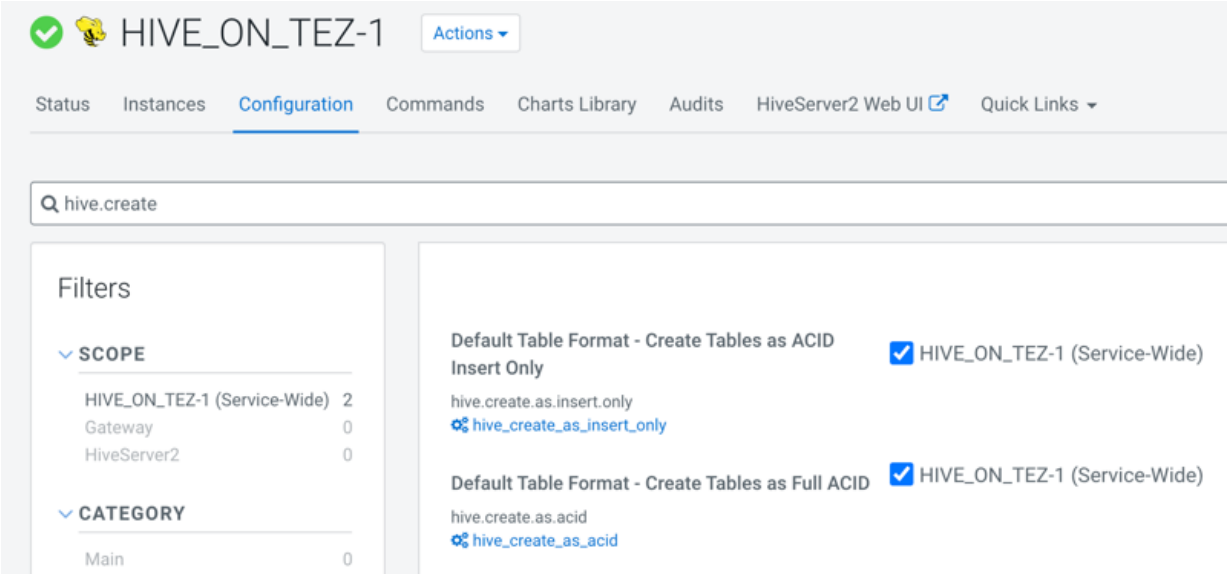
About this task

In Cloudera, running a CREATE TABLE statement by default creates a full ACID table for ORC file format and insert-only ACID table for other file formats. You can change the default behavior to use the legacy CREATE TABLE behavior. When you configure legacy behavior, CREATE TABLE creates external tables with the purge functionality enabled (external.table.purge = 'true'). Therefore, when the table is dropped, data is also deleted from the file system.

You can configure legacy CREATE TABLE behavior at the site level by configuring properties in Cloudera Manager. When configured at the site level, the behavior persists from session to session.

Procedure

1. In Cloudera Manager, click Clusters and select the Hive on Tez service.
2. From the Hive on Tez service, go to the Configuration tab and search for hive.create.



3. If the following properties are selected, clear the selection to enable legacy CREATE TABLE behavior.
   - Default Table Format - Create Tables as ACID Insert Only (hive.create.as.insert.only)
   - Default Table Format - Create Tables as Full ACID (hive.create.as.acid)

Results

Legacy behavior is enabled and the CREATE TABLE statement now creates external tables with the external.table.purge table property set to true.

Limiting 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.

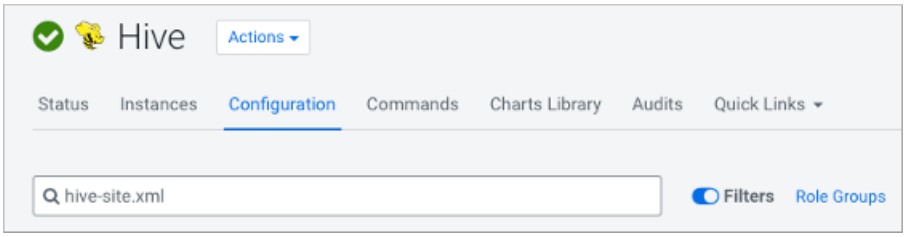
Before you begin

- The following components are running:
  - HiveServer
  - Hive Metastore

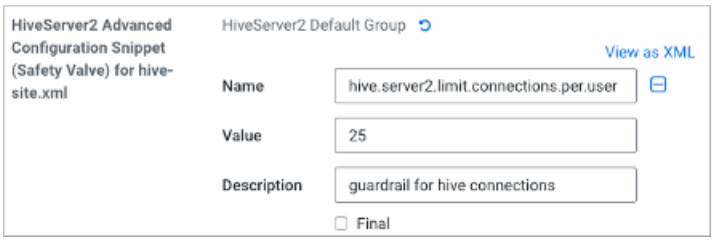
- Hive client
- Minimum Required Role: Configurator (also provided by Cluster Administrator, Full Administrator)

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.

Hive on Tez configurations

Understanding key Hive on Tez properties might help you tune performance or troubleshoot problems, such as running multiple TEZ Application Master (AM) when your default sessions configuration allows running only one. After upgrading, the number of default sessions allowed might be only one. Making Hive on Tez configuration changes is recommended for users who know what they are doing.

Property and Default Value	Description	How to Check and Configure
hive.server2.tez.default.queues (default: "default")	A list of comma separated values corresponding to YARN queues for which to maintain a Tez session pool	Use the Cloudera Manager Safety Valve. When specifying additional queues, they must already exist in YARN.
hive.server2.tez.sessions.per.default.queue (default:1)	The number of Tez sessions (DAGAppMaster) to maintain in the pool per YARN queue The total number of concurrent Tez session running can be calculated with: (Tez Sessions)total = HiveServer2instances x (default.queues) x (sessions.per.default.queue) The pooled Tez Sessions are always running, even on an idle cluster.	Use the Cloudera Manager Safety Valve. A value of 1 means only one query can run at a time

Property and Default Value	Description	How to Check and Configure
hive.server2.tez.initialize.default.sessions (default: false)	If enabled, HiveServer (HS2), at startup, will launch all necessary Tez sessions within the specified default.queues to meet the sessions .per.default.queue requirements. Disable this property if you want to improve query performance and do not want to limit all the queries to the same Tez AM pool. You can enable this property and increase the sessions.per.default.queue value based on your resource availability and query concurrency/ load on the server.	Use the Cloudera Manager Safety Valve.

Related Information

Custom Configuration (about Cloudera Manager Safety Valve)

Example of using the Cloudera Manager Safety Valve

Configuring HiveServer high availability using a load balancer

To enable high availability for multiple HiveServer (HS2) hosts, you need to know how to configure a load balancer to manage them. First, you configure the Hive Delegation Token Store, next you add HS2 roles, and finally, you configure the load balancer.

About this task

HiveServer HA does not automatically fail and retry long-running Hive queries. If any of the HS2 instances fail, all queries running on that instance fail and are not retried. The client application must re-submit the queries.

After you enable HS2 high availability, ensure that all your clients reflect the load balancer's principal in the connection string. On Kerberos-enabled clusters, you must use the load balancer's principal to connect to HS2 directly; otherwise, after you enable HS2 high availability, direct connections to HS2 instances fail.

Before you begin

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

Configuring the Hive Delegation Token Store

You need to enable Hive Delegation Token Store implementation as the first step in configuring HiveServer high availability using a load balancer. You also need to understand the interaction between Oozie and HS2 with regard to the delegation token.

About this task

Oozie needs this implementation for secure HiveServer high availability (HA). Otherwise, the Oozie server can get a delegation token from one HS2 server, but the actual query might run against another HS2 server, which does not recognize the HS2 delegation token.

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. Do not use the MemoryTokenStore. This can cause failures because one HS2 does not recognize the delegation token issued by another.

4. Add HiveServer (HS2) roles as described in the next topic.

Adding a HiveServer role

You can add a HiveServer (HS2) role to the Hive-on-Tez service, not to the Hive service.

Before you begin

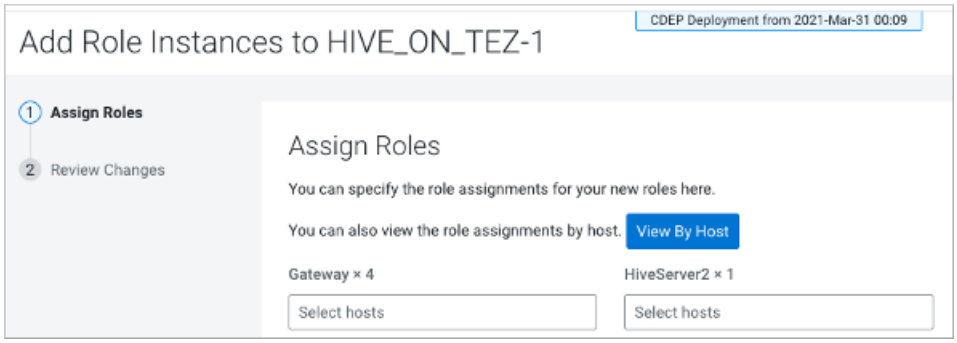
You configured the Hive Delegation Token Store.

Procedure

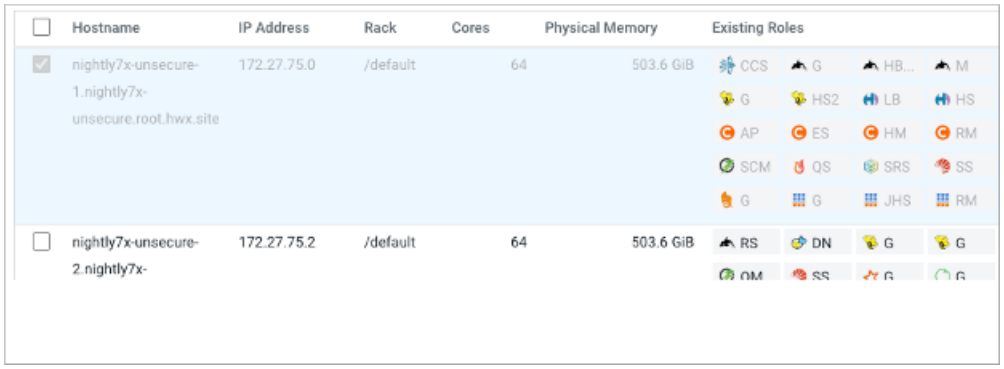
1. In Cloudera Manager, click Clusters Hive on Tez .

   Do not click Clusters Hive by mistake. Only the Hive on Tez service supports the HiveServer2 role.

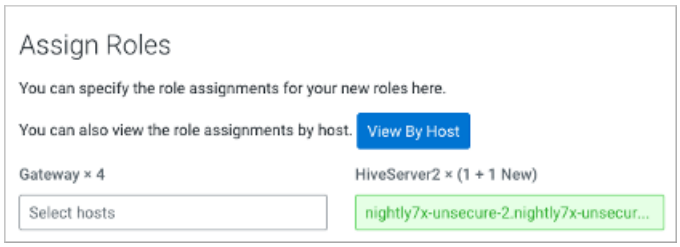
2. Click Actions Add Role Instances .



3. Click in the HiveServer2 box to select hosts.



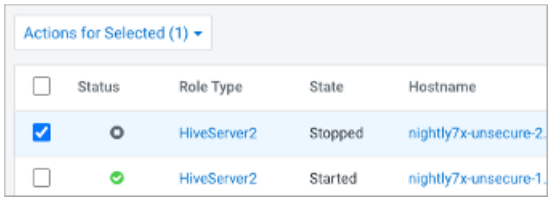
4. In the Host name column, select a host for the HiveServer2 role, and click OK.
   The selected host name you assigned the HiveServer2 role appears under HiveServer2.



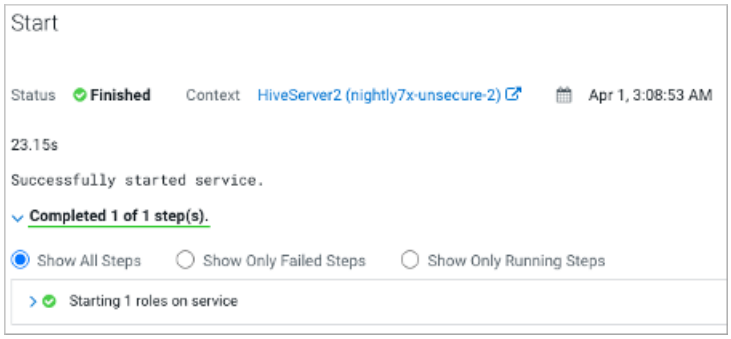
5. Click Continue.

   The new HiveServer2 role state is stopped.

6. Select the new HiveServer2 role.



7. In Actions for Selected, select Start, and then click Start to confirm.
   You see that the service successfully started.



Important: After installing Hive on Tez or upgrading the Cloudera Runtime, ensure that you refresh the Tez binaries by **Upload the Tez tar file to HDFS**. This prevents job failures while processing the Tez DAG. To refresh the Tez binaries: Cloudera Manager Tez Actions Upload Tez tar file to HDFS

Configuring the HiveServer load balancer

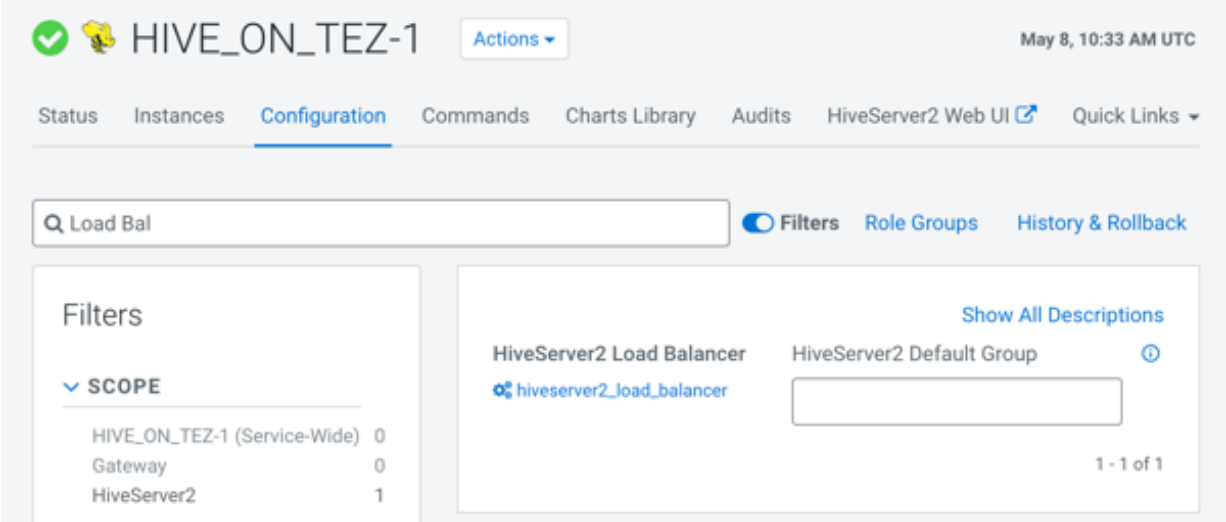
Cloudera Manager exposes the HiveServer load balancer property. You see how to access the property and set it.

Before you begin

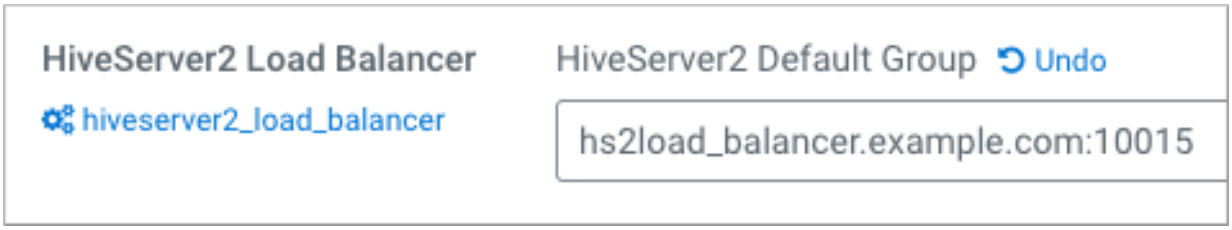
- You configured the Hive Delegation Token Store.
- You added one or more HiveServer roles.
- You must have installed an external load balancer, such as F5 or HAProxy.

Procedure

1. In Cloudera Manager, click Clusters Hive On Tez Configuration , and search for HiveServer2 Load Balancer.



2. Set the value, using the following format: <hostname>:<port number>.
For example, hs2load_balancer.example.com:10015.



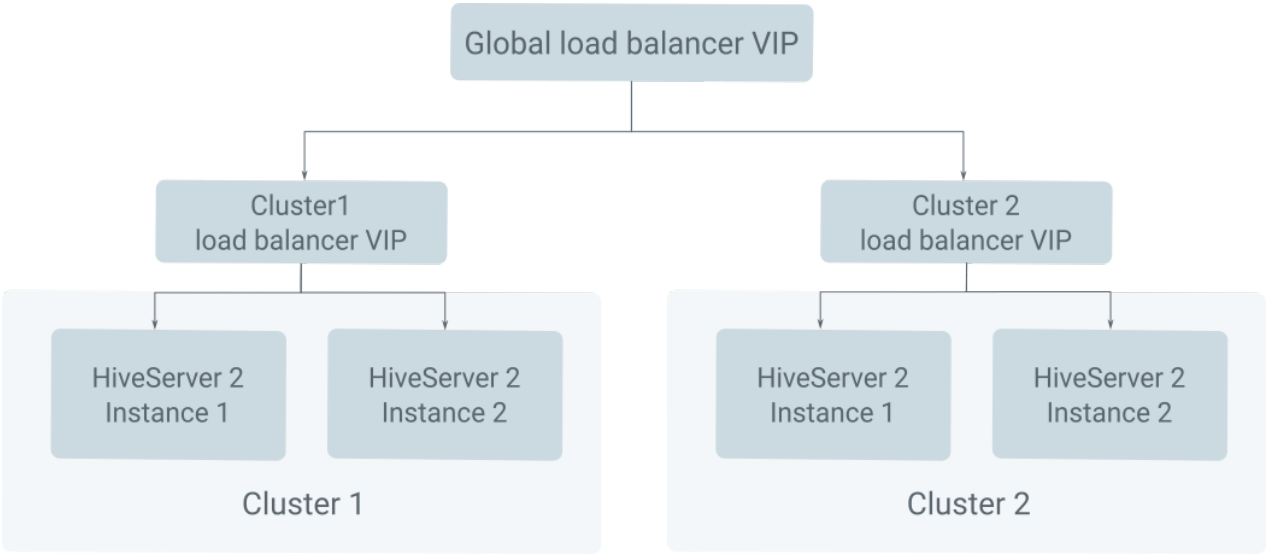
3. Save the change.

Achieving cross-cluster availability through Hive Load Balancer failover

Learn how you can manage user authentication in Hive using Kerberos, LDAP, and Knox with global and cluster-level load balancers. Also understand keytab management for multiple clusters and switching from one cluster to another using global load balancers during a primary cluster failure.

In Cloudera Manager, the Hive Load Balancer can either be set up by using an external Load Balancer or by using Zookeeper for each cluster. When configured with an external Load Balancer, Cloudera Manager manages the backend configuration that includes adding the service principal for Hive with the Load Balancer hostname to the keytab file in addition to the service principals based on the individual hiveserver2 instance hostnames.

To ensure seamless failover between two clusters, the Hive keytab needs to include in addition to the cluster level load balancer, another service principal for the global load balancer hostname. The below setup ensures that your system remains highly available and transitions smoothly during failover situations.




```
Load Balancer Details
Global Load Balancer : [***GLOBAL LOAD BALANCER VIP HOSTNAME***].example.com
Cluster1 Load Balancer :[***CLUSTER1 LOAD BALANCER VIP HOSTNAME***].example
.com
Cluster2 Load Balancer :[***CLUSTER2 LOAD BALANCER VIP HOSTNAME***].example
.com
```

Ensuring High Availability with Kerberos

The Hive service principal keytab needs to be customized for both Cluster 1 and Cluster 2. Cluster administrators need to manually manage all the service principal keytabs, rather than relying on Cloudera Manager to handle this automatically.

- The Cluster 1 Hive service principal keytab should include:

```
hive/[***GLOBAL LOAD BALANCER VIP HOSTNAME***].example.com
hive/[***CLUSTER1 LOAD BALANCER VIP HOSTNAME***].example.com
hive/hive-[***HIVESERVER2 HOSTNAME***].example.com
```

- The Cluster 2 Hive service principal keytab should include:

```
hive/[***GLOBAL LOAD BALANCER VIP HOSTNAME***].example.com
hive/[***CLUSTER2 LOAD BALANCER VIP HOSTNAME***].example.com
hive/hive-[***HIVESERVER2 HOSTNAME***].example.com
```

By ensuring these entries are present, you can achieve seamless Kerberos authentication across different clusters.

Important: To ensure proper functionality across two clusters managed by separate Cloudera Manager instances, you need to manually supply keytabs for at least one of the clusters during Kerberos configuration.

Enabling High Availability with LDAP

Kerberos authentication requires the cluster administrator to manage keytab files for service principals. However, users can authenticate by using LDAP as the authentication backend across both clusters without customizing keytab files.

To enable LDAP-based authentication, update the JDBC URL on your client interface by setting AuthMech = 3. Then, authenticate using your LDAP user account. This ensures secure access and authentication within the system. For Example:

```
jdbc:hive2://<HS2 hostname>.example.com:10000/default;SSL=1;SSLTrustStore=<p
ath>/cm-auto-global_truststore.jks;SSLTrustStorePwd=xxxx;AuthMech=3;UID=XXX;
PWD=XXXX
```

Note: Although LDAP authentication is compatible with this option, Kerberos authentication with binary transport provides the best performance.

Enabling High Availability with Knox front end

Knox acts as a gatekeeper, verifying user credentials through LDAP and granting access to Hive using Kerberos tickets. When you deploy two Knox instances in two clusters and configure the external load balancer to point to them globally, the external load balancer layer routes requests to the Knox instances, which subsequently communicates with Hive's HS2. Once Knox is enabled, the JDBC URL example is as follows:

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

Note: With Knox, it is important to understand that LDAP and Kerberos authentication cannot be used simultaneously.

Related Information

Kerberos Configuration Strategies for Cloudera

Using a custom Kerberos keytab retrieval script

Connecting to an Apache Hive endpoint through Apache Knox

Configuring HiveServer high availability using ZooKeeper

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.

Using binary transport mode in HiveServer (HS2), Knox, and Dynamic Discovery, possibly supported on your platform before upgrading to Cloudera, are not supported on Cloudera. Use alternate solutions, such as HAProxy.

Related Information

Adding a Role Instance

Generating Hive 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.

Setting 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

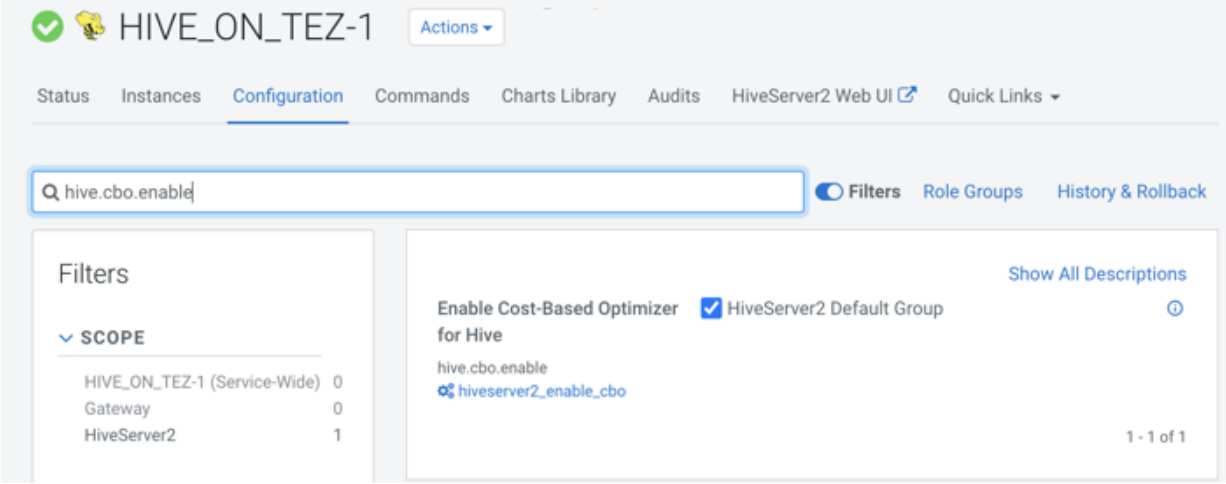
- The following components are running:
  - HiveServer

- Hive Metastore
- Hive clients

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

Procedure

1. In Cloudera Manager, click Clusters Hive On Tez Configuration .
2. In the Configuration page, search for the hive.cbo.enable property.



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. Click Actions Restart to restart the Hive on Tez service.

Related Information

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

Generating and viewing 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 or editor.
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.

Console warnings for small files

The MSCK and ANALYZE commands display a warning message in the console if the average file size for a table or partition is below the threshold. The average file size is calculated by dividing the total size by the number of files (totalSize / numFiles). If this value is less than the avgFileSize threshold, a warning is displayed so you can identify potential performance issues caused by small files.

Hive displays small-file warnings during MSCK REPAIR TABLE or ANALYZE TABLE operations to help you identify performance risks associated with small files.

The following example shows the warning message when you run the `MSCK REPAIR TABLE` command:

```
hive> MSCK REPAIR TABLE sales;
Partitions not in metastore: sales/dt=2025-01-01, sales/dt=2025-01-02
Repair: Added partition to metastore sales/dt=2025-01-01
Repair: Added partition to metastore sales/dt=2025-01-02
[MSCK] Small files detected.
[MSCK] Average file size is too small, small files exist.
Partition name: dt=2025-01-01. Small files detected: partition dt=2025-01-01
 (avgBytes=2048, files=5000, totalBytes=10240000)
[MSCK] Average file size is too small, small files exist.
Partition name: dt=2025-01-02. Small files detected: partition dt=2025-01-02
 (avgBytes=1024, files=8000, totalBytes=8192000)
OK
```

The following example shows the warning message when you run the `ANALYZE TABLE` command:

```
hive> ANALYZE TABLE sales PARTITION(dt='2025-01-01') COMPUTE STATISTICS;
...
[ANALYZE] Small files detected: partition dt=2025-01-01 (avgBytes=2048,
 files=5000, totalBytes=10240000)
OK
```

Related Information

Example of using the Cloudera Manager Safety Valve

Custom Configuration (about Cloudera Manager Safety Valve)

Configuring query audit logs to include caller context

Learn how you can configure Hive to audit partitions that are scanned as part of a Hive query. The audit information comprises the Query ID and User ID, which helps in meeting compliance requirements, such as controlling user access to data from specific regions or access only to particular time periods.

About this task

The audit information that is logged is stored in the HDFS audit log entry. You can access the audit logs from either Cloudera Manager or through the command line:

- From Cloudera Manager

  Access the log file from Clusters > HDFS-1 > NameNode Web UI > Utilities > logs > hdfs-audit.log
- From the command line:

  Access the log file from /var/log/hadoop-hdfs/hdfs-auidt.log

Procedure

1. Log in to Cloudera Manager as an administrator and go to Clusters HDFS-1 .
2. In the HDFS-1 service page, click the Configuration tab and search for the "hadoop.caller.context.enabled" property.
3. Enable the property to allow the caller context to be included in the audit logs.

   This allows additional fields to be written into HDFS NameNode audit log records for auditing coarse granularity operations.
4. Click Save Changes and restart the HDFS-1 service.

5. After the service restarts, run your Hive queries.

What to do next

You can view the audit information of the queries by accessing the hdfs-audit.log file.

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, 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, 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.